PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

Diseño de un observador robusto de blancos aéreos de alta maniobrabilidad basado en sistemas de estructura variable con modos deslizantes

Tesis para obtener el grado académico de Magíster en

Ingeniería de Control y Automatización

que presenta:

Italo Antonio Aranda Cetraro

Asesor:

Carlos Gustavo Pérez Zuñiga

Lima, 2022

Resumen

Esta tesis estudia los observadores de estados basados en sistemas de estructura variable con modos deslizantes, como una solución alterna al algoritmo de modelo múltiple interactivo (IMM) basado en filtros Kalman y filtro de partículas, para la estimación robusta de posición, velocidad y aceleración de un blanco aéreo de alta maniobrabilidad, tales como misiles antibuque o aviones de combate, a pesar de la existencia de incertidumbres o perturbaciones del modelo y utilizando mediciones de posición o velocidad con ruido angular.

En el capítulo I se efectúa el estudio del estado del arte, se expone la problemática y la solución actual a esta. Posteriormente, en el capítulo II se efectúa un estudio de los distintos modelos dinámicos y de medición de blancos aéreos de alta maniobrabilidad existentes en la literatura, proponiéndose al final del capítulo un modelo lineal incierto del blanco aéreo (misil antibuque) y presentándose una simulación de la trayectoria completa de este.

En el capítulo III se expone la teoría de sistemas de estructura variable con modos deslizantes aplicada a observadores de estado, se efectúa el diseño de los observadores más resaltantes y se presentan simulaciones de las estimaciones de la trayectoria del blanco aéreo de alta maniobrabilidad, comparándose al final del capítulo los resultados en base a criterios de desempeño establecidos. Los resultados muestran que en la ausencia de ruido los observadores "clásicos" de Edwards-Spurgeon (ESSMO), Walcott-Zak (WZSMO) y el diferenciador robusto exacto adaptativo (ARED) obtienen los mejores desempeños. Asimismo, para hacer uso de los observadores anteriormente mencionados en un ambiente de ruido angular se propone un nuevo algoritmo de filtrado, denominado diferenciador robusto exacto y uniforme filtrado (UREDF), que combina las características de filtrado estándar del diferenciador de Levant con un filtro de mediana no lineal intra pulso. Cabe resaltar que el desempeño de este algoritmo fue demostrado paralelamente al desarrollo de esta tesis en el manuscrito "Highly Maneuverable Target Tracking Under Glint Noise via Uniform Robust Exact Filtering Differentiator with Intra Pulse Median Filter", publicado en la revista IEEE "Transactions on Aerospace and Electronic Systems" y escrito por el Dr. Gustavo Pérez y el suscrito. En este manuscrito se concluye que el UREDF muestra un desempeño superior al de otros algoritmos de estimación y filtrado del estado del arte tales como el Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), Cubature Kalman Filter (CKF), Particle Filter (PF), y el Robust Student-t based Kalman Filter.

En el capítulo IV dos soluciones al problema en estudio son brindadas, siendo la primera solución (SMO1) basada en la combinación de la capacidad de filtrado del diferenciador Robusto Exacto y Uniforme filtrado (UREDF) y la robustez para estimación de variables de estado del diferenciador robusto exacto adaptativo (ARED). Por otro lado, la segunda solución (SMO2) involucra la estimación de variables de estado de un radar de traqueo pulse-doppler mediante el filtrado de las mediciones de posición y velocidad por medio de Diferenciadores Robustos Exactos y Uniformes filtrados (UREDF) y la posterior estimación de variables de estado por medio del Observador de modos deslizantes de Walcott-Zak (WZSMO). Asimismo, un diferenciador robusto exacto adaptativo es utilizado para estimar el vector de entrada de control necesario para que funcione el WZSMO.

En el capítulo V se efectuaron simulaciones en MATLAB® que comprueban que las soluciones de modos deslizantes propuestas tienen mejor capacidad de filtrado de ruido angular y robustez que el algoritmo de modelo múltiple interactivo (IMM) durante los cambios de rumbo y maniobra terminal. Finalmente, en el capítulo VI se propone la propuesta de implementación en un hardware PXI de National Instruments y en el capítulo VII se brindan conclusiones y trabajo future a realizar.

Abstract

This thesis studies state observers based on sliding mode variable structure systems, as an alternative solution to the interactive multiple model algorithm (IMM) based on Kalman and Particle Filters, for the robust estimation of position, velocity, and acceleration of a high maneuverability air target, such as anti-ship missiles or combat aircraft, despite model uncertainties or disturbances and using fire control radar's position or velocity measurements corrupted by glint noise.

In chapter I a study of the state of the art is done, exposing the problem and the current solution to it. Subsequently, in chapter II a study is made of the different dynamic and measurement models of high maneuverability air targets existing in the literature, proposing at the end of the chapter an uncertain linear model of the air target (anti-ship missile) and presenting a simulation of the complete trajectory of it.

In chapter III the theory of sliding mode variable structure systems applied to state observers is exposed, the design of the most representative observers is carried out, and simulations of the air target's estimated trajectory are conducted, comparing at the end of the chapter the results of all state observers based on established performance criteria. Results show that in the absence of noise the Edwards-Spurgeon observer (ESSMO) and Walcott-Zak observer (WZSMO) and Adaptive Robust Exact Differentiator obtain the best performances. In addition, in order to use the observers and differentiators mentioned above a new filtering algorithm is proposed, named Uniform Robust Exact filtering differentiator (UREDF), which combines Levant's standard filtering with a non-linear median intra pulse filter. It is important to state that the performance of this algorithm was demonstrated along with the writing of this thesis in the manuscript "Highly Maneuverable Target Tracking Under Glint Noise via Uniform Robust Exact Filtering Differentiator with Intra Pulse Median Filter", which has been published in the IEEE journal "Transactions on Aerospace and Electronic Systems" by Dr. Gustavo Pérez and myself. In this manuscript, it is concluded that the UREDF shows a superior performance than other state-of-the-art estimation and filtering algorithms such as the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), Cubature Kalman Filter (CKF), Particle Filter (PF), and the Robust Student-t based Kalman Filter.

In chapter IV two solutions to the studied problem are proposed, being the first solution (SMO1) based on the combination of the Uniform robust exact filtered differentiator's (UREDF) filtering capability and the Adaptive Robust Exact Differentiator's robustness, exactness, and convergence speed capabilities for state variable estimation. On the other hand, the second solution (SMO2) involves state variable estimation of a pulse-doppler tracking radar by filtering both position and doppler velocity measurements with Uniform Robust Exact Filtered Differentiators (UREDF) and estimating state variables with the Walcott-Zak Sliding Mode Observer (WZSMO). Also, an adaptive robust exact differentiator (ARED) is used to provide the estimated input control vector necessary for the WZSMO to work.

In chapter V, MATLAB® simulations were conducted, proving that the sliding mode solutions proposed in chapter IV have better glint noise filtering and robustness capabilities that the Interactive Multiple Model (IMM) algorithm during the misil's course changes and terminal maneuver. Finally, in chapter VI is proposed the implementation solution in a National Instruments' PXI and in chapter VII conclusions and future work remarks are given.

Agradecimiento

El autor agradece el financiamiento económico efectuado por la Marina de Guerra del Perú para cursar la maestría de Ingeniería de Control y Automatización y a la Pontificia Universidad Católica del Perú por los conocimientos impartidos. Asimismo, el autor agradece la asesoría brindada por el Dr. Ing. Gustavo Pérez Zúñiga para la elaboración de la presente tesis.

Índice General

Índice de Tablas

Índice de Figuras

Acrónimos

STT	Single target tracker
PT	Punto de tiro
PF	Punto futuro
LOS	Line-of-sight
IMM	Interactive Multiple Model
LSTM	Long short-term memory
CHASE	CelsiusTech High Accuracy Seaskimmer Estimator
MMPF	Multiple model Particle Filter
SMO	Sliding mode observer
SMC	Sliding mode control
GMM	Gaussian mixture model
ESSMO	Edwards-Spurgeon Sliding Mode Observer
WZSMO	Walcott-Zak Sliding Mode Observer
LQG	Linear Quadratic Gaussian
LMI	Linear Matrix Inequality
TA	Twisting algorithm
STA	Super-Twisting algorithm
STASMO	Super-Twisting algorithm Sliding Mode Observer
n-STA	Arbitrary order Super-Twisting algorithm
HOSMO	Higher-Order Sliding Mode Observer
RED	Robust Exact Differentiator
URED	Uniform Robust Exact Differentiator
ARED	Adaptive Robust Exact Differentiator
REDF	Robust Exact Filtering Differentiator
UREDF	Uniform Robust Exact Filtering Differentiator
KF	Kalman Filter
EKF	Extender Kalman Filter
UKF	Unscented Kalman Filter
CKF	Cubature Kalman Filter
PF	Particle Filter
PXI	PCI eXtensions for instrumentation
PXIe	PXI express

Nomenclatura

$\boldsymbol{x(t)}$	Vector de estados reales del modelo
$\boldsymbol{u(t)}$	Vector de entrada de control del modelo
$\boldsymbol{y(t)}$	Vector de mediciones del modelo
$\boldsymbol{w(t)}$	Ruido de proceso del modelo
$\boldsymbol{v(t)}$	Ruido de medición del modelo
$\boldsymbol{\xi(t)}$	Incertidumbres estructuradas o perturbaciones del modelo
$\boldsymbol{F_k}$	Matriz de transición de estados del modelo
$\boldsymbol{G_k}$	Matriz de entrada de control del modelo
$\boldsymbol{D_k}$	Matriz de perturbaciones
$\boldsymbol{C_k}$	Matriz de observación de estados
$\boldsymbol{\dot{x}(t)}$	Derivada del vector de estados reales del modelo
$\boldsymbol{\hat{x}(t)}$	Vector de estados restimados del observador
$\boldsymbol{\hat{u}(t)}$	Vector de entrada de control estimado del modelo
$\boldsymbol{\hat{p}(t)}$	Vector de perturbaciones estimadas del modelo
$\boldsymbol{G_l}$	Matriz de ganancias lineales del observador de modos deslizantes
$\boldsymbol{G_n}$	Matriz de ganancias no lineales del observador de modos deslizantes
$\boldsymbol{v}$	Función discontinua de control
$\boldsymbol{e_y}$	Errores de las variables de estado medibles
$\boldsymbol{L}$	Constante de Lipschitz
$\boldsymbol{z0}$	Variable de estado estimada de posición por diferenciación
$\boldsymbol{z1}$	Variable de estado estimada de velocidad por diferenciación
$\boldsymbol{z2}$	Variable de estado estimada de aceleración por diferenciación
$\boldsymbol{z3}$	Variable de estado estimada de sobre aceleración por diferenciación
$\boldsymbol{z4}$	Variable de estado estimada de la derivada de sobre aceleración por diferenciación
$\boldsymbol{Ad}$	Distancia al blanco en coordenadas polares
$\boldsymbol{Bdn}$	Acimut respecto al norte verdadero del blanco en coordenadas polares
$\boldsymbol{Ed}$	Elevación al blanco en coordenadas polares

Introducción

Los blancos aéreos de alta maniobrabilidad, tales como aviones de combate o misiles antibuque, son amenazas latentes para un buque o unidad de superficie debido a sus características inherentes de maniobrabilidad, velocidad y letalidad. Este tipo de blancos tienen la facilidad de penetrar las defensas antiaéreas del objetivo y efectuar maniobras evasivas o terminales que dificultan el traqueo efectuado por el radar de control de tiro y por consiguiente el uso de las armas antiaéreas; la neutralización o destrucción de estos blancos es preponderante para preservar la integridad de la unidad de superficie y poder cumplir con la misión encomendada, por lo que en la actualidad se utilizan sistemas de control de tiro antiaéreos sofisticados para lograrlo.

En la figura 1.1 se muestra una unidad naval de superficie tipo fragata misilera que cuenta con cuatro antenas de traqueo correspondientes a los sistemas de control de tiro que efectúan el seguimiento automático de blancos aéreos de alta maniobrabilidad, dos antenas a cada banda, una antena a proa y la otra a popa de la unidad.

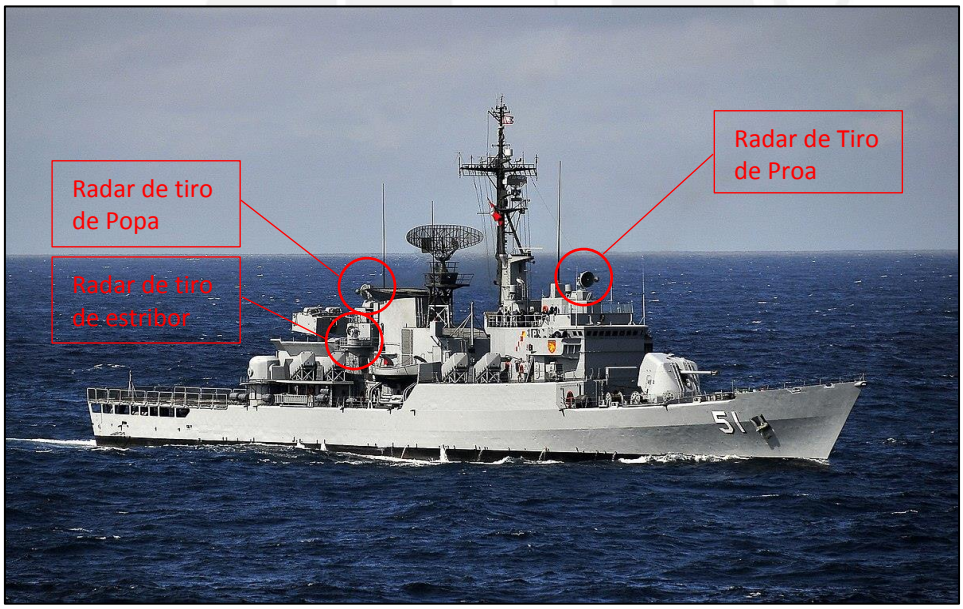

Figura 1.1.: Fragata Misilera Tipo “Lupo” con sistema de control de tiro antiaéreo
Fuente: Marina de Guerra del Perú.

De acuerdo a la literatura estudiada, los sistemas de control de tiro del estado del arte permiten la búsqueda, adquisición, traqueo, afectación de armas y neutralización o destrucción de amenazas aéreas de forma automática [1], contando por lo general con componentes de radar, componentes de control u operación y sensores ópticos adicionales. Asimismo, estos se encuentran conectados entre sí y a elementos foráneos del sistema tales como armas antiaéreas de tipo cañón.

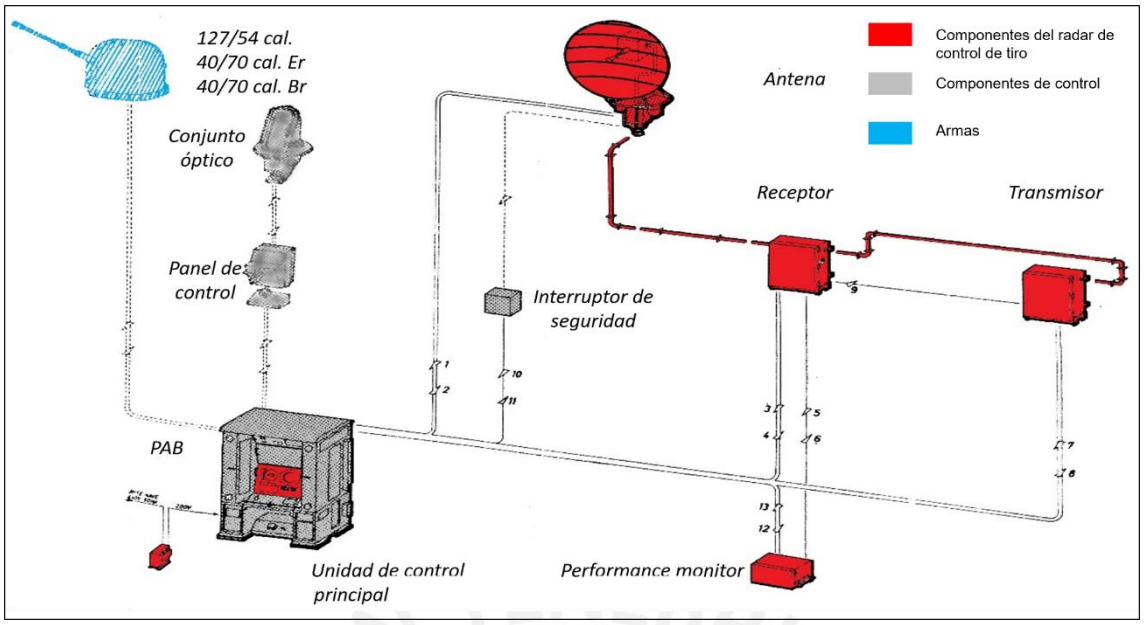

Figura 1.2.: Ejemplo de sistema de control de tiro
Fuente: Selenia Spa.

Los sistemas de control de tiro deben cumplir las siguientes tareas para efectuar la neutralización o destrucción de blancos aéreos de forma efectiva [1]:

(1) Determinación exacta y continua de la posición actual del blanco (PA), al efectuar mediciones de posición en coordenadas esféricas (azimuth, elevación y distancia) por medio de un radar de control de tiro o *Single Target Tracker (STT por sus siglas en inglés)*. En la actualidad este tipo de radares implementan técnicas de lobing secuencial de tipo monopulso[1] que les permiten obtener una mayor precisión durante el traqueo.

(2) Cómputo de la posición, rumbo, velocidad y aceleración estimada del blanco en un sistema de coordenadas estabilizado a partir de las mediciones ruidosas de radar por medio de un observador de variables de estado. Asimismo, el sistema debe compensar por el balance (roll) y cabeceo (pitch) del buque o plataforma en el que se encuentra instalado.

(3) Cómputo del tiempo de vuelo de la cabeza de combate hacia el objetivo o blanco.

(4) Determinación del punto de tiro (PT), a fin de que la cabeza de combate impacte en el blanco en el punto futuro (PF). Para el cómputo de las coordenadas del punto de tiro, se toma en consideración lo siguiente:

- Corrección por paralaje de la antena (O) y el arma (K).
- Correcciones meteorológicas y balísticas.
- Correcciones arbitrarias que son introducidas manualmente.
- Compensación por gravedad, resistencia del aire y deriva.

(5) Conversión de las coordenadas estabilizadas del punto de tiro en órdenes de ronza y elevación para el arma asignada por el sistema de control de tiro.

En la figura 1.3 se aprecia un buque o unidad de superficie y un blanco aéreo en aproximación, siendo su posición actual (D) medida por el radar del sistema de control de tiro. El objetivo de control es mantener la línea de mira o línea de puntería (LOS) apuntando al blanco sin errores de traqueo en todo momento, a fin de que el computador de tiro resuelva el denominado problema de tiro. Es decir, el computador de tiro genera la orden correspondiente al ángulo de

[1] La técnica de lobing secuencial denominada “monopulso” mantienen al blanco sobre la línea de mira del radar al extraer los errores angulares de traqueo en elevación y acimut, así como el error de traqueo en distancia en un solo pulso de energía electromagnética emitida por el radar de control de tiro [2].

acimut y elevación que el arma antiaérea, en este caso un cañón de 76mm, debe tener para poder batir al blanco en el punto futuro (T).

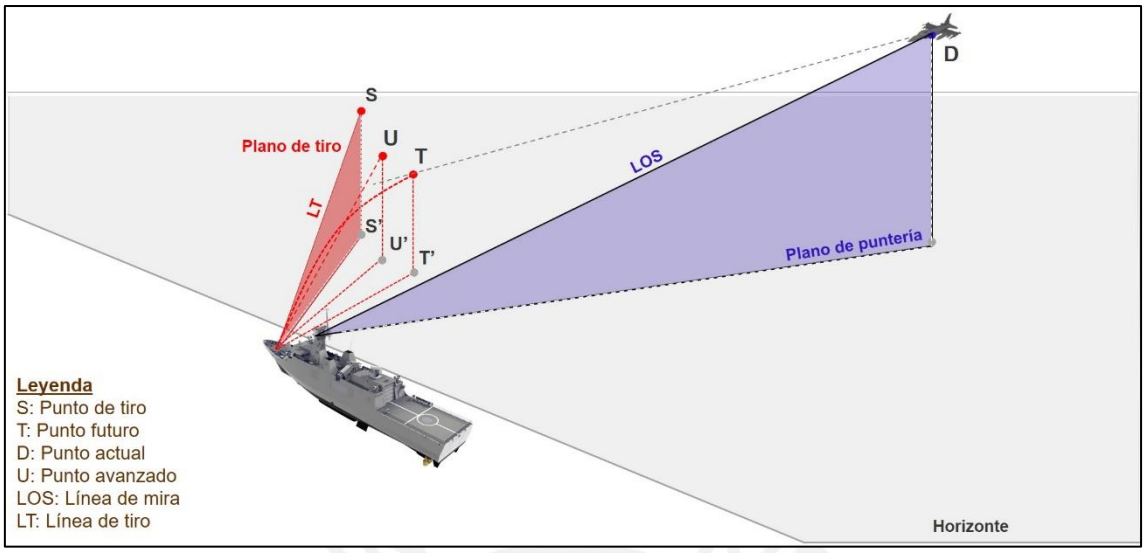

Figura 1.3.: El problema de tiro
Fuente: Elaboración propia.

De lo anterior se deduce que las tareas (3), (4) y (5) se efectuarán correctamente si es que se conoce la posición exacta del blanco en todo momento. Sin embargo, las mediciones efectuadas por el radar de control de tiro en la tarea (1) poseen ruido, por lo que los sistemas de control de tiro modernos cuentan con algoritmos de observación de estados que permiten obtener variables de posición, velocidad y aceleración estimadas con bajo ruido.

Se conoce además que los blancos de alta maniobrabilidad ejecutan maniobras a aceleraciones "g" elevadas que dificultan el traqueo y muchas veces causan divergencia de las variables de estado reales de las estimadas, por lo que existen diversos métodos propuestos para resolver este problema, cada uno de estos con un nivel de efectividad. Sin embargo, el filtro interactivo de modelos múltiples (IMM) es el más resaltante, siendo un tipo de filtro híbrido subóptimo que tiene la capacidad de adaptarse a la dinámica del blanco. El IMM, propuesto en la década de los 90, ha probado ser la mejor solución para traqueo de blancos aéreos maniobrables debido a que logra obtener la mejor relación entre complejidad y desempeño disponible en este campo [3].

Capítulo I

Estudio del estado del arte

1.1 Introducción

El estado del arte en lo que respecta a observación de variables de estado cinemáticas de un blanco aéreo de alta maniobrabilidad ha sido extensamente estudiado y lo largo de los años se han ofrecido innumerables soluciones al problema en estudio. A continuación, se hace una breve síntesis de las investigaciones realizadas en este campo, las cuales no todas han sido implementadas en sistemas de control de tiro reales, pero sirven para enfocar la presente tesis.

1.2 Estudio del estado del arte de los sistemas de observación de variables de estado de blancos aéreos

De acuerdo a la literatura estudiada, se hace una breve síntesis de los observadores mas representativos en este campo:

- **Observadores adaptativos Kalman,** siendo las soluciones más representativas descritas en [4] y [5]. En este tipo de filtros se efectúa la adaptación de parámetros online a fin de hallar las ganancias óptimas de traqueo.
- **Observadores Kalman basado en modelo con ruido de proceso correlacionado y de semi markov,** siendo estudiado e implementado extensivamente por Singer et al. [6] y Singer [7] utilizando técnicas de particionamiento con modelos múltiples. Asimismo, otra forma de implementar este tipo de filtros es utilizando bancos de filtros Kalman, cada uno contando con un peso dependiente de una función probabilística de detección de maniobras, siendo esto propuesto por Thorp [8] y Sun y Chiang [9].
- **Observadores basados en modelo de proceso Poisson de aceleración,** siendo propuesto por Lim y Farooq [10], en el cual se representa a los comandos de movimiento del piloto como maniobras determinísticas gobernadas por un proceso poisson.
- **Observadores basados en modelo de proceso de renovación de aceleración,** mostrando ser una solución alterna al modelo de proceso de markov para representar la dinámica temporal del blanco. En Sworder et al. [11] se propone un filtro de este tipo, logrando resultados satisfactorios en las simulaciones efectuadas.
- **Observadores Kalman con estimación de vector de entrada,** siendo este tipo de filtros propuesto en Chan et al. [12], Chan y Couture [13], Bogler [14] y Zhou et al. [15]. Estos observadores efectúan la estimación del vector de entrada al modelo dinámico del blanco generando un residual por el método de mínimos cuadrados. Posteriormente, la norma de este vector es utilizada en la etapa de detección de maniobra para declarar si el blanco ha efectuado un movimiento brusco, para finalmente efectuar la actualización de pesos del filtro Kalman.
- **Observador de dimensión variable,** propuesto en 1982 por Yaakov Bar-Shalom y Kailash Birmiwal [16], introduce estados adicionales al modelo de espacio de estados cuando se detecta una maniobra, teniendo como elemento de detección de la maniobra la estimación recursiva del vector aceleración. Posteriormente, Khaloozadeh y Karsaz [17] proponen un filtro de estado aumentado utilizando al vector de entrada del sistema como un estado aditivo al vector de estado original. Asimismo, en esta investigación se plantea lógica difusa (*fuzzy logic)* para mejorar el traqueo durante cambios de rumbo del blanco

- **Filtros de división de pistas,** propuesto por West y Haddsd [18], utiliza un modelo de proceso markoviano común, y para la etapa de filtrado utiliza un filtro Kalman para cada rama del algoritmo "track-splitting" convencional.
- **Filtros Pseudo-Bayesianos,** siendo propuesto en Jaffer y Gupta [19] un filtro de estimación Bayesiano recursivo para efectuar observaciones que compensen incertidumbres y en Zhang et al. [20] un filtro de primer orden pseudo-bayesiano generalizado.
- **Filtro interactivo de modelos múltiples (IMM)**, es uno de los estimadores más utilizados para resolver el problema en estudio debido a su baja complejidad computacional, propuesto en 1988 por Blom y Bar-Shalom [21]. Este tipo de estimador cuenta con múltiples filtros, cada uno de ellos adscrito a un modelo cinemático específico que representa la dinámica del blanco para un caso particular. Por tanto, el algoritmo utiliza un módulo de decisión probabilístico para detectar los cambios de cinemática del blanco y así seleccionar el filtro que se adecue mejor a la cinemática actual del blanco. Asimismo, en Liu y Chen [22] se propone un filtro interactivo de modelos múltiples que reemplaza el módulo de decisión probabilístico convencional con un algoritmo de decisión basado en lógica difusa.
- **Observadores basados en redes neuronales recurrentes,** siendo uno de los resultados más resaltantes expuesto en Gao [23], se basan en la utilización de aprendizaje profundo (*deep learning)* para predecir la trayectoria del blanco a partir del entrenamiento por medio de datos generados que contienen los movimientos de un blanco aéreo. A partir de este aprendizaje se efectúa la inferencia en tiempo real. Cabe resaltar que en los últimos años este tipo de soluciones a partir del uso de las *Long Short-term Memory* o *LSTM* por sus siglas en inglés son las más comunes, tal y como se muestra en Gao et al. [24]. Cabe resaltar que este tipo de algoritmos son difíciles de implementar en soluciones de tiempo real duro dado que por lo general la inferencia en tiempo real de la red neuronal es computacionalmente costosa dada las profundidades requeridas de la red neuronal.
- **Observadores basados en filtros de partículas de modelo múltiple (MMPF),** los cuales se basan en un método de aproximación numérico para implementar un estimador bayesiano recursivo utilizando modelos múltiples, tal y como se expone en Yu y Cheng [25]. Cabe resaltar que los filtros de partículas han sido utilizados en la arquitectura IMM conjuntamente con filtros Kalman.
- **Observador basado en norma H-infinito (H-infinity Observer),** se basan en la teoría de control robusto H-infinito al poder estimar los estados de un proceso con incertidumbres y/o perturbaciones bajo el concepto de norma H-infinita. En Tsaknakis y Athans [26] se propone un observador de estas características para el traqueo de blanco aéreos maniobrables, llegándose a la conclusión que la calibración de este tipo de observadores es crucial para poder lograr un mejor desempeño que un filtro Kalman que haya sido inicializado correctamente.
- **Observadores basados en modos deslizantes (Sliding Mode Observer),** se basan en la teoría de control robusto de modos deslizantes al poder estimar los estados de un proceso con incertidumbres y/o perturbaciones al establecer una superficie de control deslizante. En Harikumar et al. [27] se propone un observador de modos deslizantes con entrada desconocida de aceleración, obteniendo buenos resultados en la ausencia de ruido.

Descritos los distintos observadores y filtros más representativos de la literatura, en Blom y Bar-Shalom [21] y Bar-Shalom et al. [28] se exponen los beneficios de la utilización de la arquitectura interactiva de modelos múltiples (IMM) para aplicaciones de traqueo de blanco aéreos de alta maniobrabilidad de acuerdo al siguiente detalle:

- Posee la capacidad de conmutar de un modelo matemático a otro al detectar diferentes modos de comportamiento.
- Es un filtro que posee un ancho de banda variable, ideal para el traqueo de blancos de alta maniobrabilidad.
- Muestra buena relación complejidad-rendimiento.
- Sus requisitos computacionales son similares al de un algoritmo con complejidad cuadrática.
- Posee similar rendimiento a un filtro bayesiano.
- Es un algoritmo recursivo y modular.

1.2 Descripción del blanco aéreo de alta maniobrabilidad

Los blancos aéreos de alta maniobrabilidad que representan una amenaza para un buque o unidad de superficie son los aviones de combate y los misiles antibuque debido a su maniobrabilidad y facilidad de burlar los sistemas de detección propios de la unidad.

Los aviones de combate poseen la capacidad de efectuar un vuelo rasante a la superficie del mar para evitar ser detectados por los radares de alarma temprana y efectuar maniobras evasivas antes de efectuar su ataque final, normalmente efectuándolo en escuadras de 2 a 4 aviones. Asimismo, se conoce que pueden volar a velocidades del sonido de hasta Mach 2 o 1500 millas por hora (2778 km/hr) y ejecutar maniobras evasivas con un factor de carga máximo promedio de 10g. Por ejemplo, en la figura 1.4 se muestra al avión de combate F-16 *Fighting Falcon*, el cual cuenta con las siguientes características de diseño:

- Longitud: 15.027 mts
- Altura: 5.090 mts
- Velocidad máxima: 1500 mph (Mach 2)
- Envergadura: 9.449 mts
- Peso vacío: 9,207 kg
- Empuje del motor: 13,000 kg
- TOGW máximo: 21,772 kg
- Factor de carga máximo: 9 g 2

Figura 1.4.: Avión F-16 Fighting Falcon
Fuente: Lockheed Martin

Por otro lado, los misiles antibuque, por ser no tripulados, pueden soportar más fuerzas “g” que los aviones de combate, lo que les permite efectuar maniobras de cambio de rumbo más pronunciadas y al término de su trayectoria al alcanzar el objetivo efectuar maniobras evasivas o maniobra terminal para generar la pérdida del traqueo por el radar de control de tiro embarcado en la unidad. Por ejemplo, en la figura 1.5 se muestra al misil antibuque RBS-15 Gungnir de la empresa SAAB, el cual cuenta con las siguientes características de diseño:

- Longitud: 4.35 mts
- Diámetro del fuselaje: 0.50 mts
- Envergadura: 1.40 mts
- Peso (durante el vuelo): ca 650 kg
- Peso (con boosters): ca 810 kg
- Buscador: Radar active
- Velocidad máxima: 0.9 mach (sub sónico)
- Alcance máximo: >300 km
- Peso de la cabeza de combate: ~200 kg
- Trayectoria: Waypoints en 3D multiples
- Navegación: INS y anti jamming GPS3
- Factor de carga máximo: 45 g’s

[2] Datos técnicos extraídos de https://www.lockheedmartin.com/en-us/products/f-16.html

[3] Datos técnicos extraídos de https://saab.com/air/weapon-systems/air-to-surface-missile-systems/rbs15-family/

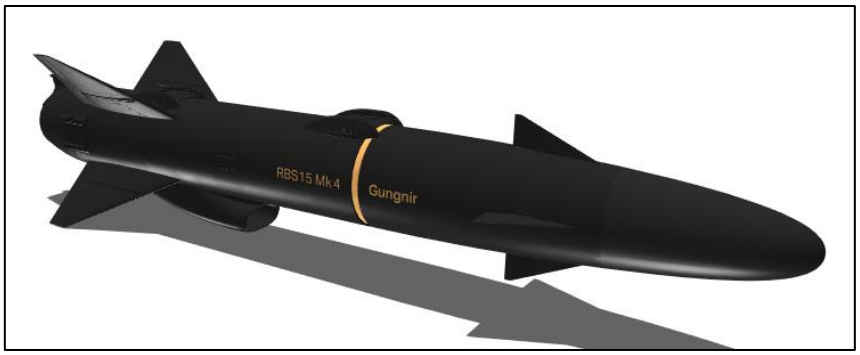

Figura 1.5.: Misil antibuque RBS-15 Gungnir
Fuente: SAAB.

De las características anteriormente presentadas se puede deducir que los factores de carga de los misiles antibuque son mayores que el de los aviones de combate, lo cual permite efectuar maniobras evasivas a mayores fuerzas “g” y con duraciones de tiempo mayores. Asimismo, el problema del traqueo de misiles antibuque radica en que poseen una mínima área reflectora de radar, por lo que son difíciles de radarizar, y que al volar a pocos metros por encima del nivel del mar generan ruido de medición en las variables de posición o velocidad con características de distribución no gaussiana denominado ruido angular. Por lo anteriormente expuesto, para la presente tesis se considerará al misil antibuque como el blanco aéreo de alta maniobrabilidad a modelar debido a que este representa el peor escenario de traqueo para un radar de control de tiro.

Los aviones de combate y misiles antibuque presentan una dinámica incierta debido a que se caracteriza por contar con dos tipos de comportamiento mutuamente excluyentes. Por un lado, la dinámica de este tipo de blancos aéreos puede ser lineal, cada vez que estos se movilizan de punto a punto cumpliendo una trayectoria lineal. Sin embargo, la dinámica puede cambiar bruscamente de lineal a no lineal, cada vez que los blancos aéreos efectúen cambios de rumbo a fuerzas “g” elevadas o realicen maniobras evasivas con características aleatorias, denominadas comúnmente como “maniobras terminales”, durante la fase final de ataque, con la finalidad de evadir las defensas antiaéreas del objetivo al que pretenden destruir o neutralizar. Por ello, el algoritmo interactivo de modelos múltiples (IMM) presenta mejores resultados que otras soluciones debido a que utiliza un mayor número de modelos del blanco en paralelo, lo que le permite contemplar un mayor número de posibles dinámicas de este. Sin embargo, el éxito de este algoritmo radica en que los modelos de las posibles dinámicas sean exactos.

1.3 Problemática de la observación de estados de un blanco aéreo de alta maniobrabilidad

1.3.1 Observación de variables de estado con ruido angular (Glint)

Al efectuar las mediciones de posición del blanco aéreo por medio de un radar a pulsos, se asume que el punto de referencia de traqueo es el centro de masa de la distribución de reflectividad del blanco [1]. Dado que la forma del blanco es irregular por diseño, el ruido angular o *glint noise* es un cambio aparente de la posición del blanco respecto a este punto de referencia dado que al interactuar o rebotar el frente de onda del pulso de radar con la superficie del blanco se generan interferencias dadas las irregularidades propias de la superficie [29]. Patterson y Ashley [30] caracterizan al ruido angular como fluctuaciones de la medición del ángulo de arribo de las ondas electromagnéticas reflejadas transmitidas por un sistema de radar.

El ruido angular posee una distribución espectral de energía que es dependiente de la frecuencia de transmisión del radar y el movimiento aleatorio del blanco respecto al buque traqueador. Por ejemplo, un radar de control de tiro en banda X puede transmitir una onda electromagnética de 8 a 12.5 GHz. Skolnik [1] presenta la siguiente fórmula para representar la forma espectral del ruido angular:

$$N(f) = {\sigma_{ang}}^2 \frac{2B}{\pi(B^2 + f^2)}$$

Donde $N(f)$ es la densidad espectral de potencia del ruido en mW/Hz, B es el ancho de banda del ruido en Hz y f es la frecuencia de transmisión en Hz. En la siguiente figura se aprecia la evolución en el tiempo de datos obtenidos en experimentos del ruido angular generado al traquear un misil tipo BQM-34A, observándose que se generan picos altos de ruido que sobresalen de la media del ruido, siendo este tipo de distribución no gaussiana conocida como distribución de cola larga o *long-tailed distribution*.

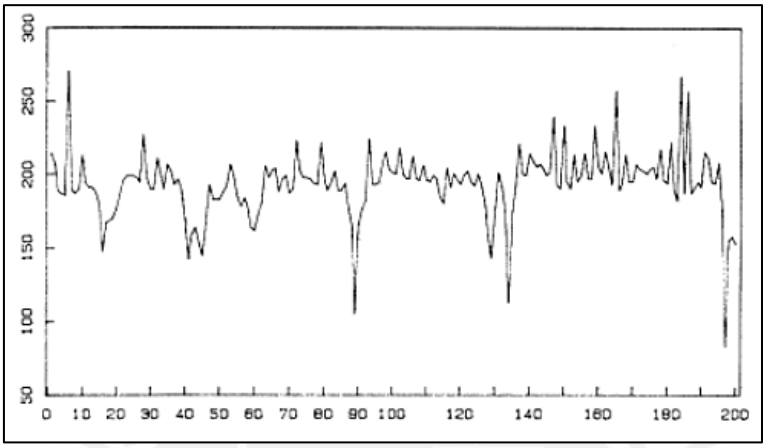

Figura 1.6.: Ruido glint del misil BQM-34A.
Fuente: Kim et al. [31]

Debido a los picos altos aleatorios que se generan, el ruido glint cuenta con una función de densidad probabilística no gaussiana [29,30,31], por lo que usualmente para efectos de modelamiento se utiliza el modelo estadístico por distribución de Student, el modelo de distribuciones gaussianas mixtas y el modelo mixto de distribución Gaussiana y Laplaciana [31], siendo sus gráficos cuantil-cuantil o *Q-Q plot* mostrados en la figura 1.7; Patterson y Ashley [30] exponen los modelos de ruido angular cuadrático y cúbico, presentando la distribución simulada que se muestra en la figura 1.8. Según lo descrito por Kim et al. [31] el modelo de distribuciones gaussianas mixtas es el modelo más eficiente computacionalmente. Asimismo, se conoce que Daeipour y Bar-Shalom [32] modelan el ruido angular como una distribución guassiana mixta para aplicación en una arquitectura de modelos múltiples. Por tanto, la probabilidad mixta del modelo de distribución gaussiana mixta es representada como $p(x) = (1 - \varepsilon)p_{g_1} + \varepsilon p_{g_2}$, donde ε es la probabilidad glint, $p_{g_1} \sim N(0, {\sigma_1}^2)$ y $p_{g_2} \sim N(0, {\sigma_2}^2)$ son distribuciones gaussianas con media cero y desviaciones estándar distintas.

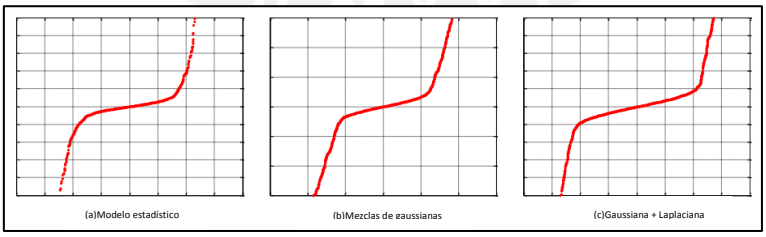

Figura 1.7.: Gráficas Q-Q del ruido glint.
Fuente: Kim et al. [31]

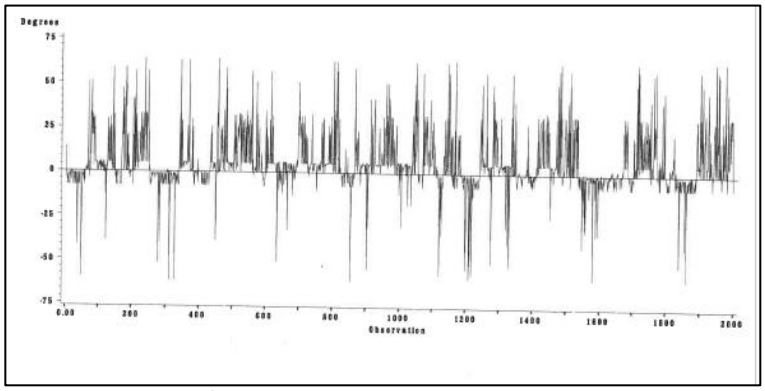

Figura 1.8.: Simulación de ruido glint.
Fuente: Patterson y Ashley [30]

1.3.2 Observación de variables de estado a baja cota o altura

Cuando el blanco aéreo, un misil sea-skimming o una aeronave de combate, vuela a pocos metros por encima del nivel del mar con la finalidad de evitar ser detectado por el radar de un buque traqueador se genera un fenómeno llamado multi trayecto o *multipath*, en el cual los ecos de retorno del blanco llegan a la antena receptora por distintos trayectos. Los trayectos que no corresponden a la línea imaginaria que une a la antena de radar con el blanco son generados a partir del rebote de ondas electromagnéticas en distintos puntos de la superficie del mar, generando lecturas erróneas de posición [1], tal y como se muestra en la figura 1.9. El fenómeno de multi trayecto o efecto imagen normalmente puede ser mitigado diseñando radares con lóbulos de energía bastante estrechos o de tipo *pencil beam* [1] y por medio de software que permita la observación de estados y filtrado de las mediciones de radar.

Por ello, empresas como SAAB del país de Suecia ofrecen productos tales como el radar de control de tiro CEROS 2000 con el algoritmo denominado “CelsiusTech High Accuracy Seaskimmer Estimator (CHASE)”, el cual permite obtener errores angulares de estimación de hasta 0.4 mili radianes en mares agitados a baja cota.

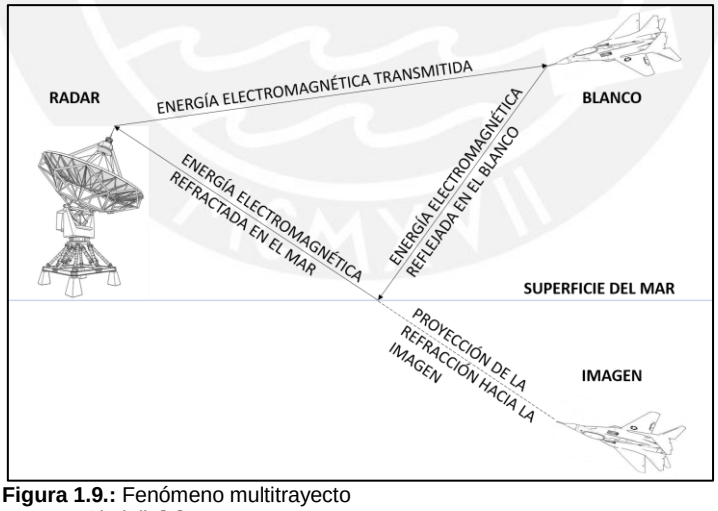

Figura 1.9.: Fenómeno multitrayecto
Fuente: Skolnik [1]

Según lo descrito por Skolnik [1] existen tres métodos fundamentales para predecir el fenómeno de multi trayecto, los cuales dependerán si el rebote o imagen es detectado por la antena a corto, medio o largo alcance.

Para el caso de detección a largo alcance implica que el ruido haya sido detectado por el lóbulo principal del radar el ruido de multi trayecto queda representado por un tipo de ruido angular:

$$e = 2h\frac{\rho^2 + \rho cos(\emptyset)}{1 + \rho^2 + 2\rho cos(\emptyset)}$$

Donde e es el error por multi trayecto en metros, ρ es el coeficiente de reflexión de la superficie del mar, h es la altura del blanco y $\emptyset$ es la fase relativa determinada por la geometría de la trayectoria de la señal que rebota. Por otro lado, cuando se detecta el fenómeno a corto alcance, esto quiere decir que la imagen se encuentra por debajo del lóbulo principal de radar y ha sido detectada por los lóbulos laterales; el error multi trayecto es cíclico, casi senoidal y con un valor RMS determinado por la siguiente formula [1]:

$$\sigma_e = \frac{\rho\theta_B}{\sqrt{8G_{se}(pico)}}$$

Donde σ_e es el valor RMS del error en elevación por multi trayecto en dBm, θ_B es el ancho de haz de la antena en dBm (solo el trayecto de ida), ρ coeficiente de reflexión de la superficie del mar y G_{se} es la relación de potencia pico entre el lóbulo principal y el lóbulo lateral en el ángulo de arribo de la señal de la imagen. Asimismo, la razón cíclica del error multi trayecto puede ser calculada de la siguiente forma:

$$f_m = \frac{2hE}{\lambda}$$

Donde f_m es la frecuencia del error multi trayecto cíclico, E es la razón de cambio de la elevación observada por el radar en rad/s y λ es la longitud de onda del pulso de radar en metros.

Finalmente, para el caso del rango intermedio el error multi trayecto es difícil de aproximar, dado que esta región está en el rango de detección no lineal del patrón del lóbulo de radar [1], por lo que usualmente se utiliza la figura 1.10 para aproximarlo.

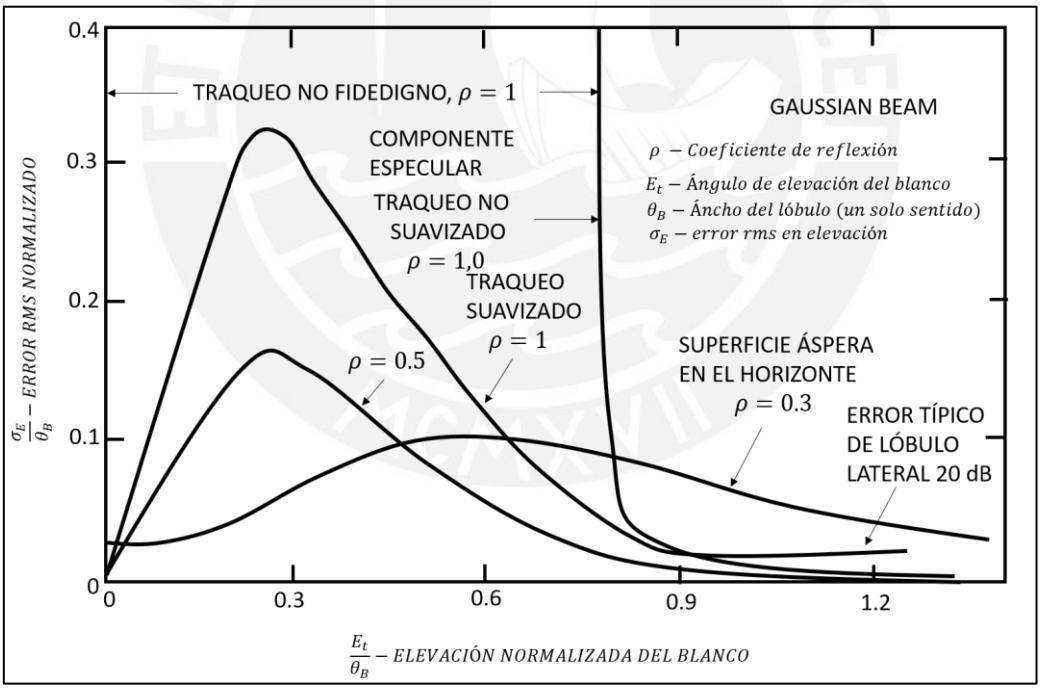

Figura 1.10.: Error rms multi trayecto a mediano alcance.
Fuente: Skolnik [1]

1.3.3 Observación de estados cuando las mediciones de radar no se encuentran disponibles

Esto sucede cuando existe un objeto que bloquea la línea de mira del radar respecto al blanco, denegando la obtención de mediciones por períodos de tiempo indeterminados. Por ejemplo, suele suceder que el blanco pase por la popa del buque o unidad de superficie y su mástil bloquee la línea de mira, siendo esto resuelto por lo general con una modalidad denominada “traqueo memoria”. Asimismo, puede suceder que exista un error angular entre el eje radio eléctrico de la antena respecto al eje mecánico de esta, que por lo general es resuelto al efectuar la colimación

de la antena, que genere un sesgo en las mediciones. Por otro lado, los sesgos pueden ser generados por el avión de combate o misil antibuque al utilizar contramedidas electrónicas tales como jamming de tipo "robo de compuerta"[4].

Por otro lado, no siempre se dispondrá de radares de control de tiro que posean la capacidad de medir la velocidad por medio del efecto doppler, por lo que la observación de variables de estado del blanco dependerá solo de la medición ruidosa de posición del blanco y por consiguiente se tendrá menos información para elaborar una solución de observación precisa.

1.3.4 Observación de estados con dinámicas inciertas

Los blancos aéreos de alta maniobrabilidad presentan dinámica lineal durante el vuelo de medio curso (mid-course) y dinámica no lineal durante maniobras de cambio de rumbo, maniobras evasivas y maniobras terminales, siendo difícil capturar todas sus posibles dinámicas utilizando uno o un grupo de modelos cinemáticos. En adición, los modelos cinemáticos utilizados en su mayoría son lineales o no lineales linealizados, por lo que siempre existirá una incertidumbre del modelo matemático y esto generará errores de traqueo.

1.4 Solución actual a la problemática

La solución actual a la problemática presentada en el apartado anterior involucra los siguientes aspectos:

- Contar con un sistema de control de tiro con radar de tipo Single Target Tracker (STT) con técnica de escansión monopulso. Esto permite obtener la mejor exactitud de medición de la posición del blanco, mejor inmunidad al ruido y mejor rechazo a contramedidas electrónicas tales como el "robo de compuerta".
- Contar con estimadores estocásticos de tipo Kalman, filtro de partículas, entre otros, con arquitecturas de modelo múltiple interactivo (IMM), los cuales no son computacionalmente costos a comparación de otros algoritmos del estudio del arte. Sin embargo, el algoritmo de modelo múltiple interactivo (IMM) tiene un rendimiento razonable en el traqueo de blancos de alta maniobrabilidad solo si el comportamiento dinámico del blanco correlaciona a alguno de los modelos cargados en el algoritmo. Cabe resaltar que la carga computacional se incrementa exponencialmente al aumentar el número de modelos en paralelo [20].

Si bien es cierto, existe un amplio estudio e implementación del algoritmo IMM, se desconoce su performance durante el seguimiento a bajas alturas y en maniobras terminales con ruido angular, dado que no se ha podido encontrar simulaciones en la literatura que muestren este tipo de escenarios.

La estructura básica del filtro IMM es mostrada en la figura 1.11, siendo el funcionamiento descrito en Genovese [33] resumido a continuación:

- Los exponentes 1 y 2 en los estados estimados de la iteración anterior $\hat{X}^1$ y $\hat{X}^2$ representan la hipótesis de que la cinemática del modelo 1 describe la cinemática real del blanco y la hipótesis de que la cinemática del modelo 2 describe la cinemática del blanco, respectivamente. Estos estados estimados son mezclados en el bloque de interacción de estados antes de la actualización de estados, utilizando las probabilidades condicionales del modelo $\tilde{\mu}^{i|j}$; estas probabilidades condicionales del modelo son calculadas mediante las probabilidades condicionales de la actualización anterior y la matriz de conmutación de estados *a priori*.
- Posteriormente, los estados estimados mezclados $\hat{\mathrm{X}}^{01}$ y $\hat{\mathrm{X}}^{02}$ son actualizados en los filtros M^1 y M^2, respectivamente, utilizando los vectores de innovación Z^1 y Z^2. Estos vectores de innovación, conjuntamente con la matriz de covarianza de innovaciones permite calcular las

[4] El robo de compuerta es una contramedida electrónica generada por los aviones de combate y misiles antibuque en la cual el avión de combate o misil emite un pulso de las mismas características que el pulso de radar propio, pero con un retardo. Este retardo es aumentado progresivamente para que la compuerta de traqueo en distancia del radar se enganche en el pulso y lo siga. Esto genera que la compuerta de traqueo se "enganche" en el falso retorno y se pierda el traqueo del retorno verdadero.

posibilidades Λ^1 y Λ^2. Cabe resaltar que cada filtro contará con un modelo cinemático distinto del blanco. Por ejemplo, el primer filtro puede ser un filtro Kalman Lineal con un modelo lineal del blanco y el segundo filtro un filtro Kalman Extendido con un modelo cinemático no lineal linealizado.

- Las posibilidades Λ^1 y Λ^2, las probabilidades condicionales anteriores del modelo $\tilde{\mu}^{i|j}$ y la matriz de conmutación de estados son utilizadas posteriormente para actualizar las probabilidades del modelo.
- Finalmente, al actualizar las probabilidades del modelo se efectúa una suma pesada en el bloque de combinación de estimación de estados.

A continuación, se detalla el funcionamiento de la arquitectura de modelos múltiples de forma matemática. Sea el vector z_k de mediciones de radar en coordenadas esféricas (azimuth, elevación y distancia), el vector de estados estimados $\hat{x}_j(k|k)$ del modo j en el instante k condicionado a la medición de radar z_k, donde un modo es considerado como algún tipo de filtro kalman lineal o no lineal, la matriz de covarianza $P_j(k|k)$, la probabilidad de modo $\mu_j(k)$, la probabilidad mixta $\mu_{i|j}(k|k)$ y $\Lambda_j(k)$ es la función de probabilidad del modo j, el algoritmo básico de IMM planteado en [28] se implementa de la siguiente manera:

Etapa de interacción

Para $\forall\ i,j\ \in\ M_f,\ \mu_{i|j}(k-1|k-1) = \frac{1}{\bar{c}_j} p_{ij} u_i(k-1)$, donde $\bar{c}_j$ es un factor de normalización,

$$\bar{c}_j = \sum p_{ij} u_i(k-1)$$

Las condiciones mixtas iniciales para el vector de estados estimados $\hat{x}_{0j}(k-1|k-1)$ y matriz de covarianza $P_{0|j}(k-1|k-1)$ del filtro de modo número j son:

$$\hat{x}_{0j}(k-1|k-1) = \sum_i \hat{x}_i(k-1|k-1) u_{i|j}(k-1|k-1)$$

$$P_{0j}(k-1|k-1) = \sum_i (P_i(k-1|k-1) + [\hat{x}_i(k-1|k-1) - \hat{x}_{0j}(k-1|k-1)][\hat{x}_i(k-1|k-1) - \hat{x}_{0j}]^T)(u_{i|j}(k-1|k-1))$$

Etapa de Filtrado

Para $\forall\ j\ \in\ M_f$,

$$\hat{x}_j(k \mid k-1) = F_j(k-1)\hat{x}_{0j}(k-1 \mid k-1) + \Gamma_j(k-1)\bar{v}_j(k-1)$$
$$P_j(k \mid k-1) = F_j(k-1)P_{0j}(k-1 \mid k-1)F_j(k-1)^T + \Gamma_j(k-1)Q_j(k-1)\Gamma_j(k-1)^T$$
$$\hat{x}_j(k \mid k) = \hat{x}_j(k \mid k-1) + W_j(k) r_j(k)$$
$$P_j(k \mid k) = P_j(k \mid k-1) - W_j(k)S_j(k)W_j(k)^T\ r_j(k) = z(k) - \hat{z}_j(k \mid k-1)\ (residual)$$
$$\hat{z}_j(k \mid k-1) = H_j(k)\hat{x}_j(k \mid k-1)\ (predicción\ de\ las\ mediciones)$$
$$s_j(k) = H_j(k)P_j(k \mid k-1)H_j(k)^T + R_j(k)\ (covarianza\ residual)$$
$$W_j(k) = P_j(k \mid k-1)H_j(k)^T S_j(k)^{-1}\ (ganancias\ del\ filtro)$$
$$\Lambda_j(k) = N(r_j(k); 0, S_j(k))\ (función\ de\ probabilidad)$$
$$\mu_j(k) = \frac{1}{c}\Lambda_j(k) \sum_i p_{ij}\mu_j(k-1)\ (probabilidad\ de\ modo)$$

Etapa de combinación

Para $\forall\ j\ \in\ M_f$,

$$\hat{x}_j(k \mid k) = \sum_j \hat{x}_j(k \mid k)\, \mu_j(k)$$

$$P(k \mid k) = \sum_j \left(P_j(k \mid k) + \left[\hat{x}_j(k \mid k) - \hat{x}(k \mid k)\right]\left[\hat{x}_j(k \mid k) - \hat{x}(k \mid k)\right]^T\right) \mu_j(k)$$

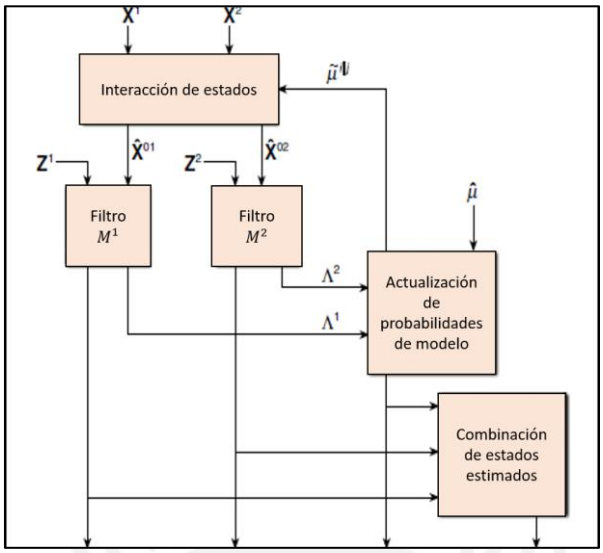

Figura 1.11.: Estructura IMM
Fuente: Genovese [33]

1.5 Propuesta de solución

De acuerdo a lo descrito en los apartados anteriores, se propone el diseño de un observador basado en un sistema de estructura variable de modos deslizantes que posea una robustez adecuada para la estimación de parámetros de posición, velocidad y aceleración de un blanco aéreo de alta maniobrabilidad en presencia de perturbaciones no lineales. Por lo tanto, esta solución permitirá mitigar las problemáticas mencionadas de los métodos encontrados en la literatura actual.

La propuesta de solución a ser elaborada, en base a lo que se describirá en los capítulos II y III, deberá abarcar lo siguiente:

- Deducir un modelo dinámico del blanco que permita modelar las no linealidades del blanco por medio de perturbaciones que se suscitan durante los cambios de rumbo, maniobras evasivas y maniobras terminales de este. Asimismo, el modelo de mediciones deberá representar las mediciones de posición del blanco en coordenadas esféricas de acimut, elevación y distancia efectuadas por el radar de control de tiro, contando con una distribución no gaussiana que presente al fenómeno de ruido angular
- La entrada u_k del sistema en estudio será considerada como la derivada de la aceleración o sobre aceleración, la cual es una magnitud física que guarda relación directa con los efectos que causan los comandos de movimiento del piloto de la aeronave o los comandos de movimiento del elaborador de órdenes de guiado del misil sobre el blanco. Dado que la entrada al sistema u_k no es medible deberá ser estimada de forma recursiva por algún método apropiado.
- Al contar con el vector de mediciones y_k y el vector de entrada estimado $\hat{u}_k$, se deberá proponer el diseño de un observador robusto de modos deslizantes (SMO) que permita no perder el traqueo durante las trayectorias no lineales del blanco.

1.6 Objetivo general

Efectuar el diseño de un observador de estados basado en sistemas de estructura variable con modos deslizantes para blancos aéreos de alta maniobrabilidad, a fin de poder comparar su desempeño con el algoritmo interactivo de modelos múltiples (IMM) basado en filtros Kalman.

1.6.1 Objetivos específicos

Para poder alcanzar este objetivo es necesario realizar lo siguiente:

- Proponer un modelo cinemático y modelo de mediciones que represente a un blanco aéreo de alta maniobrabilidad con dinámica no lineal y ruido de medición con distribución no gaussiana.
- Diseñar un observador robusto por medio de sistemas de estructura variable con modos deslizantes (SMO) que permita estimar las variables de estado de posición, velocidad y aceleración en tres dimensiones.
- Diseñar un estimador del vector de entrada de control u_k.
- Comparar el desempeño de los observadores diseñados por medio de simulaciones con y sin ruido angular.
- Proponer una solución de modos deslizantes que permita efectuar una comparación de desempeño con la arquitectura IMM.
- Comparar el desempeño de la propuesta de solución con la arquitectura IMM por medio de simulaciones.
- Proponer una propuesta de implementación en un hardware de tiempo real.

1.7 Estructura de la tesis de maestría

El contenido de la presente tesis ha sido organizado de la siguiente forma:

- **Capítulo II:** Contiene el modelamiento del sistema en estudio, así como la trayectoria simulada del blanco aéreo de alta maniobrabilidad que deberá ser estimada en el capítulo III por los observadores propuestos.
- **Capítulo III:** Contiene el diseño de todos los observadores de estados basados en sistemas de estructura variable con modos deslizantes propuestos.
- **Capítulo IV:** Plantea las propuestas de solución.
- **Capítulo V:** Efectúa la comparación de las propuestas de solución con el algoritmo IMM por medio de simulaciones.
- **Capítulo VI:** Contiene la propuesta de implementación en un hardware PXI de National Instruments.
- Finalmente, en las últimas secciones se cierra el presente manuscrito con las conclusiones y recomendaciones del trabajo a futuro por realizar.

Capítulo II

Modelamiento del blanco aéreo de alta maniobrabilidad

2.1 Introducción

Los modelos matemáticos utilizados para representar el comportamiento de un blanco aéreo en el espacio constan de un modelo dinámico y un modelo de medición. Los modelos dinámicos utilizados en los sistemas de seguimiento automático del estado del arte consideran al blanco aéreo como una partícula en el espacio, a pesar de que este en la realidad cuenta con una forma y una orientación determinada [34]. Estos modelos representan la evolución cinemática de la posición y sus derivadas respecto al tiempo a causa de una fuerza externa que genera el movimiento del mismo. Asimismo, el modelo de medición del blanco representa al radar como tal; un elemento del sistema de control de tiro que brinda datos de posición del blanco en coordenadas esféricas, y en algunos casos, datos de velocidad por medio del efecto Doppler. A continuación, se brinda una breve introducción a los modelos existentes en la literatura de blancos aéreos a fin de poder contar con elementos teóricos para poder proponer un modelo adecuado para la presente tesis.

2.2 Modelamiento de blancos aéreos

Sea el modelo dinámico y modelo de medición del blanco aéreo en su forma general:

$$\dot{x}(t) = f(x(t), u(t), w(t)) \tag{2.1a}$$

$$y(t) = h\big(x(t)\big) + v(t) \tag{2.1b}$$

donde (2.1a) y (2.2b) representan el modelo dinámico y el modelo de medición del blanco, respectivamente. Asimismo, $x(t), y(t)\ y\ u(t)$ son los vectores de estados, vector de mediciones y vector entrada de control en el instante de tiempo t, respectivamente, $w(t)$ y $v(t)$ representan los vectores de ruido de proceso y de medición, respectivamente, y $f\ y\ h$ representan funciones variantes en el tiempo, las cuales pueden ser lineales o no lineales dependiendo del modelo utilizado. Cabe resaltar que la función h es no lineal si las mediciones de radar se efectúan en coordenadas esféricas de acimut, elevación y distancia.

En Xiang Rong Li y Vesselin Jilkov [34] se efectúa una investigación extensa de modelos matemáticos de blancos aéreos, por lo que a continuación se exponen los más resaltantes y que son de interés.

2.2.1 Modelos dinámicos de blancos no maniobrables o de velocidad constante (modelo CV)

Sea el vector de estados $x(t) = [\,x \quad \dot{x} \quad y \quad \dot{y} \quad z \quad \dot{z}]'$ que representa los estados del blanco de posición y velocidad en coordenadas cartesianas, un modelo dinámico no maniobrable cumple $\dot{x}(t) = w(t) \cong 0,$ donde $w(t)$ es ruido blanco de proceso y que representa errores de modelamiento [34]. El modelo dinámico no maniobrable a velocidad constante (modelo CV) se puede representar de la siguiente forma:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} w(t) \quad (2.3)$$

Donde $x(t) = [x \quad \dot{x} \quad y \quad \dot{y} \quad z]'$ representa el vector de estados y $w(t) = [\dot{w}_x(t) \quad \dot{w}_y(t) \quad \dot{w}_z(t)]$ representa el vector de ruido blanco de proceso, considerando que en las coordenadas x, y es ruido de proceso de aceleración y en la coordenada z ruido de velocidad. Se puede observar que este modelo dinámico considera al vector de entrada al modelo $u(t) = 0$.

Asimismo, el modelo de espacio de estados discreto puede ser representado de la siguiente forma:

$$x_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \frac{T^2}{2} & 0 & 0 \\ T & 0 & 0 \\ 0 & \frac{T^2}{2} & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{bmatrix} w_k \quad (2.4)$$

Donde T representa el tiempo de muestreo, x_k el vector de estados discreto y w_k el vector secuencial de ruido de proceso. Cabe resaltar que el modelo (2.4) es deducido directamente de una evolución discreta de los estados y no de discretizar el modelo dinámico continúo planteado en (2.3).

El modelo anteriormente planteado es ideal para representar la dinámica del blanco aéreo cuando se traslada en una trayectoria recta a velocidad constante de un punto "A" a un punto "B", siendo este el caso de los aviones comerciales. Sin embargo, este tipo de modelo por sí solo no posee la capacidad de representar la dinámica compleja del blanco aéreo de alta maniobrabilidad.

2.2.2 Modelos de maniobra con coordenadas desacopladas

Las coordenadas tridimensionales de un blanco aéreo se acoplan durante la ejecución de cambios de rumbo, maniobras evasivas y terminales, sin embargo, muchos modelos matemáticos no consideran este acoplamiento para simplificar el modelamiento. Asimismo, este tipo de modelos con coordenadas desacopladas modelan al vector de entrada $u(t)$ como un proceso aleatorio dado que el mismo no es medible.

Los modelos de maniobra con coordenadas desacopladas pueden ser clasificados, según lo descrito en [34], de la siguiente manera:

- **Modelos de ruido blanco:** Este tipo de modelos consideran al vector de entrada de control $u(t)$ como un proceso de ruido blanco, siendo los más comunes los modelos a velocidad constante (CV), aceleración constante (CA) y los polinomiales.
- **Modelos de proceso de Markov:** Este tipo de modelos consideran al vector de entrada de control $u(t)$ como un proceso de Markov[5], siendo el modelo más representativo el modelo de Singer descrito en [34], el cual asume que la aceleración del blanco $a(t)$ es un proceso de markov de primer orden estacionario con media cero continuo en el tiempo:

$$\dot{a}(t) = -\alpha a(t) + w(t) \quad (2.5)$$

[5] El proceso de Markov o cadena de Markov es un tipo de proceso aleatorio en el cual la probabilidad de que ocurra un evento depende de la propiedad de Markov. Esto es, la probabilidad de que ocurra el evento depende exclusivamente del evento inmediatamente anterior.

Donde $w(t)$ es ruido blanco de proceso con media cero con densidad espectral de potencia $S_w = 2\alpha\sigma^2, a(t)$ es la aceleración del blanco con función de autocorrelación $R_a(\tau) = E[a(t+\tau)a(t)] = \sigma^2 e^{-\alpha|\tau|}$ y espectro de potencia $S_a(w) = \frac{2\alpha\sigma^2}{w^2+\alpha^2}$, α es la recíproca de la constante de tiempo de maniobra τ_m, la cual tiene un valor de entre 10 a 20 segundos para una maniobra evasiva según lo descrito en Singer [35] y σ^2 es la varianza instantánea de la aceleración.

Su modelo de espacio de estados viene estructurado de la siguiente forma:

$$\dot{x(t)} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha \end{bmatrix} x(t) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} w(t) \quad (2.6)$$

Donde $x(t) = [x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}, z, \dot{z}, \ddot{z}]'$ es el vector de estados y $w(t) = [w_x(t), w_y(t), w_z(t)]'$ es el vector de ruido blanco de proceso. Asimismo, el modelo equivalente en tiempo discreto:

$$\dot{a}_{k+1} = -\beta a_k + w_k \quad (2.7)$$

Donde w_k es una secuencia de ruido blanco con varianza $\sigma^2(1-\beta^2)$, β está relacionado al modelo continuo mediante $\beta = e^{-\alpha T}$ y T es el tiempo de muestreo y su modelo dinámico de espacio de estados es igual a:

$$x_{k+1} = diag[F_k, F_k, F_k]x_k + w_k \quad (2.8)$$

donde,

$$F_k = \begin{bmatrix} 1 & T & \frac{(\alpha T - 1 + e^{-\alpha T})}{\alpha^2} \\ 0 & 1 & \frac{(1 - e^{-\alpha T})}{\alpha} \\ 0 & 0 & e^{-\alpha T} \end{bmatrix} \quad (2.9)$$

Como se puede apreciar, la exactitud del modelo de Singer depende de la correcta elección de los valores α y σ^2, los cuales son variantes en el tiempo y dependientes de la dinámica del blanco, por lo que métodos adaptativos o redes neuronales pueden ser utilizados para estimar estos parámetros de forma online. Sin embargo, al presentar la aceleración de un blanco aéreo por medio de un proceso de markov con media cero genera divergencias, por lo que el modelo de Singer con media adaptativa es propuesto por Zhou y Kumar [36] para mejorar la performance del modelo estándar de Singer.

El modelo de Singer con media adaptativa propuesto por Zhou y Kumar [35] plantea la siguiente ecuación para la aceleración $a(t)$ del blanco:

$$a(t) = \tilde{a}(t) + \bar{a}(t) \quad (2.10a)$$
$$\dot{a}(t) = -\alpha a(t) + \alpha\bar{a}(t) + w(t) \quad (2.11a)$$

Donde $\tilde{a}(t)$ representa la aceleración del modelo de Singer y $\bar{a}(t)$ es la media de la aceleración de Singer a calcular adaptativamente. El modelo dinámico continuo de Singer con media adaptativa en espacio de estados está representado de la siguiente forma:

$$\dot{x}(\mathrm{t}) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha \end{bmatrix} x(t) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \alpha & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \alpha \end{bmatrix} \bar{a}(t) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} w(t)$$

Y el modelo equivalente en tiempo discreto:

$$x_{k+1} = diag[F_k, F_k, F_k]x_k + \ \mathrm{diag}[G_k, G_k, G_k]\bar{a}(t) + \ w_k \qquad (2.12)$$

donde F_k es la matriz de transición considerada en (2.9) y G_k está dada de la siguiente manera:

$$G_k = \begin{bmatrix} \frac{T^2}{2} - \frac{\alpha T - 1 + e^{-\alpha T}}{\alpha^2} \\ T - \frac{1 - e^{-\alpha T}}{\alpha} \\ 1 - e^{-\alpha T} \end{bmatrix} \qquad (2.13)$$

2.2.3 Modelos de proceso de salto de Semi-Markov

La dificultad de los modelos dinámicos planteado como proceso de markov radica en que no se conoce en que intervalos de tiempo la aceleración es constante con media diferente de cero, por lo que los modelos de proceso de salto de Semi-Markov asumen que la aceleración o entrada de control $u(t)$ salta de un modo a otro en tiempos aleatorios, a diferencia del proceso de salto de Markov que cuenta con saltos a tiempos determinísticos, según lo expuesto por Zhou y Kumar [36].

2.2.4 Modelos de sobre aceleración (Jerk models)

Los modelos dinámicos comúnmente consideran al vector de entrada de control $u(t)$ como la aceleración, dado que esta es una consecuencia directa de la fuerza que genera el movimiento del blanco en el espacio. Sin embargo, Merohtra y Mahapatra [37] exponen que, al incluir la derivada de la aceleración en el modelamiento, esta es la sobre aceleración o *jerk,* se pueden obtener mejores resultados en el traqueo o seguimiento de blancos aéreos de alta maniobrabilidad, dado que la sobre aceleración es una magnitud física que representa de mejor manera a la fuerza que causa el movimiento.

El modelo dinámico discreto de sobre aceleración planteado planteado por Merohtra y Mahapatra [37] es el siguiente:

$$x_{k+1} = diag[F_k, F_k, F_k]x_k + \ w_k \qquad (2.14)$$

Donde $x(t) = [x, \dot{x}, \ddot{x}, \dddot{x}, y, \dot{y}, \ddot{y}, \dddot{y}, z, \dot{z}, \ddot{z}, \dddot{z}]'$ es el vector de estados, w_k es la secuencia de ruido de proceso y F_k viene dado de la siguiente manera:

$$F_k = \begin{bmatrix} 1 & T & \frac{T^2}{2} & p_1 \\ 0 & 1 & T & q_1 \\ 0 & 0 & 1 & r_1 \\ 0 & 0 & 0 & s_1 \end{bmatrix} \qquad (2.15)$$

Donde $p_1 = \frac{T^3}{3!} - \frac{\alpha T^4}{4!} + \frac{\alpha^2 T^5}{5!} - \cdots = \frac{2 - 2\alpha T + \alpha^2 T^2 - 2e^{-\alpha T}}{2\alpha^3}$, $q_1 = \frac{T^2}{2!} - \frac{\alpha T^3}{3!} + \frac{\alpha^2 T^4}{4!} - \cdots = \frac{e^{-\alpha T} - 1 + \alpha T}{\alpha^2}$, $r_1 = T - \frac{\alpha T^2}{2!} + \frac{\alpha^2 T^3}{3!} - \frac{\alpha^3 T^4}{4!} + \cdots = \frac{1 - e^{-\alpha T}}{\alpha}$, $s_1 = 1 - \alpha T + \frac{\alpha^2 T^2}{2!} - \frac{\alpha^3 T^3}{3!} + \frac{\alpha^4 T^4}{4!} - \cdots = e^{-\alpha T}$

Se observa que los términos $p_1, q_1, r_1\ y\ s_1$ son series de Taylor, siendo la constante α la correlación en el tiempo con la que cuenta la sobre aceleración. Esto es, si el blanco cuenta con una sobre aceleración determinada en el instante t, la correlación en el tiempo determina si es que la sobre aceleración será la misma en el instante $t + \tau$. Las maniobras lentas poseen alta correlación de la sobre aceleración en el tiempo, mientras que las maniobras rápidas evasivas y terminales muestra baja correlación.

Si a partir del concepto anterior se asume que el producto αT es lo suficientemente pequeño, la matriz de transición F_k puede ser representada de la siguiente manera:

$$F_k = \begin{bmatrix} 1 & T & \frac{T^2}{2} & \frac{T^3}{6} \\ 0 & 1 & T & \frac{T^2}{2} \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

Otro modelo de sobre aceleración, utilizado para la implementación de filtros de traqueo de tipo Kalman, es el modelo planteado por Ramachandra [38]. Este modelo es un modelo de tres estados discreto unidimensional de un blanco aéreo a aceleración constante, en el cual se considera a la entrada de control u_k como un proceso aleatorio, representado por la sobre aceleración o *jerk* del blanco aéreo. Este modelo, denominado "modelo número II de Ramachandra", está representado por el siguiente espacio de estados:

$$x_{k+1} = diag[F_k, F_k, F_k]x_k + \ \text{diag}[G_k, G_k, G_k]w_k \tag{2.17}$$

donde $F_k = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}$ es la matriz de transición de estados, $G_k = \begin{bmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix}$ es la matriz de entrada de control y w_k es la secuencia de ruido blanco que representa la sobre aceleración del blanco.

Se observa que el modelo planteado en (2.17) es la versión no aumentada del modelo (2.14), dado que al unir F_k y G_k y agregar una fila [0 0 0 1] en una sola matriz resulta en la matriz de transición (2.16). Por lo cual, el modelo número II de ramachandra es en esencia el modelo planteado en [31] Kishore Merohtra y Pravas Mahapatra pero con cero correlación en el tiempo de la sobre aceleración dado que el parámetro α de las series de Taylor del modelo 2.14 es igual a cero.

2.2.5 Modelos de movimiento horizontal con acoplamiento en dos dimensiones

Este tipo de modelos se basan en la cinemática del blanco en un plano horizontal, a diferencia de los modelos descritos anteriormente que cuentan con una señal de entrada de control $u(t)$ representada por un proceso aleatorio. Por ello, este tipo de modelos describen de mejor manera el acoplamiento espacial que existe entre las coordenadas cartesianas del sistema. Asimismo, este tipo de modelos son utilizados para el traqueo de aeronaves comerciales, dado que sus maniobras de cambio de rumbo son ejecutadas en un plano horizontal a un ángulo de timón constante.

Uno de los modelos más conocidos en la literatura es el de giro constante (CT turn model), tanto en coordenadas cartesianas como en polares, respecto a un centro de giro ficticio en el espacio. El modelo en tres dimensiones descrito en [34] tiene la siguiente forma:

$$\dot{a}(t) = \ -w^2 v(t) + w(t) \tag{2.18}$$

Donde $v(t)$ es el vector de velocidad, $w(t)$ es el vector de ruido blanco de proceso y w es la velocidad de giro con que se ejecuta la maniobra (°/seg). El modelo dinámico continuo de espacio de estados para cada coordenada cartesiana con vector de estado $x = \ [x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}, z, \dot{z}, \ddot{z}]'$está dado de la siguiente forma:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -w^2 & 0 \end{bmatrix} x(t) + w(t) \qquad (2.19)$$

Y su modelo dinámico discreto:

$$x_{k+1} = \begin{bmatrix} 1 & \frac{sinwT}{w} & \frac{1-coswT}{w^2} \\ 0 & coswT & \frac{sinwT}{w} \\ 0 & -wsinwT & coswT \end{bmatrix} x_k + \begin{bmatrix} \frac{wT-sinwT}{w^3} \\ \frac{1-coswT}{w^2} \\ \frac{sinwT}{w} \end{bmatrix} w_k \qquad (2.20)$$

Cabe resaltar que las coordenadas cartesianas de este modelo se encuentran acopladas por medio de la velocidad de giro del blanco, la cual es un parámetro de entrada desconocido que debe ser aproximado adaptativamente durante el traqueo.

2.2.6 Modelos de movimiento con acoplamiento en tres dimensiones

Como se ha mencionado anteriormente los modelos dinámicos para aeronaves comerciales utilizan en su mayoría modelos de movimiento acoplado en dos dimensiones, debido a que los giros se efectúan en el plano horizontal x-y. Asimismo, en adición al modelo dinámico planteado en el plano x-y se utiliza normalmente un modelo a velocidad constante (CV) en el eje z, dado que normalmente los cambios de altitud no son efectuados simultáneamente con los giros. Sin embargo, este método no es el adecuado cuando se intenta traquear aeronaves de combate o misiles antibuque debido al acoplamiento que existe entre las tres coordenadas para este tipo de blancos aéreos. Xiaohua Nie y Fuming Zhang [39] proponen un modelo cinemático en 3D de giro variable (3DVT), el cual es utilizado para traqueo adaptativo de aeronaves militares con capacidad de maniobra a altas gravedades o *high g*, de acuerdo al siguiente detalle:

Sea el modelo continuo en espacio de estados,

$$\dot{x}(t) = diag[A, A, A]x(t) + diag[B, B, B]w(t) \qquad (2.21)$$

Donde $x = [x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}, z, \dot{z}, \ddot{z}]'$ es el vector de estado, $w(t)$ es el vector de ruido de proceso, $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2\alpha^2 - w^2 & -2\alpha \end{bmatrix}$ es la matriz de transición en la coordenada x, y o z, $B = [0 \quad 0 \quad 1]'$ es la matriz de entrada de control en la coordenada x, y o z, α es el coeficiente de amortiguamiento de la maniobra y w es la velocidad de giro de la maniobra.

Su modelo discreto en espacio de datos viene representado de la siguiente forma:

$$x_{k+1} = diag[F_k, F_k, F_k]x_k + w_k \qquad (2.22)$$

Donde $F_k = \begin{bmatrix} 1 & f_{12} & f_{13} \\ 0 & f_{22} & f_{23} \\ 0 & f_{32} & f_{33} \end{bmatrix}$, es la matriz de transición del modelo dinámico del blanco, y sus elementos son $f_{12} = \frac{2\alpha w - e^{-\alpha T}(2\alpha w coswT + (\alpha^2 - w^2)sinwT)}{w(\alpha^2 + w^2)}$, $f_{13} = \frac{w - e^{-\alpha T}(wcoswT + \alpha sinwT)}{w(\alpha^2 + w^2)}$, $f_{22} = \frac{e^{-\alpha T}(wcoswT + \alpha sinwT)}{w}$, $f_{23} = \frac{e^{-\alpha T}sinwT}{w}$, $f_{32} = -\frac{e^{-\alpha T}(\alpha^2 + w^2)sinwT}{w}$, $f_{33} = \frac{e^{-\alpha T}(wcoswT - \alpha sinwT)}{w}$

2.3 Modelo propuesto para el blanco aéreo de alta maniobrabilidad

A partir del estudio de los modelos dinámicos propuestos en la literatura del apartado anterior, se propone utilizar un ***modelo lineal incierto*** de tres dimensiones en espacio de estados:

$$x_{k+1} = F_k x_k + G_k u_k + D_k \xi_k + w_k \quad (2.23a)$$
$$y_k = H_k + v_k \quad (2.23b)$$

donde (2.23a) es el modelo dinámico que representa el movimiento cinemático del blanco en tres dimensiones y (2.23b) el modelo de medición que representa las mediciones efectuadas por el radar del sistema de seguimiento automático. Asimismo, el vector de estados en coordenadas cartesianas es $x_k = [x, \dot{x}, \ddot{x}, \mathrm{y}, \dot{y}, \ddot{y}, \mathrm{z}, \dot{z}, \ddot{z}]'$, de dimensión $9\ x\ 1$, u_k es el vector de entrada de sobre aceleraciones, de dimensión $3\ x\ 1$, ξ_k es el vector de incertidumbres acotado[6], de dimensión $3\ x\ 1$, y_k es el vector de mediciones, v_k es el ruido de medición y w_k es el ruido de proceso. La matriz de transición, de dimensión $9\ x\ 9$, viene representada de la siguiente forma:

$$F_k = \begin{bmatrix} 1 & T & \frac{T^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & \frac{T^2}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & T & \frac{T^2}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.24)$$

Y la matriz de entrada de control, de dimensión $9\ x\ 3$:

$$G_k = \begin{bmatrix} \mathrm{p}_1 & 0 & 0 \\ \mathrm{q}_1 & 0 & 0 \\ r_1 & 0 & 0 \\ 0 & \mathrm{p}_1 & 0 \\ 0 & \mathrm{q}_1 & 0 \\ 0 & r_1 & 0 \\ 0 & 0 & \mathrm{p}_1 \\ 0 & 0 & \mathrm{q}_1 \\ 0 & 0 & r_1 \end{bmatrix}^{7} \quad (2.25)$$

Donde $p_1 = \frac{T^3}{3!} - \frac{\alpha T^3}{4!} + \frac{\alpha^2 T^5}{5!} - \cdots = \frac{2-2\alpha T+\alpha^2 T^2-2e^{-\alpha T}}{2\alpha^3}$, $q_1 = \frac{T^2}{2!} - \frac{\alpha T^3}{3!} + \frac{\alpha^2 T^4}{4!} - \cdots = \frac{e^{-\alpha T}-1+\alpha T}{\alpha^2}$, $r_1 = T - \frac{\alpha T^2}{2!} + \frac{\alpha^2 T^3}{3!} - \frac{\alpha^3 T^4}{4!} + \cdots = \frac{1-e^{-\alpha T}}{\alpha}$ y α la correlación en el tiempo con la que cuenta la sobre aceleración. Por otro lado, D_k es la matriz de entrada de incertidumbres, de dimensión $9\ x\ 3$, la cual posee los coeficientes $\rho_{px} = 1, \rho_{vx} = 1, \rho_{ax} = 2, \rho_{py} = 1, \rho_{vy} = 1, \rho_{ay} = 2, \rho_{pz} = 1, \rho_{vz} = 1\ y\ \rho_{az} = 2$. Estos coeficientes modelan el efecto de las incertidumbres estructuradas o no linealidades del modelo sobre la posición, velocidad y aceleración del blanco durante las maniobras de cambio de rumbo y maniobra terminal del misil, permitiendo obtener movimientos altamente ágiles del blanco.

[6] El vector de perturbaciones ξ_k representa las incertidumbres no lineales no contempladas en el modelo lineal. Estas incertidumbres aparecen cuando el blanco efectúa maniobras evasivas y maniobras terminales. Asimismo, estas incertidumbres deberán ser funciones senoidales y cosenoidales, encontrándose acopladas en las tres coordenadas por medio de una velocidad de giro w determinada, permitiendo que el modelo lineal incierto, que aparentemente podría ser considerado un modelo desacoplado, cuente con acoplamiento de coordenadas durante las maniobras del blanco.

[7] La matriz G_k puede simplificarse de acuerdo a lo expuesto en (2.17) si se considera un período de muestreo T y una constante α lo suficientemente pequeños.

$$D_k = \begin{bmatrix} \rho_{px} & 0 & 0 \\ \rho_{vx} & 0 & 0 \\ \rho_{ax} & 0 & 0 \\ 0 & \rho_{py} & 0 \\ 0 & \rho_{vy} & 0 \\ 0 & \rho_{ay} & 0 \\ 0 & 0 & \rho_{pz} \\ 0 & 0 & \rho_{vz} \\ 0 & 0 & \rho_{az} \end{bmatrix} \quad (2.26)$$

H_k es la matriz de observación no lineal, de dimensión $3\ x\ 1$:

$$H_k = \begin{bmatrix} Ad\cos(Ed)\cos(Bdn) \\ Ad\cos(Ed)\sin(Bdn) \\ Adsin(Ed) \end{bmatrix} \quad (2.27)$$

donde Ad, Ed y Bdn son la distancia, elevación y acimut del blanco, respectivamente. Cabe resaltar que, si previamente se ha efectuado la conversión de coordenadas polares a cartesianas, la ecuación 2.23b queda representada como $y_k = H_k x_k + v_k$, siendo la matriz H_k representada en coordenadas cartesianas de la siguiente manera:

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (2.28)$$

La matriz anterior, de dimensión $3\ x\ 6$, representa un radar con capacidad de medir la variable de estado de posición en cada coordenada cartesiana, sin embargo, si se cuenta con un radar a pulsos doppler, que en adición a la posición mida la velocidad del blanco, se puede considerar la siguiente matriz de observación H_k:

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.29)$$

El modelo lineal incierto anteriormente planteado permite la implementación de observadores basados en sistemas de estructura variable, sin embargo, ***el vector de sobre aceleraciones u_k debe ser estimado por un método apropiado***. Finalmente, en la figura 2.1 se muestra el modelo lineal incierto discreto de espacio de estados en diagrama de bloques:

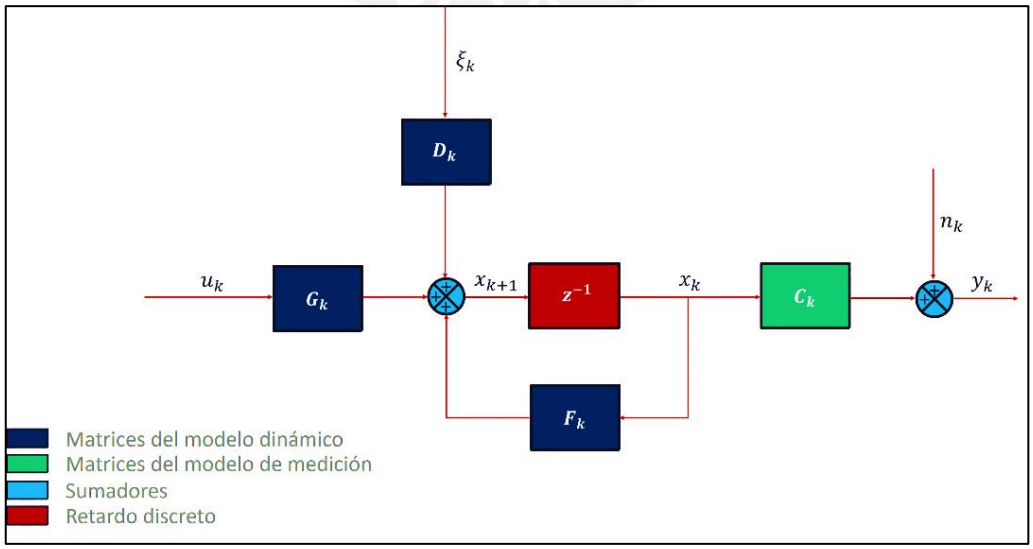

Figura 2.1.: Diagrama de bloques del modelo lineal incierto discreto de espacio de estados.
Fuente: Elaboración propia.

2.4 Simulación del modelo propuesto

En este apartado se efectuó la simulación del modelo lineal incierto del blanco en el software MATLAB®. Para tal efecto, fue necesario generar una entrada de control u_k, un vector de incertidumbres estructuradas ξ_k y un modelo de ruido de medición apropiado para el modelo lineal incierto.

2.4.1 Generación de vector de entrada de control u_k

La sobre aceleración o jerk está definida como la derivada de la aceleración, siendo sus unidades representadas en m/s3 o g/s, donde g es 1 gravedad. Al verificar gráficamente la relación entre la posición, velocidad, aceleración y sobre aceleración uno puede apreciar que esta última puede ser aproximada como un pulso angosto. Por tanto, en el software MATLAB® se utilizó la función "gmonopuls" para generar los impulsos de sobre aceleración que excitan al modelo lineal incierto de forma adecuada y permiten obtener trayectorias reales del blanco. Para tal efecto, fue necesario establecer los límites de sobre aceleración, a fin de que estos valores al excitar el modelo del sistema planteado permitan generar posiciones, velocidades y aceleraciones realistas y dentro de parámetros aceptables.

Para el caso de las sobre aceleraciones generadas en una aeronave de combate, en [40] se explica que el piloto de una aeronave se encuentra sometido a aceleraciones lineales en los ejes transversales, laterales y verticales durante el vuelo, siendo determinados por medio de estudios científicos los límites de tolerancia del ser humano a estas aceleraciones. Por tanto, en la figura 2.2 se muestran las curvas de aceleración versus sobre aceleración, las cuales denotan los límites máximos de aceleración (en unidades de "g") y sobre aceleración (en unidades de "g/s") y que servirán para diseñar datos de entrada apropiados al modelo planteado del sistema.

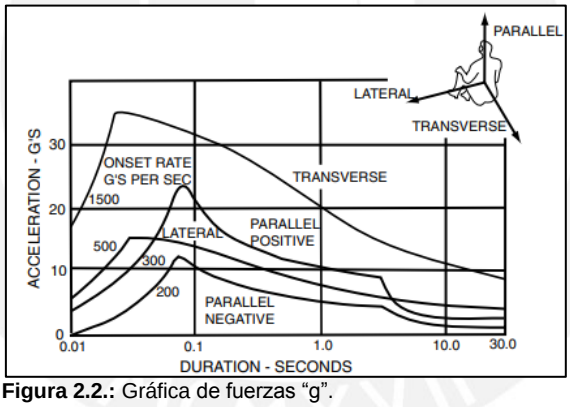

Figura 2.2.: Gráfica de fuerzas "g".
Fuente: Brulle [39].

En la gráfica se puede observar que, por ejemplo, el piloto de una aeronave puede resistir como máximo una aceleración paralela negativa de 10g por un tiempo de 0.1 segundos en la curva de sobre aceleración de 200 g/s. Por tanto, se extrajeron los datos numéricos de las curvas de la gráfica con la finalidad de tener valores límite para el vector de entrada de control u_k.

Sin embargo, Siouris [41] establece que la máxima aceleración soportada por un misil antibuque se encuentra alrededor de los 45 g's. Asimismo, Shkolnikov et al. [42] establece como condición limitante en un modelo del sistema de guiado de un misil a base de modos deslizantes que la aceleración máxima sea de 30 g's y la sobre aceleración de 100 g/s. Por ello, para las simulaciones del vector de entrada de control u_k se tomó como valores máximos de aceleración 45 g's y sobre aceleración 100 g/s, brindado la máxima maniobrabilidad con la que podría contar el blanco aéreo, que para la presente tesis será modelado como un misil antibuque.

El algoritmo 2.1 describe la función implementada en el programa de MATLAB® del apéndice 1.1, la cual permite generar un vector de entrada de control tridimensional apropiado, a fin de generar una trayectoria realista del misil hacia un objetivo determinado.

Algoritmo 2.1: Generación de vector de entrada de control u_k

Entradas: Gravedad $\boldsymbol{g}$, factor de saturación de sobre aceleración $\boldsymbol{load\_factor}$, tiempo de muestreo $\boldsymbol{T}$ y número de muestras $\boldsymbol{nt}$.
Salidas: Vector de entrada de control $\boldsymbol{u_k}$
Inicio del programa:
1: Declara la frecuencia central $\boldsymbol{f_c}$ y ratio de muestreo $\boldsymbol{f_s}$ del monopulso gaussiano
2: Halla el tiempo de corte $\boldsymbol{t_c}$ del monopulso gaussiano
3: Halla $\sigma = \frac{1}{2\pi f_c}$
4: Establece los vectores de tiempo ***t(i) = 0*T:1/f_s:n(i)*T*t_c /0.6366***
5: Establece los monopulsos gaussianos para cada vector de tiempo *t(i),* donde $x(i) = Ae^{1/2}\frac{t(i)}{\sigma}e^{-(\frac{t(i)}{\sigma})^2/2}$ es el monopulso gaussiano y ***A*** es la amplitud del pulso en gravedades "g".
6: Concatena los vectores de los monopulsos $x(0), x(1), x(2),..$ en el vector $\boldsymbol{x_x}$
7: Efectuar el mismo procedimiento para hallar los vectores de sobre aceleraciones en la coordenada "y" y "z" $\boldsymbol{x_y}$ y $\boldsymbol{x_z}$.
8: Convierte $\boldsymbol{x_x}$, $\boldsymbol{x_y}$ y $\boldsymbol{x_z}$ a m/s3
9: Concatena $u_k = [\boldsymbol{x_x}\ \boldsymbol{x_y}\ \boldsymbol{x_z}]'$
10: Para k = 1:nt **Hacer**
11: **Para** i = 1:3
12: **Si** $\boldsymbol{u_k}(\boldsymbol{i},\boldsymbol{k})$ **mayor que** $load\_factor$
13: $\boldsymbol{u_k}(\boldsymbol{i},\boldsymbol{k}) = load\_factor$
14: **Fin Si**
15: **Si** $\boldsymbol{u_k}(\boldsymbol{i},\boldsymbol{k})$ **menor que** -$load\_factor$
16: $\boldsymbol{u_k}(\boldsymbol{i},\boldsymbol{k}) = -load\_factor$
17: **Fin Si**
18: **Fin Para**
19: Fin Para
20: Fin del programa

Al ejecutar el programa 2.1 se generaron las gráficas de las salidas $\boldsymbol{u_k}$ en coordenadas cartesianas, obteniéndose una sobre aceleración máxima del misil de 60 g/s.

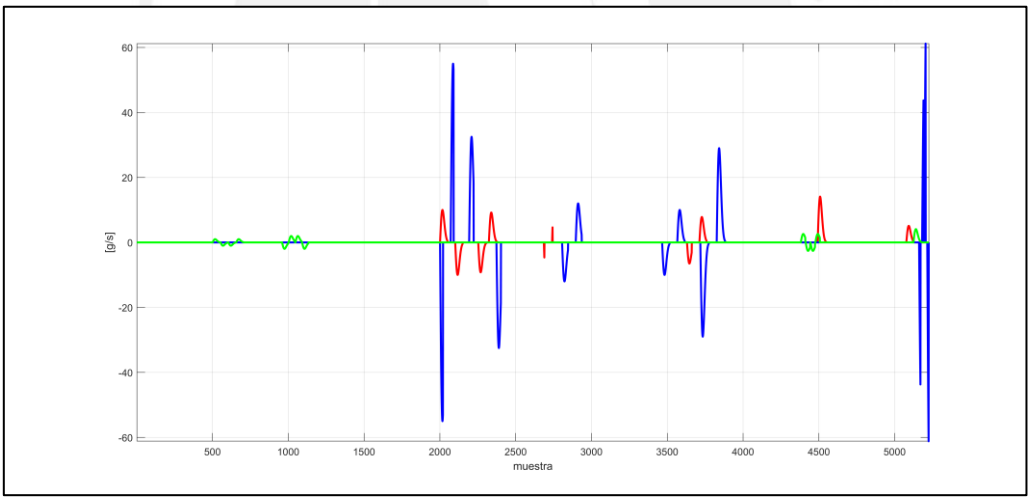

Figura 2.3.: Entrada de control al modelo en coordenadas cartesianas. Componente coordenada "x" (Línea roja). Componente coordenada "y" (línea azul). Componente coordenada "z" (Línea verde).
Fuente: Elaboración propia.

2.4.2 Generación de incertidumbres estructuradas ξ_k

Dado que en el modelamiento no se contemplaron las no linealidades que posee un blanco aéreo de alta maniobrabilidad, el vector de incertidumbres estructuradas ξ_k permite perturbar al modelo lineal de tal forma que este, para maniobras bruscas de cambio de rumbo y maniobras evasivas o terminales, posea una dinámica no lineal. Para tal efecto, se utilizaron funciones senoidales y cosenoidales, en unidades de m/s3, para modelar los cambios de rumbo no lineales y el patrón helicoidal ejecutado por el misil durante su trayectoria terminal al objetivo.

El algoritmo 2.2 describe la función implementada en el programa de MATLAB® del apéndice 1.2, la cual permite generar un vector de entrada de control tridimensional apropiado, a fin de generar una trayectoria realista del misil hacia un objetivo determinado.

Algoritmo 2.2: Generación de vector de perturbaciones ξ_k

Entradas: Gravedad $\boldsymbol{g}$, tiempo inicial de simulación $\boldsymbol{t_i}$, tiempo de muestreo $\boldsymbol{T}$ y tiempo final de simulación $\boldsymbol{t_{ff}}$

Salidas: Vector de perturbaciones $\boldsymbol{\xi_k}$

Inicio del programa:

1: Declara la frecuencia base de perturbaciones $\boldsymbol{f_b}$ en Hz.
2: Declara amplitudes de perturbaciones $\boldsymbol{a}$, $\boldsymbol{b}$ y $\boldsymbol{c}$ en m/s3
3: Declara el número de muestra $\boldsymbol{k} = \mathbf{1}$
4: Para t_t= t_i:T: t_{ff} **Hacer**
5: **Si** k **mayor o igual que** 1160 **y menor que** 1260
6: $\boldsymbol{\xi_{k_x}} = -0.0135b$
7: $\boldsymbol{\xi_{k_y}} = 0$
8: $\boldsymbol{\xi_{k_z}} = 0$
9: **Fin Si**
10: **Si** k **mayor o igual que** 1260 **y menor que** 1477
11: $\boldsymbol{\xi_{k_x}} = 0.25bsen(100\pi fkT/(2c))$
12: $\boldsymbol{\xi_{k_y}} = 0$
13: $\boldsymbol{\xi_{k_z}} = 0$
14: **Fin Si**
15: **Si** k **mayor o igual que** 3900 **y menor que** 4000
16: $\boldsymbol{\xi_{k_x}} = -0.012b$
17: $\boldsymbol{\xi_{k_y}} = 0$
18: $\boldsymbol{\xi_{k_z}} = 0$
19: **Fin Si**
20: **Si** k **mayor o igual que** 4000 **y menor que** 4354
21: $\boldsymbol{\xi_{k_x}} = 0.3bsen(200\pi f_b kT/(2c))$
22: $\boldsymbol{\xi_{k_y}} = 0$
23: $\boldsymbol{\xi_{k_z}} = 0$
24: **Fin Si**
25: **Si** k **mayor o igual que** 4650 **y menor que** 5167
26: $\boldsymbol{\xi_{k_x}} = bsen(2\pi f_b t/(2c))cos(2\pi f_b t)$
27: $\boldsymbol{\xi_{k_y}} = cos(32\pi f_b t/(2c))$
28: $\boldsymbol{\xi_{k_z}} = asen(2\pi ft/(2c))sen(2\pi ft)$
29: **Fin Si**
30: Guardar vector de perturbaciones $\boldsymbol{\xi_k(k)} = [\xi_{k_x}\ \xi_{k_y}\ \xi_{k_z}]'$
31: Fin Para
32: Fin Programa

Al ejecutar el programa 2.2 se generaron las gráficas de las salidas $\boldsymbol{\xi_k}$ en coordenadas cartesianas, obteniéndose una perturbación máxima de 3 g/s. En la siguiente figura se pueden observar las perturbaciones:

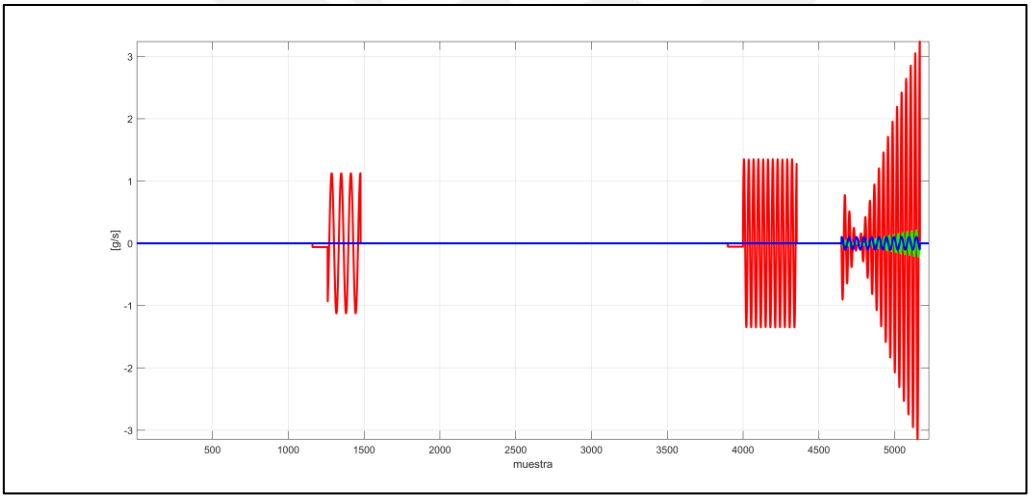

Figura 2.4.: Perturbación al modelo en coordenadas cartesianas. Componente coordenada "x" (Línea roja). Componente coordenada "y" (línea azul). Componente coordenada "z" (Línea verde).
Fuente: Elaboración propia.

2.4.3 Generación del modelo ruido

De acuerdo a lo explicado en el apartado 1.3 del capítulo I, el ruido que posee mayor influencia sobre el traqueo de un blanco aéreo que trayectoria sea skimming o rasante sobre el nivel del mar es el ruido angular. Para tal efecto, según lo descrito en Kim et al. [31] y Daeipour y Bar-Shalom [32], se diseñó un modelo de distribución gaussiana mixta con la siguiente estructura:

- **Modelo mixto gaussiano de distancia, acimut y elevación Caso 1:** Una distribución gaussiana mixta de dos distribuciones y tres variables (distancia, acimut y elevación) cuando el blanco efectúe vuelos por encima de los cinco metros, de acuerdo al siguiente detalle:
  - Las medias en distancia, acimut y elevación correspondientes a la distribución gaussiana nro. 1 son $\mu_{Ad} = 0.1m$, $\mu_{Bdn} = 0.01mrad$ y $\mu_{Ed} = 0.01mrad$, respectivamente, y medias en distancia, acimut y elevación correspondientes a la distribución gaussiana nro. 2 son $\mu_{Ad} = 0.05\,m$, $\mu_{Bdn} = 0.02\,mrad$ y $\mu_{Ed} = 0.03\,mrad$, respectivamente.
  - Los elementos de las diagonales de las matrices de covarianza del ruido de la distribución gaussiana nro. 1 y 2 aumentan progresivamente mediante un algoritmo que multiplica al elemento de la diagonal por el número de muestra en que se encuentra la trayectoria. Esto permite que se generen 100 segmentos de 52 muestras y 1 segmento de 26 muestras, cada segmento con una distribución gaussiana mixta, y que los 5226 datos de la distribución de ruido generada tengan una amplitud ascendente a medida que el misil se acerca al objetivo; este aumento de amplitud es característico en el ruido angular. Las matrices de covarianza mixta tienen unidades de varianza en distancia, acimut y elevación de m^2, $mrad^2$ y $mrad^2$, respectivamente, siendo sus valores máximos los siguientes:

$$P_1 = \begin{bmatrix} 2.02 & 0.00 & 0.00 \\ 0.00 & 0.5 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix} \quad P_2 = \begin{bmatrix} 29.997 & 0.00 & 0.00 \\ 0.00 & 3.535 & 0.00 \\ 0.00 & 0.00 & 4.545 \end{bmatrix}$$

  - La proporción de mezcla es de 0.95 para la distribución número uno, y 0.05 para la distribución número dos.

- **Modelo mixto gaussiano de distancia, acimut y elevación Caso 2:** Una distribución gaussiana mixta de dos distribuciones y tres variables (distancia, acimut y elevación) cuando el blanco efectúe vuelos rasantes a la superficie del mar o trayectoria "sea skimming", por debajo de los cinco metros. Esta distribución gaussiana mixta posee picos de ruido de mayor amplitud, de acuerdo al siguiente detalle:
  - Las medias en distancia, acimut y elevación correspondientes a la distribución gaussiana nro. 1 son $\mu_{Ad} = 0.2\,m$, $\mu_{Bdn} = 0.02\,mrad$ y $\mu_{Ed} = 0.02\,mrad$, respectivamente, y medias en distancia, acimut y elevación correspondientes a la distribución gaussiana nro. 2 son $\mu_{Ad} = 0.1\,m$, $\mu_{Bdn} = 0.04\,mrad$ y $\mu_{Ed} = 0.06\,mrad$, respectivamente.
  - Las matrices de covarianza del ruido de las distribuciones gaussianas nro. 1 y 2, en unidades de varianza de distancia, acimut y elevación de m^2, $mrad^2$ y $mrad^2$, poseen una norma mayor que las matrices del caso número uno o de vuelo por encima de los cinco metros. Al respecto, sus covarianzas finales son las siguientes:

$$P_1 = \begin{bmatrix} 5.05 & 0.00 & 0.00 \\ 0.00 & 1.01 & 0.00 \\ 0.00 & 0.00 & 2.02 \end{bmatrix} \quad P_2 = \begin{bmatrix} 70.094 & 0.00 & 0.00 \\ 0.00 & 4.44 & 0.00 \\ 0.00 & 0.00 & 4.95 \end{bmatrix}$$

  - La proporción de mezcla es de 0.95 para la distribución número uno, y 0.05 para la distribución número dos.

- **Modelo mixto gaussiano de velocidad Caso 1:** Una distribución gaussiana mixta de dos distribuciones y una variable (velocidad) cuando el blanco efectúe vuelos por encima de los cinco metros, de acuerdo al siguiente detalle:
  - Las medias en velocidad correspondientes a las distribuciones gaussianas nro. 1 y 2 son $\mu_{Vm} = 0.02\,m/s$ y $\mu_{Vm} = 0.1\,m/s$, respectivamente.

- Las varianzas del ruido de las distribuciones gaussianas nro. 1 y 2 aumentan progresivamente, con unidades de $\frac{m^2}{s^2}$, siendo sus valores máximos iguales a $\sigma^2 = 1.01\ \frac{m^2}{s^2}$y $3.53\ \frac{m^2}{s^2}$
- La proporción de mezcla es de 0.95 y 0.05.

- **Modelo mixto gaussiano de velocidad Caso 2:** Una distribución gaussiana mixta de dos distribuciones y una variable (velocidad) cuando el blanco efectúe vuelos rasantes a la superficie del mar o trayectoria "sea skimming", por debajo de los cinco metros.
  - Las medias en velocidad correspondientes a las distribuciones gaussianas nro. 1 y 2 son $\mu_{Vm} = 0.03\ m/s$ y $\mu_{Vm} = 0.11\ m/s$, respectivamente.
  - Las varianzas del ruido de las distribuciones gaussianas nro. 1 y 2 poseen amplitud ascendente, con unidades de m^2/s^2 , y sus valores máximos son iguales a $\sigma^2 = 1.515\ \frac{m^2}{s^2}$y $4.545\ \frac{m^2}{s^2}$.
  - La proporción de mezcla es de 0.9 y 0.1.

De los modelos gaussianos mixtos planteados para los casos 1 y 2, se desarrolló el programa de MATLAB® del apéndice 1.3, presentándose su algoritmo correspondiente a continuación:

Algoritmo 2.3: Generación de modelo mixto gaussiano (GMM)

Entradas: Número de muestras de la trayectoria $\boldsymbol{nt}$, número de segmentos $\boldsymbol{seg}$, número de muestras por segmentos $\boldsymbol{nt\_set}$ y $\boldsymbol{nt\_set2}$

Salidas: Vectores de distribución gaussiana mixta de distancia, acimut y elevación $\boldsymbol{Xgm12t}$ y $\boldsymbol{Xgm34t}$.
Vectores de distribución gaussiana mixta de velocidad $\boldsymbol{Xgm56t}$ y $\boldsymbol{Xgm78t}$
Gráficos QQ y evolución en el tiempo de las distribuciones.

Inicio del programa:

1: Declara la matriz de medias $\boldsymbol{\mu_{12}}$ (GMM de distancia, acimut y elevación. Caso altura mayor a 5 mts)
2: Declara el vector de proporción de mezclas $\boldsymbol{p_{12}}$ (GMM de distancia, acimut y elevación. Caso altura mayor a 5 mts)
3: Declara la matriz de medias $\boldsymbol{\mu_{34}}$ (GMM de distancia, acimut y elevación. Caso altura menor a 5 mts)
4: Declara el vector de proporción de mezclas $\boldsymbol{p_{34}}$ (GMM de distancia, acimut y elevación. Caso altura menor a 5 mts)
5: Declara la matriz de medias $\boldsymbol{\mu_{56}}$ (GMM de velocidad. Caso altura mayor a 5 mts)
6: Declara el vector de proporción de mezclas $\boldsymbol{p_{56}}$ (GMM de velocidad. Caso altura mayor a 5 mts)
7: Declara la matriz de medias $\boldsymbol{\mu_{78}}$ (GMM de velocidad. Caso altura menor a 5 mts)
8: Declara el vector de proporción de mezclas $\boldsymbol{p_{78}}$ (GMM de velocidad. Caso altura menor a 5 mts)
9: Para k **igual a** 1 **hasta** seg **Hacer**
10: Establece las matrices de covarianza de la probabilidad 1 y probabilidad 2 del modelo mixto gaussiano de distancia, acimut y elevación para el caso de altura mayor a 5 mts.
11: Multiplica los elementos de la diagonal de las matrices de covarianza por el número de muestra actual ***k***
12: Declarar la matriz $\boldsymbol{\sigma_{12}}$ mediante el comando de MATLAB® **"Cat"**
13: Crear modelo de distribución mixta $\boldsymbol{gm_{12}}$ mediante el comando de MATLAB® "gmdistribution"
14: **Si** k **igual a** 101 **Hacer**
Generar datos aleatorios ***Xgm12*** con el modelo $\boldsymbol{gm_{12}}$ y cantidad de muestras $\boldsymbol{nt\_set2}$.
15: **de otro modo**
16: Generar datos aleatorios ***Xgm12*** con el modelo $\boldsymbol{gm_{12}}$ y cantidad de muestras $\boldsymbol{nt\_set}$.
17: **Fin Si**
18: Efectuar procedimiento entre las líneas 10 a 17 para obtener ***Xgm34***, ***Xgm56*** y ***Xgm78***
19: **Si** k **mayor a** 1 **Hacer**
20: Concatenar **Xgm12t = [Xgm12old Xgm12]'**
21: Concatenar igual para obtener **Xgm34t, Xgm56t y Xgm78t**
22: **de otro modo**
23: **Xgm12t = Xgm12'**
24: Hacer lo mismo para obtener **Xgm34t, Xgm56t y Xgm78t**
25: **Fin Si**
26: Guardar vector **Xgm12old = Xgm12t** para siguiente iteración
27: Hacer lo mismo para **Xgm34old, Xgm56old** y **Xgm78old**
28: Fin Para
29: Graficar QQ-plots y evolución en el tiempo de las distribuciones

Al ejecutar el programa del apéndice 1.3 se obtuvieron las gráficas Q-Q para el caso nro. 1 (altura mayor a 5 metros) y caso nro. 2 (altura menor a 5 metros), donde Ad, Bdn, Ed y Vm son las variables de distancia, acimut, elevación y velocidad, respectivamente. Se puede observar que las gráficas tienen similar forma a las presentadas en las figuras 1.7 y 1.8, demostrándose que el modelo establecido posee características de ruido angular.

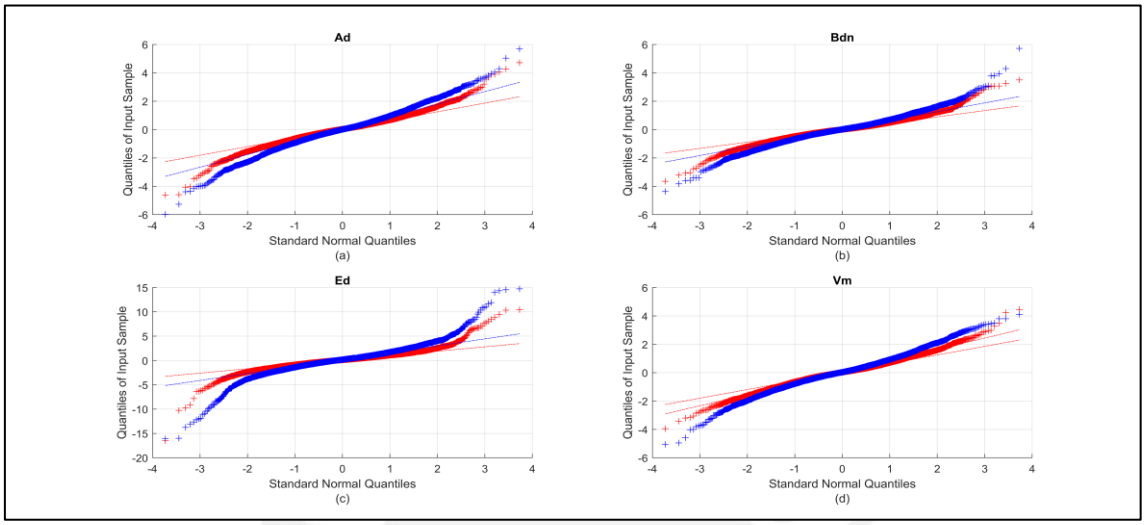

Figura 2.5.: Q-Q plots del modelo mixto gaussiano (GMM). Caso Nro. 1 (color rojo). Caso Nro. 2 (color azul). (a) GMM para alcance Ad. (b) GMM para azimuth verdadero Bdn. (c) GMM para elevación Ed. (d) GMM para velocidad Vm.
Fuente: Elaboración propia.

Por otro lado, en la siguiente figura se presenta la evolución en el tiempo del ruido angular o glint generado, demostrándose que posee distribución mixta y una amplitud ascendente:

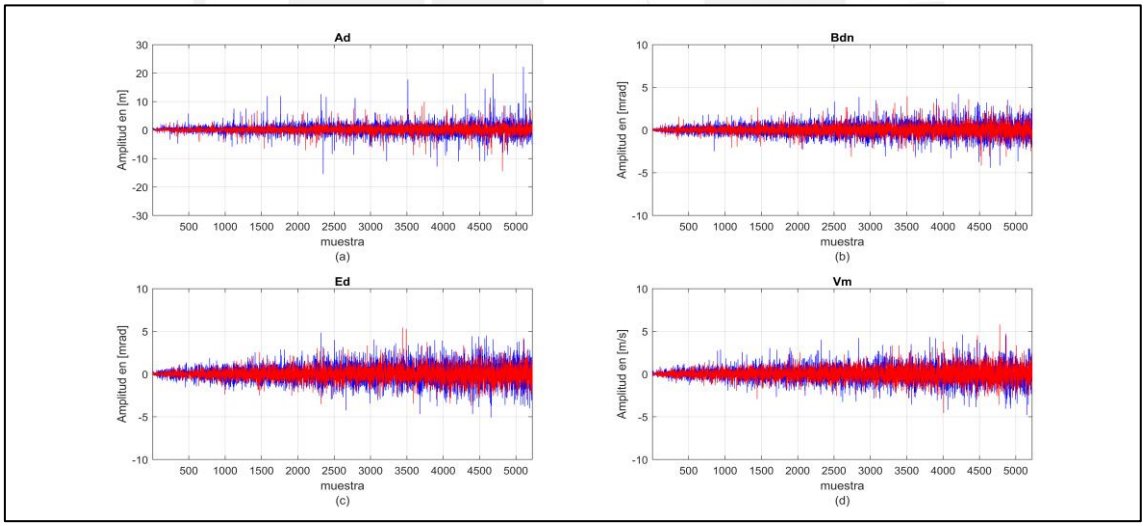

Figura 2.6.: Modelo mixto gaussiano (GMM) en el tiempo. Caso Nro. 1 (color rojo). Caso Nro. 2 (color azul). (a) GMM para alcance Ad. (b) GMM para azimuth verdadero Bdn. (c) GMM para elevación Ed. (d) GMM para velocidad Vm.
Fuente: Elaboración propia.

2.5 Resultados de la simulación del modelo propuesto

A partir del vector de entrada de control u_k, vector de perturbaciones ξ_k y modelo mixto gaussiano generados se efectuó la simulación de una trayectoria "sea skimming" de un misil anti buque hacia un objetivo en el mar. Para tal efecto, se desarrolló el programa del apéndice 1.4, el cual utiliza los datos generados por los programas 2.1, 2.2 y 2.3. Su algoritmo se presenta a continuación:

Algoritmo 2.4: Simulación de trayectoria del blanco

Entradas: Señal de entrada de control $\boldsymbol{\mu_k}$, perturbaciones $\boldsymbol{\xi_k}$ y modelo mixto gaussiano de ruido

Salidas: Vector de variables de estado reales del modelo $\boldsymbol{x_k}$
Vector de variables de estado medidas del modelo $\boldsymbol{y_k}$
Gráficos de evolución en el tiempo de las variables de estado del modelo

Inicio del programa:

1: Declara constantes tales como tiempo de muestreo $\boldsymbol{T}$, gravedad $\boldsymbol{g}$, constante de mach **M**, sobre aceleración máxima ***load_factor*** tiempo inicial de simulación $\boldsymbol{t_i}$, tiempo final de simulación t_{ff} y obtén el número total de muestras $\boldsymbol{nt}$.

2: Declara las condiciones iniciales de traqueo tales como distancia al blanco $\boldsymbol{Ad}$, acimut verdadero al blanco $\boldsymbol{Bdn}$, elevación al blanco $\boldsymbol{Ed}$, velocidad del blanco $\boldsymbol{Vm}$ y rumbo verdadero al blanco $\boldsymbol{Rv}$.

3: Conversión de coordenadas polares a cartesianas de las condiciones iniciales de traqueo.

4: Declara el vector de estados inicial $\boldsymbol{x_k(0)}$

5: Declara la matriz de transición de estados $\boldsymbol{F_k}$, matriz de entrada de control $\boldsymbol{G_k}$, matriz de perturbaciones $\boldsymbol{D_k}$ y matriz de mediciones $\boldsymbol{C_k}$.

6: Ejecuta la función ***gen_jerk_sea_skimming3.m*** **(programa del apéndice 2.5)** para obtener el vector de entrada de control $\boldsymbol{u_k}$

7: Ejecuta la función ***perturb_1.m*** para obtener el vector de perturbaciones $\boldsymbol{\xi_k}$ **(programa del apéndice 2.6)**

8: Cargar modelo de distribución gaussiana mixta ***glint_puntos.mat*** **(datos guardados del programa del apéndice 2.7)**

9: Ingrese "1" para simulación con ruido y "0" para simulación sin ruido.

10: Para tt = ti:T:tff **Hacer**

11: Ejecuta el modelo de transición de estados para obtener $\boldsymbol{x_k}$ futuro.

12: Adquiere datos medidos por el radar: Ejecuta la función ***c2p.mat*** para convertir $\boldsymbol{x_k}$ en coordenadas polares y computa la velocidad Doppler del blanco.

13: Ejecuta la función ***yk_noise.m*** para obtener el vector de mediciones y_k

14: Almacena datos $\boldsymbol{x_k}$ y $\boldsymbol{y_k}$ para graficar posteriormente.

15: Fin Para

16: Conversión de unidades del vector de estados a Mn, Mach y g's.

17: Graficar

18: Fin de programa

La simulación en MATLAB® del programa del apéndice 1.4 fue efectuada con las siguientes condiciones:

- Modelo dinámico del blanco lineal incierto (2.23a) con correlación en el tiempo de la sobre aceleración $\alpha = 0$, ruido de proceso $w_k = 0$ y tiempo de muestreo *T=0.02 seg.*
- Modelo de medición del blanco lineal incierto (2.23b) que representa a un radar en banda X, PRF = 1800 Hz, con capacidad de medición de posición y velocidad Doppler y con ruido de medición v_k simulado de acuerdo al modelo mixto gaussiano diseñado.
- Tiempo de simulación: 104.5 segs (5226 muestras adquiridas por el radar).
- Datos de traqueo iniciales:
  - Distancia inicial al blanco: 35 km
  - Elevación inicial al blanco: 0.01°
  - Acimut inicial al blanco: 030°
  - Velocidad inicial del blanco: 0.9 Mach
  - Rumbo verdadero inicial del blanco: 210°
- Tipo de trayectoria: Sea skimming con maniobra terminal helicoidal.

Como se puede observar en la figura 2.7, el misil se encuentra inicialmente a 35km del objetivo, procede al rumbo 210° a una altura de vuelo de 6 mts, siendo su primer ascenso en la muestra k = 500 a una altitud de 17.15 mts a fin de evitar una isla que se encuentra en la trayectoria del misil al objetivo. El misil se mantiene en esta altura hasta iniciar su descenso en la muestra k = 968, para posteriormente establecerse a una altitud de 1.974 mts sobre el nivel del mar. En la muestra k = 4394 inicia su ascenso a una altura de 10.97 mts para prepararse para iniciar su maniobra terminal tipo helicoidal, la cual inicia en la muestra k = 4650. Posteriormente inicia su descenso a la altura de impacto programada, a fin de destruir al objetivo. Cabe resaltar que el misil efectúa cambios de rumbo durante la trayectoria en el plano x-y a fin de evitar obstáculos y prevenir que el objetivo pueda deducir la dirección de proveniencia de este.

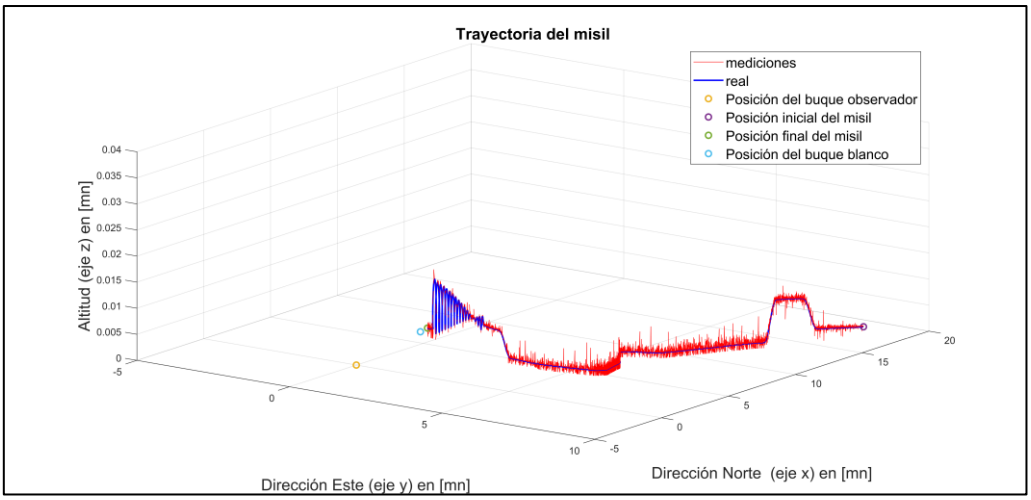

Figura 2.7.: Trayectoria en tres dimensiones del misil (modelo). Real (línea azul). Medida (línea roja).
Fuente: Elaboración propia.

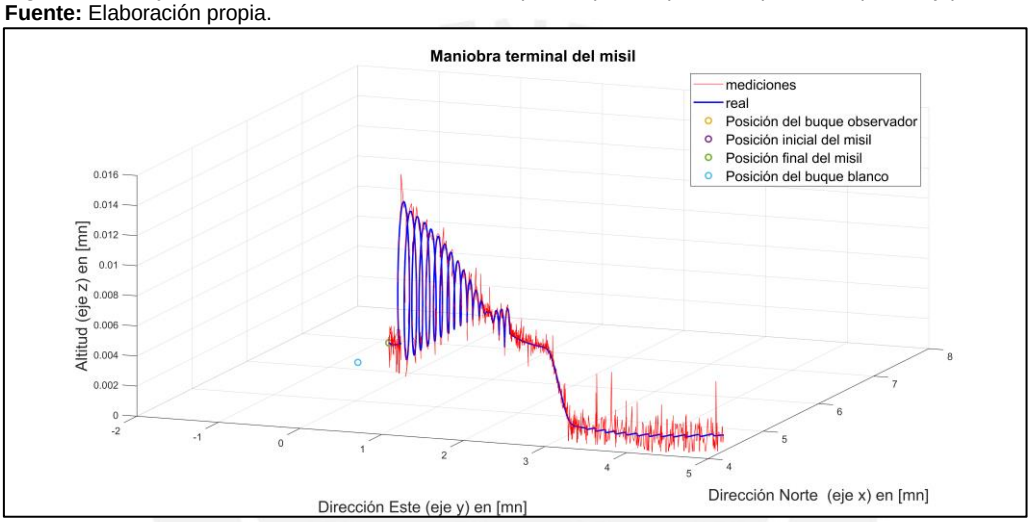

Figura 2.8.: Maniobra terminal del misil (modelo). Real (línea azul). Medida (línea roja).
Fuente: Elaboración propia.

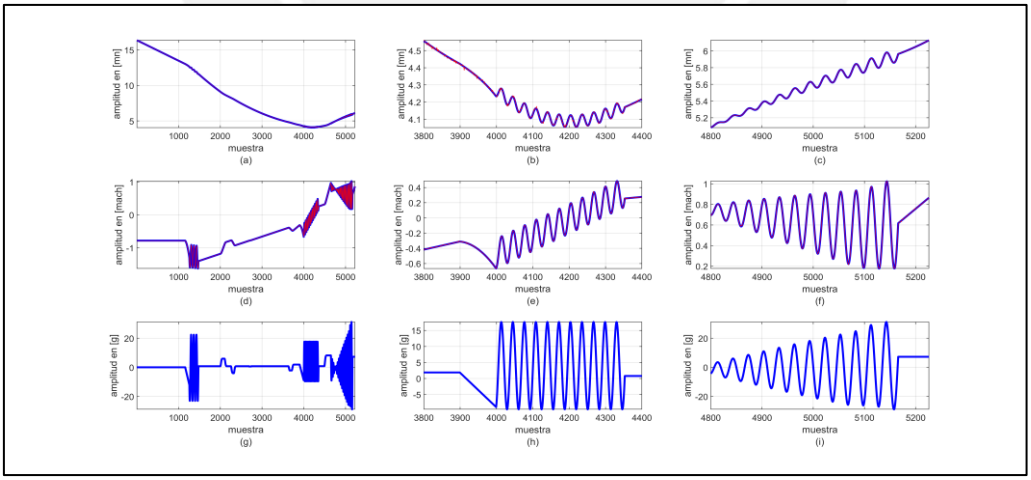

Figura 2.9.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "x" (modelo). Variable de estado real (línea azul). Variable de estado medida (línea roja). (a) Posición (k = [0 5226]). (b) Posición (k = [3800 4400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [3800 4400]) (i) Aceleración (k = [4800 5226]).
Fuente: Elaboración propia.

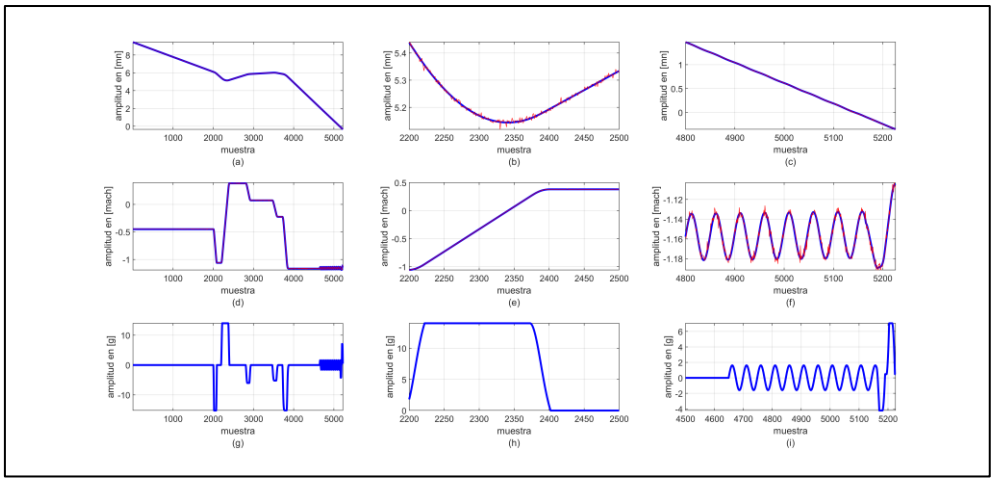

Figura 2.10.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "y" (modelo). Variable de estado real (línea azul). Variable de estado medida (línea roja). (a) Posición (k = [0 5226]). (b) Posición (k = [2200 2500]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [2200 2500]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [2200 2500]) (i) Aceleración (k = [4800 5226]).
Fuente: Elaboración propia.

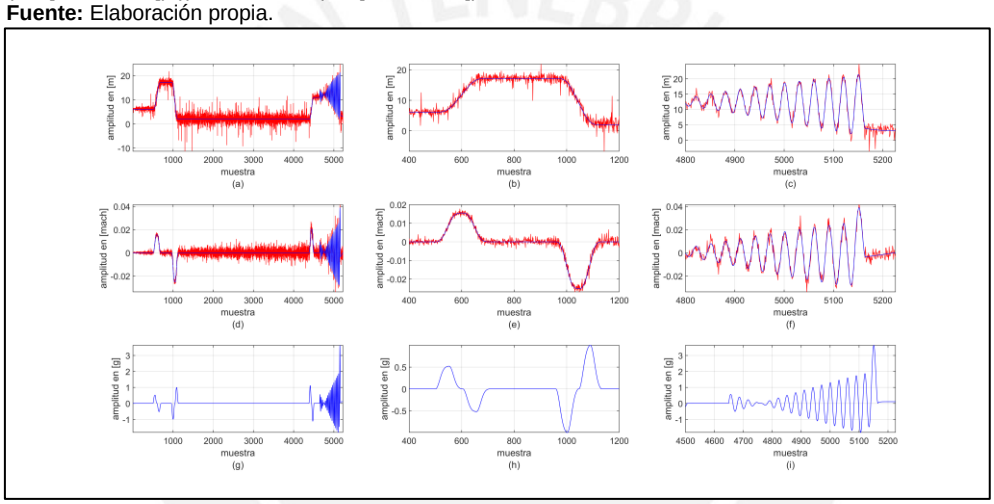

Figura 2.11.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "z" (modelo). Variable de estado real (línea azul). Variable de estado medida (línea roja). (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [400 1200]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [400 1200]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

2.6 Conclusiones del capítulo

Al analizar los resultados obtenidos de la simulación del modelo propuesto se llegan a las siguientes conclusiones:

1) Se logró obtener una trayectoria sea-skimming realista de un blanco aéreo de alta maniobrabilidad de tipo misil antibuque mediante simulaciones en MATLAB®.
2) El blanco aéreo de alta maniobrabilidad simulado presenta comportamiento lineal durante los trayectos rectos de punto a punto y comportamiento no lineal durante los cambios de rumbo, maniobras evasivas y maniobra terminal helicoidal a partir de la introducción de perturbaciones o incertidumbres estructuradas.
3) Las perturbaciones generadas permiten obtener una maniobra terminal de tipo helicoidal al inyectar al modelo lineal incierto funciones senoidales y cosenoidales acopladas en las tres coordenadas.
4) Se ha logrado obtener un vector de entrada de control $\boldsymbol{u_k}$ dentro de los parámetros máximos permisibles de aceleración y sobre aceleración. Sin embargo, este vector deberá ser estimado por algún método debido a que este no puede ser medido.
5) El modelo de distribución mixta gaussiana diseñado es adecuado para simular ruido angular de radar o *glint noise* dado que muestra características no gaussianas.

Capítulo III

Diseño de observadores robustos basados en sistemas de estructura variable con modos deslizantes

3.1 Introducción

En teoría de control se dice que un sistema es ***robusto*** si se mantiene estable y logra obtener índices de desempeño aceptables a pesar de la presencia de incertidumbres y perturbaciones que afecten a la planta o proceso según Ogata [43]. Para tal efecto, el diseño de controladores y observadores de variables de estado bajo el marco de la teoría de control robusto parte de la asunción que el modelo que se utiliza durante el diseño contiene errores [43], siendo este el caso del problema en estudio.

El diseño robusto permite obtener controladores u observadores de variables de estado que cuenten con las siguientes propiedades, según lo expuesto por Gu et al. [44]:

- **Estabilidad robusta:** El controlador u observador diseñado es estable ante la presencia de perturbaciones, parámetros del modelo y dinámicas parásitas desconocidas.
- **Desempeño robusto:** El controlador u observador diseñado presenta características de respuesta en frecuencia o tiempo preestablecidas ante la presencia de perturbaciones, parámetros del modelo y dinámicas parásitas desconocidas.

En el Capítulo II se expone que un gran número de modelos propuestos en la literatura para los blancos aéreos de alta maniobrabilidad cuentan con errores de modelado, debido a la dinámica cambiante del blanco durante su trayectoria. Por ello, el estado del arte propone el uso de varios modelos en simultáneo y un algoritmo de detección de la dinámica del blanco a fin de poder mejorar la performance de traqueo o seguimiento. Cabe resaltar que en muchos casos los modelos utilizados son linealizados por algún método en particular. Sin embargo, al final del capítulo II se propone un modelo lineal del blanco con incertidumbres que representan la dinámica no lineal del blanco durante cambios rumbo, maniobras evasivas y maniobras terminales. Por tanto, para el adecuado diseño de un observador de variables de estado que permita la estimación adecuada de posición, velocidad y aceleración es necesario aplicar principios de control robusto aplicado a la observación de estados.

Un observador de variables de estado es en esencia una réplica matemática de la planta o proceso, el cual permite estimar los estados no medibles de esta a partir de las entradas y salidas medibles [43]. Por tanto, se puede clasificar a los observadores de la siguiente manera:

- Observadores de orden completo o *Full-State State Observers*: Aquellos observadores que, independientemente del número de variables de estado que sean medibles, efectúa la observación de todas las variables de estado [43].
- Observadores de orden reducido o *Reduced-Order State Observers:* Aquellos observadores que estiman un número menor que la cantidad total de variables de estado [43].

En la figura 3.1 se muestra el diagrama de bloques de un observador de estados de orden completo, donde en la parte superior se puede observar el diagrama de bloques del espacio de estados del sistema lineal con matriz de transición de estados *A*, matriz de entrada de control *B*, matriz de observación *C* y siendo la entrada al sistema el vector de entrada de control u, x el vector de variables de estado del sistema e *y* el vector de estados medibles. En la parte inferior se observa al observador de estados de orden completo, el cual tiene como entradas al vector de entrada de control *u* y al vector de mediciones *y.* En su estructura contiene a las matrices del sistema lineal y en adición una ganancia de retroalimentación lineal del error entre las variables de estado estimadas $\tilde{y}$, correspondiente a las variables de estado medidas del sistema, y las variables de estado reales medidas del sistema. Esto permite llevar el error a cero en un tiempo finito y que el vector de estados estimados $\tilde{x}$ representa de forma exacta las variables de estado reales del sistema lineal.

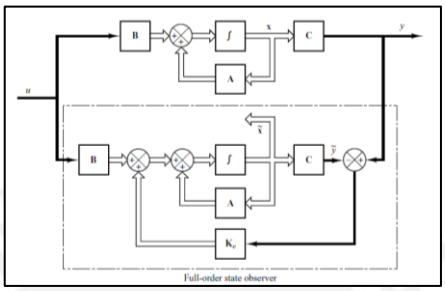

Figura 3.1.: Observador de estado de orden completo.
Fuente: Ogata [42].

Este observador lineal es el denominado observador de Luenberger, propuesto por primera en el año de 1964. Cabe resaltar que este tipo de observadores lineales muestran divergencias de los estados estimados en relación a los reales cuando se presentan perturbaciones, errores en el modelamiento y ruido de medición. Por ello, la solución de mayor uso para la estimación de variables de estado con la presencia de ruido de medición es por excelencia el filtro de Kalman. Sin embargo, se conoce que este tipo de filtros no cuentan con robustez innata para mitigar perturbaciones o incertidumbres estructuradas e incertidumbres paramétricas que pueda tener el modelo. Por ello, la teoría de control de sistemas de estructura variable o *Variable Structure Systems* (VSS por sus siglas en inglés) se presenta como una opción interesante para el problema en estudio, dado que permite la observación robusta de una planta o proceso con incertidumbres o perturbaciones.

Los sistemas de estructura variable, originados en la Unión Soviética durante la década de los años sesenta, son aquellos sistemas de lazo cerrado que cuentan con un control que cambia su estructura en el tiempo a fin de mantener un desempeño deseado del sistema durante el proceso de control [45]. Para tal efecto, consta de un conjunto de subsistemas continuos y una lógica de conmutación entre estos que originan acciones de control discontinuas. Al respecto, sea el sistema $\dot{x} = f(t,x,u)$ en el dominio $D \in \mathbb{R}^n$, el control por estructura variable puede ser expresado de la siguiente manera:

$$u = \begin{cases} u_1(t,x), & (t,x) \in \mathbb{R}^+ \, x \, D_1 \\ u_2(t,x), & (t,x) \in \mathbb{R}^+ \, x \, D_2 \\ \quad . & \\ u_q(t,x), & (t,x) \in \mathbb{R}^+ \, x \, D_q \end{cases}$$

donde, las funciones $\mathrm{u_i(t,x)}$ son continuas para $i = 1,2,\ldots,q$. La estructura de las funciones $u_i(t,x)$ y $u_j(t,x)$ son diferentes para $i \neq j$ y $i,j = 1,2,\ldots,q$ $(q \geq 2)$. Asimismo, los dominios $D_i \in \mathbb{R}^n$ para $i = 1,2,\ldots,n$ deben satisfacer $D_1 \cup D_2 \cup \ldots \cup D_n = D$ y $D_i \cap D_j = \emptyset$ si $i \neq j$ para $i,j = 1,2,\ldots,q$. Cuando el control por estructura variable es aplicado al sistema $\dot{x} = f(t,x,u)$, el sistema de lazo cerrado resultante es denominado un *sistema de estructura variable*. En particular, los sistemas de control de modos deslizantes o *Sliding mode control* (SMC por sus siglas en inglés) son un tipo de sistemas de estructura variable, diseñados para llevar y mantener las variables de estado en el vecindario de la función de conmutación, a partir de una función del error conocida como superficie deslizante; estos sistemas permiten diseñar un comportamiento dinámico a partir de la elección de una función discontinua de conmutación en particular,

obteniendo una respuesta en lazo cerrado del sistema inmune a incertidumbres. Sin embargo, la generación de la señal de inyección discontinua que permite llevar el error del sistema a cero en un tiempo finito genera el fenómeno de castañeo o *chattering*, el cual se presenta como oscilaciones discontinuas de alta frecuencia, las cuales pueden ser reducidas por el diseñador por medio de distintos métodos de requerirse.

Una posible clasificación de los sistemas de estructura variable en base a modos deslizantes, propuesta por Shtessel et al [46] es la siguiente:

- **Primera generación:** El control de modos deslizantes convencional (SMC), incluyendo sistemas en forma canónica, sistemas SISO y MIMO en espacio de estados arbitrario, sistemas de dimensión infinita, control integral, entre otros. A partir de esta teoría de control nacen los observadores de modos deslizantes "clásicos".
- **Segunda Generación:** Algoritmos "Twisting" y "Terminales" para sistemas de segundo orden. A partir de esta teoría de control nacen los observadores con este tipo de algoritmos aplicado a sistemas de segundo orden.
- **Tercera Generación:** Algoritmos "Super-Twisting", "Super-Twisting" de orden arbitrario y "Super-Twisting" de orden superior o también conocido como HOSM. De estos algoritmos de control nacen observadores "Super-Twisting" para sistemas de segundo orden, "Super-Twisting" de orden arbitrario para sistemas por encima del segundo orden, el observador de modos deslizantes de orden superior (HOSMO) y el diferenciador robusto exacto (RED).
- **Cuarta Generación:** Algoritmos anidados, los cuales no son considerados como sistemas de estructura variable por algunos autores tales como Vadim Utkin et al. [47] debido a que según su perspectiva no logran obtener un modo deslizante que los caracterice como observadores de modos deslizantes.
- **Quinta Generación:** Algoritmos Cuasi continuos, similares a los de cuarta generación. Estos no serán investigados en esta tesis.

La teoría de Sistemas de Estructura Variable por modos deslizantes aplicado algoritmos de observación permite diseñar observadores con la capacidad de efectuar la reconstrucción de la perturbación a la planta o proceso por medio del control equivalente o al introducir al observador una variable adicional que represente la integral de un término discontinuo, sin embargo, en Zhang et al. [48] se explica que existe una cierta oposición por la comunidad de diagnóstico de fallas a incluir a los observadores de modos deslizantes en la teoría de diagnóstico de fallas, dado que según su perspectiva no ofrecen la misma calidad de reconstrucción de fallas que poseen los algoritmos de diagnóstico de fallas. Por tanto, en el presente capítulo se efectúa el diseño de dos observadores de estados basados en sistemas de estructura variable por modos deslizantes, denominados "clásicos" (según lo expuesto por Shtessel et al. [46]), de Edwards-Spurgeon y Walcott-Zak, los cuales muestran distintas técnicas para sintetizar las ganancias lineales y no lineales del observador, y posteriormente se aborda lo relacionado a la teoría de algoritmos de super torsión o "super-twisting" por medio del estudio de los observadores de orden superior (HOSMO) y los diferenciadores robustos exactos. Al final del capítulo se efectúa una comparación de desempeño que servirá como guía para la propuesta de solución que se hace en el capítulo IV.

3.2 Fundamentos teóricos de sistemas de estructura variable

Distintos autores consideran que los sistemas de estructura variable son una extensión del método de plano de fase de la teoría de osciladores, propuesto por Alexandr Andronov et al. [49], a fines de la década de los años cuarenta; Andronov propone seleccionar una superficie en el plano de fase que permita obtener funciones de control discontinuas. Posteriormente, la teoría de control robusto de modos deslizantes es propuesta formalmente en los estudios de Stanislav Emel'yanov en la década de los años sesenta, la cual se caracteriza por describir un movimiento en el plano de fase que ocasiona que la trayectoria de los estados del sistema, en la vecindad de la superficie donde la función de control presenta discontinuidades, sean dirigidas hacia esa superficie. Por ello, ante perturbaciones o incertidumbres que afectan a los estados de la trayectoria los modos deslizantes presenta insensibilidad, evitando la necesidad de efectuar un modelamiento exacto, según lo expuesto por Utkin et al. [47]. Por ello, estos sistemas han sido extensamente aplicados principalmente como soluciones de control en los campos de robótica, control de procesos, control de movimiento, entre otros; Vadim Utkin [50] expone extensamente

la teoría de sistemas de estructura variable por modos deslizantes, continuando con las investigaciones realizadas por Emel'yanov y Barbashin.

3.2.1 Sistema de estructura variable de modos deslizantes

Considérese, como ejemplo, un sistema de tipo relé de segundo orden, el cual es naturalmente un sistema de estructura variable por poseer la capacidad de conmutar entre un estado alto y uno bajo, el cual puede ser representado mediante la siguiente ecuación de segundo orden:

$$\ddot{x}(t) + a_2\dot{x}(t) + a_1x(t) = u(t) \quad (3.1a)$$
$$u(t) = -Msign(s) \quad (3.1b)$$
$$s(t) = cx(t) + \dot{x}(t)^{8} \quad (3.1c)$$

Donde $a_1, a_2, M\ y\ C$ son constantes, $s(t)$ es el denominado colector o superficie deslizante y $u(t)$ es una señal de control que presenta discontinuidades a lo largo de la línea $s(t) = 0$ en el plano de fase $(x, \dot{x})$. La señal de control discontinua $u(t)$ solo toma dos valores, $-M\ y\ M$ dado que posee dos estructuras continuas inestables dependientes del signo de la función $s(t)$.

En la figura 3.2, sub figura de la izquierda, se observa que cuando la función $u(t)$ toma el valor $u(t) = -M$, el mapa de fase del plano $x - \dot{x}$ corresponde a un foco inestable, y en la sub figura central se observa que cuando la función es igual a $u(t) = M$ el mapa de fase corresponde a un punto de silla, por lo que es evidente que estas estructuras, cada una por sí sola, no permitirían que el error del sistema sea asintóticamente estable. Al establecer el sistema de estructura variable mediante la función discontinua $u(t) = -Msign(s)$, el mapa de fase del plano $x - \dot{x}$ se convierte en un nodo atractor o foco estable, de acuerdo a la sub figura de la derecha. Por tanto, para cualquier condición inicial las variables de estado alcanzarán el colector deslizante $s(t) = 0$ por medio de movimientos discontinuos, desarrollando el denominado "modos deslizantes"[9].

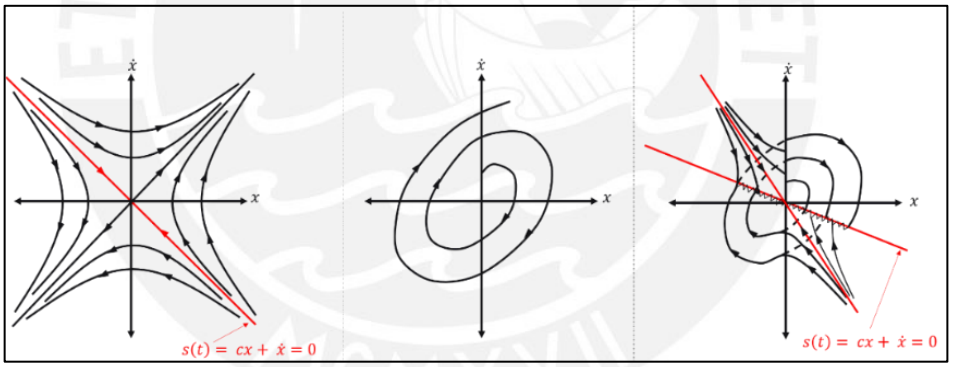

Figura 3.2.: Sistema de Estructura Variable.
Fuente: Sabanovic et al. [51].

3.2.2 Condiciones de existencia de los modos deslizantes

Sea el sistema de la forma general,

$$\dot{x}(t) = f(x, u, t) \quad (3.2a)$$

$$u_i(x,t) = \begin{cases} u_i^{+}(x,t)\ , & si\ s_i(x) > 0 \\ u_i^{-}(x,t), & si\ s_i(x) < 0 \end{cases} \quad i = 1,2,\dots,m \quad (3.2b)$$

donde x representa el vector de estados del sistema, de dimensión $n\ x\ 1$, u es el vector de entrada de control, de dimensión $m\ x\ 1$ y $u_i^{+}(x,t)$, $u_i^{-}(x,t)$ y $s_i(x)$ son funciones continuas que

[8] La ecuación del colector o superficie deslizante siempre deberá ser de orden menor que el orden de la ecuación del sistema. Esto permite que siempre se trabaje con un sistema de orden reducido, simplificando el diseño.
[9] Se debe considerar que después de iniciado el modo deslizante, el movimiento de los estados en el colector deslizante no dependerán de los parámetros de la planta o proceso [49].

cumplen $u_i{}^+(x,t) \neq u_i{}^-(x,t)$. Como se puede apreciar, (3.2b) indica que la función $u_i(x,t)$ presenta discontinuidades en alguna superficie en el espacio de estados del sistema [49].

Teorema 3.1 (Existencia de un modo deslizante [50]): *Sea el sistema planteado en (3.2a) y su control discontinuo (3.2b), se dice que un modo deslizante existe si hay regiones sobre la superficie* $s(t) = 0$ *con valor distinto a cero donde las proyecciones de los vectores* $f^+ = f(x,t,u^+)$ *y* $f^- = f(x,t,u^-)$ *sobre la gradiente superficial son de signo opuesto y direccionados hacia la superficie. Por tanto, se debe cumplir* $\lim_{s\to+0} \dot{s} < 0$ *y* $\lim_{s\to-0} \dot{s} > 0$.

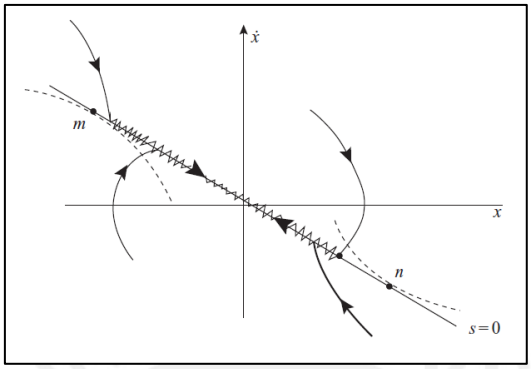

Figura 3.3.: Modos deslizantes
Fuente: Sabanovic et al. [50].

En la figura 3.4 se aprecia que los modos deslizantes se pueden lograr tanto en una superficie deslizante (segmentos 12 y 23 de la superficie deslizante $s_j(x) = 0$) así como en la intersección de estas (segmento 24 de la intersección de las superficies deslizantes $s_j(x) = 0$ y $s_i(x) = 0$. Por ello, se puede concluir que para el sistema (3.2a) pueda existir un modo deslizante en la intersección de todas las superficies discontinuas $s_{1,i,j,..,m}(x) = 0$ o en el colector $s(x) = 0$ de dimensión $(n-m)\ x\ 1$.

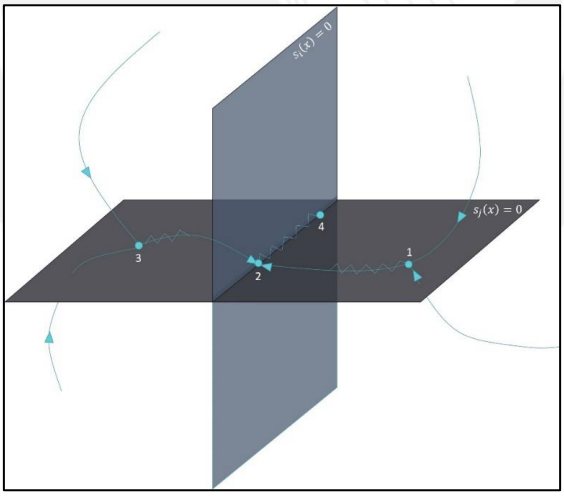

Figura 3.4.: Colectores y modos deslizantes.
Fuente: Sabanovic et al. [50].

3.2.3 Método de continuación de Filippov para ecuaciones diferenciales con discontinuidades en el lado derecho

De acuerdo a lo expuesto por Utkin [50] los sistemas de ecuaciones que cuentan con términos discontinuos en su lado derecho no satisfacen los teoremas convencionales de existencia y unicidad de las soluciones. Por ello, es necesario introducir las definiciones y teoremas efectuados por Alexandr Filippov [52] dado que las ecuaciones descritas en (3.2b) del sistema dinámico discontinuo planteado en (3.2a) solo representan el movimiento del sistema afuera de los límites de discontinuidad, quedando inconcluso la representación de la dinámica del sistema sobre la superficie de discontinuidad.

.

3.2.3.1 Definiciones básicas

Definición 3.1 (Existencia de una solución para una ecuación diferencial continua [52]): *Una solución de la ecuación diferencial* $\frac{dx}{dt} = f(t,x)$ *con lado derecho continuo es una función de* $x(t)$*, la cual tiene una derivada y satisface esta ecuación en cualquier de un intervalo dado.*

La definición 3.1 no es válida para las ecuaciones diferenciales con discontinuidades en el lado derecho de la ecuación. Por ello, es necesario presentar definiciones adicionales para poder plantear una solución apropiada para este tipo de ecuaciones.

Definición 3.2 (Función a intervalos [52]): *Sea* x *un punto en el espacio de dimensión* n *con coordenadas* $x_1, x_2, \dots, x_n$*;* $|x_n| = (x_1, x_2, \dots, x_n)^{1/2}$*. Una función* $f(x,t)$ *es denominada función a intervalos en el dominio finito* G *de un espacio* (x,t) *de dimensión* $(n+1)$ *si el dominio* G *consiste en un número finito de dominios* $G_i (i = 1, \dots, l)$*, donde en cada uno de estos la función* f *es continua hasta el límite, y un conjunto M de medida cero que es conformado por puntos límites de estos dominios. La función es contin0ua en el dominio hasta el límite si, cuando su argumento se aproxima a cada punto del límite, la función tiende a un límite finito, posiblemente a límites diferentes para puntos límites.*

Definición 3.3 (Hiper superficie [52]): *En un espacio m-dimensional un conjunto* S *es denominado hiper superficie k-dimensional si en la vecindad de cada uno de sus puntos "a" todas las coordenadas de los puntos del conjunto S son funciones continuas de algunas "k" de estas coordenadas, que varían sobre un dominio k-dimensional cierto* ${G^k}_a$*.*

3.2.3.2 Inclusión Diferencial

Definición 3.4 (Inclusión Diferencial [52]): *Sea* $\dot{x} \in F(t,x)$ *(3.3) la denominada Inclusión Diferencial. Esta es, una función vectorial absolutamente continua* $x(t)$ *definida en el intervalo o en un segmento* I *para el cual* $\dot{x}(t) \in F(t, x(t))$ *casi en todo el segmento* I*.*

Teorema 3.1 (Solución de una inclusión diferencial [52]): *Sea el sistema y dominios planteados en la definición (3.2). Se define un conjunto M de medida cero conformado por puntos de discontinuidad de la función* f*, definiéndose para cada punto* (t,x) *del dominio* G *un conjunto* $F(t,x)$ *en un espacio n-dimensional. Si en el punto* (t,x) *la función* f *es continua, el conjunto* $F(t,x)$ *consiste de un punto que coincide con el valor de la función* f *en ese punto. Si* (t,x) *es un punto de discontinuidad de la función* f*, el conjunto* $F(t,x)$ *será definido de otra forma. Una solución al sistema planteado en la definición (3.2) es conocido como una solución a una inclusión diferencial.*

3.2.3.3 Solución para una ecuación diferencial con discontinuidades en su lado derecho

Con la finalidad de representar a la función $F(t,x)$ en los puntos discontinuidad de la función f existen diversos teoremas, tales como el método convexo y el método de control equivalente, siendo este último utilizado en modos deslizantes. Al respecto, las ecuaciones del sistema planteado en (3.2a) y su control discontinuo (3.2b) determinan el comportamiento del sistema fuera de los límites de discontinuidad, por lo que queda inconcluso la representación del comportamiento del sistema en las superficies deslizantes. Por ello, como solución a este problema se utiliza el método de control equivalente, en el cual se igualan a cero las derivadas de la superficie $s_i(x)$ y el sistema de ecuaciones resultantes es resuelto despejando los componentes del vector de control u. La solución es denominada vector de control equivalente u_{eq}, el cual es sustituido en el sistema original (3.2a) para obtener las ecuaciones de modos deslizantes ideales [52].

Definición 3.5 (Discontinuidad del conjunto cerrado $U_i(t,x)$ [52]): *Sea el sistema de ecuaciones diferenciales,*

$$\dot{x} = f\big(t, x, u_1(t), \dots, u_r(t)\big) \tag{3.4}$$

donde $x \in \mathbb{R}^n$, $f(t, x, u_1(t), \dots, u_r(t))$ *es continua para el conjunto de argumentos, la función o escalar* $u_i(t)$ *es discontinua para el conjunto* $M_i, i = 1, \dots, r$ *los cuales poseen puntos comunes y hasta podrían coincidir. En cada punto* (t, x) *de la función de discontinuidad* u_i *debe existir un conjunto cerrado* $U_i(t, x)$ *que representa los posibles valores del argumento* u_i *de la función* $f(t, x, u_1(t), \dots, u_r(t))$*. En estos puntos de discontinuidad de la función* u_i *el conjunto cerrado* $U_i(t, x)$ *debe contener todos los puntos límites para cualquier secuencia de la forma* $v_k \in U_i(t_k, x_k)$*, donde* $x_k \to x$*,* $k = 1,2, \dots$ *; Se requiere usualmente que el conjunto cerrado* $U_i(t, x)$ *sea convexo.*

Teorema 3.2 (Representación de $F(t, x)$ en los puntos discontinuidad de la función f por el método de control equivalente [52]): *Sea* $F_1(t, x) = f(t, x, U_1(t), \dots, U_r(t))$ *un conjunto de valores de la función* $f(t, x, u_1(t), \dots, u_r(t))$*, donde* t, x *son constantes y* $u_1, \dots, u_r$ *varían independientemente uno del otro en los conjuntos* $U_1(t, x), \dots, U_r(t, x)$*, respectivamente. Las soluciones del sistema de ecuaciones diferenciales planteado en la definición (3.4) son soluciones de la inclusión diferencial (3.3) planteada en el teorema (3.1), donde* $F(t, x) = F_1(t, x)$*. En los puntos que pertenecen a una superficie, o simultáneamente a varias superficies* $S_1, \dots, S_m$ *, donde* $1 \leq m \leq r$*, se asume, si la solución no puede escapar inmediatamente de aquella superficie o de la intersección de tales superficies, lo siguiente:*

$$\dot{x} = f(t, x, {u_1}^{eq}(t, x); \dots, {u_m}^{eq}(t, x), {u_{m+1}}^{eq}(t, x), \dots, {u_r}^{eq}(t, x)) \qquad (3.5)$$

Donde los controles equivalentes ${u_1}^{eq}, \dots, {u_m}^{eq}$ *son definidos de tal forma que la función vectorial f en (3.4) es tangente a las superficies* $S_1, \dots, S_m$ *y el valor* ${u_i}^{eq}(t, x)$ *está contenido en el intervalo* $[{u_i}^-(t, x), {u_i}^+(t, x)]$*. Los valores* ${u_i}^-(t, x), {u_i}^+(t, x)$ *son los valores límites de la función* u_i *en ambos lados de la superficie* S_i*, i = 1,…,m. Entonces, las funciones* ${u_i}^{eq}(t, x)$*,* $i = 1, \dots, m$*, son determinadas del sistema de ecuaciones,*

$$\nabla \varphi_i(x) \cdot f(t, x, {u_1}^{eq}, \dots, {u_m}^{eq}, u_{m+1}(t, x), \dots, u_r(t, x)) = 0, \qquad i = 1, \dots, m \qquad (3.6)$$

Una solución es una función vectorial absolutamente continua la cual fuera de las superficies S_i *satisface la ecuación (3.4) y sobre estas superficies y sus intersecciones satisface la ecuación (3.5).*

3.2.3.4 Significado físico del control equivalente

En el modo deslizante ideal la función de control presenta discontinuidades a una frecuencia infinita y con un vector de velocidad dirigido de forma precisa a la superficie de discontinuidad, tal y como se describe en Sabanovic et al. [51]. Sin embargo, en la realidad existen no linealidades tales como histéresis, zonas muertas, retardos en la conmutación, entre otros, [50] que hacen que el modo deslizante ideal cuente con una función de control con conmutación finita, por lo que en esta teoría de control se habla acerca del control equivalente.

El control equivalente, según Vadim Utkin, es un procedimiento formal por el cual se obtienen ecuaciones de modos deslizantes que coinciden con un valor promedio de la función de control u_{av} [50]. Se conoce que en un tiempo finito el valor de u_{av} tiende a u_{eq} si es que para cualquier par de números positivos $\Delta t < T$ y ε existe un valor $\delta > 0$ tal que para $0 < \tau \leq \delta$ y $\frac{\Delta}{t} \leq \delta$ la desigualdad $\|u - u_{eq}\| \leq \varepsilon$ es válida para $\Delta t \leq t \leq T$ [49]. Por tanto, según lo descrito por Vadim Utkin:

> *"El control equivalente tiene un significado físico tangible debido a que esta función presenta el comportamiento, por ejemplo, de la salida de un filtro de primer orden si es que la constante de tiempo del filtro es calibrada en relación a las dimensiones de la superficie deslizante [50]".*

3.2.3.5 Conjuntos convexos y funciones de valores establecidos

Definición 3.6 (Definición de conjunto cerrado y convexo [52]): *Un conjunto A es cerrado y convexo si este contiene todos sus puntos límite y si para cualquier par de puntos a y b todos los puntos de un segmento que une los puntos a y b pertenecen a este conjunto. Esto es, si para cualquier punto* $a \in A, b \in A$, *se tiene* $\alpha a + (1 - \alpha)b \in A$ *para todo* α, $0 \leq \alpha \leq 1$. *Las siguientes afirmaciones puede ser fácilmente demostradas:*

- *La unión de un conjunto finito de conjuntos cerrados es cerrada.*
- *La intersección de cualquier conjunto de conjuntos cerrados o convexos es un conjunto cerrado, y consiguientemente convexo.*
- *En un conjunto no vacío* A *siempre habrá un punto* a *cerca de un punto* b, *tal que* $\rho(b, a) = \rho(b, A)$.
- $\rho(b, A) = \rho(b, \bar{A})$, $\rho(A, B) = \rho(\bar{A}, \bar{B})$.
- *La función* $\varphi(x) = \rho(x, A)$ *es continua,* $|\rho(x, A) - \rho(y, A)| \leq \rho(x, y)$.

Lema 3.1 ([52]): Si los conjuntos cerrados no vacíos A y B no poseen puntos comunes y B es acotado, entonces existen puntos $a \in A$ y $b \in B$, tal que:

$$\rho(A, B) = \rho(a, b) > 0 \tag{3.7}$$

Lema 3.2 (Relación entre dos puntos pertenecientes a un conjunto convexo cerrado [52]): En un conjunto convexo cerrado no vacío A existe solo un punto a más cercano a un punto b, tal que:

$$\rho(b, a) = \rho(b, A) \tag{3.8}$$

Lema 3.3 (Existencia de un plano de dimensión $n - 1$ [52]): Sea $b \notin A$, siendo A un conjunto cerrado convexo no vacío. Por tanto, existe un plano de dimensión $n - 1$ que separa el punto b del conjunto A.

Lema 3.4 (Intersección de medios espacios cerrados [52]): Un conjunto cerrado convexo A es una intersección de todos los medios espacios cerrados que contienen este conjunto.

Lema 3.5 (Separación de conjuntos por plano de dimensión [52]): Si A y B son conjuntos cerrados convexos en $\mathbb{R}^n$ sin puntos comunes, y el conjunto B es acotado, entonces existe un plano de dimensión $n - 1$ que separa a A y B.

Lema 3.6 (Trazado de un plano de soporte [52]): A través de cualquier punto del límite Γ de un conjunto cerrado convexo A se puede trazar un plano de soporte.

> **Demostración del Lema 3.6** ([52]): *Sea* $a \in \Gamma$, *los puntos* $b_i \notin A$, $b_i \to a$ $(i \to \infty)$. *De acuerdo al lema (3.3) el punto* b_i *es separado de* A *por un plano* P_i. *Si* v_i *es un vector de longitud 1, cumpliéndose* $v_i \perp P_i$ *y direccionado de* a *hacia* P_i. *Entonces para todo* $x \in A, y \in P_i$ *se tiene* $v_i \cdot x < v_i \cdot y < v_i \cdot b_i$. *De la secuencia* v_i *se escoge una subsecuencia convergente* $v_i \to v$. *Pasando al límite en esta subsecuencia, se obtiene* $v \cdot x \leq v \cdot a$ *para todo* $x \in A$, *esto es, el conjunto* A *se ubica en un lado del plano* $v \cdot x = v \cdot a$ *y el punto* a *se ubica sobre este plano. Por consiguiente, este plano en un plano de soporte. Demostración finalizada.*

El conjunto convexo más pequeño que contiene el conjunto A se denota como co A, el cual siempre existe y es la intersección de todos los conjuntos convexos que contienen a A. Dado el lema (3.4), $\overline{\text{co}}$ A es también la intersección de todos los medios espacios cerrados conteniendo a A. Asimismo, cada punto del conjunto A descrito de la forma,

$$x = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_m x_m \tag{3.9}$$

donde,

$$\alpha_i \geq 0 \ (i = 0,1, \ldots, m), \qquad \alpha_1 + \alpha_2 + \cdots + \alpha_m = 1 \tag{3.10}$$

Por consiguiente, (3.9) se denomina una combinación convexa de los puntos $x_1, x_2, \dots, x_m$, siendo esta lineal. Sin embargo, no todas las combinaciones lineales son convexas salvo las que cumplen la condición (3.10).

Lema 3.7 (Conjunto de combinaciones convexas [52]): Si un conjunto A consiste de un número finito de puntos, $co\ A$ es el conjunto de todas las combinaciones convexas de estos puntos.

Lema 3.8 (Conjunto de combinaciones convexas complementario [52]): Sea un vector c, un conjunto A y sea la desigualdad $c \cdot x \leq \gamma$ válida para todo $x \in A$. Por tanto, esto también es válido para todo $x \in \overline{co}\ A$.

Definición 3.7 (Definición de ecuación diferencial de Caratheodory [52])**:** *Se conoce que la ecuación diferencial de la forma $\dot{x} = f(t,x)$, con un lado derecho continuo, es equivalente a la siguiente ecuación integral:*

$$x(t) = x(t_0) + \int_{t_0}^{t} f\big(s, x(s)\big)ds \tag{3.11}$$

Si la función $f(t,x)$ es discontinua en t y continua en x las funciones que satisfacen la ecuación (3.11) pueden ser llamadas soluciones de la ecuación diferencial de Caratheodory $\dot{x} = f(t,x)$, asumiéndose que la integral de (3.11) satisface el concepto de integral de Lebesgue. Asimismo, se asume que la función $f(t,x)$ satisface las siguientes condiciones de Caratheodory en el dominio D del espacio (t,x):

- *La función $f(t,x)$ se encuentra definida, siendo continua en x para casi todos los valores de t.*
- *La función $f(t,x)$ debe ser medible en t para cada x.*
- *$|f(t,x)| \leq m(t)$, siendo la función $m(t)$ sumable en cada intervalo finito si t no es acotada en el dominio D.*

Lema 3.9 (Función compuesta sumable [52]): Si se deja que la función $f(t,x)$ *satisfaga las condiciones de Caratheodory y que la función $x(t)$ en el intervalo $a \leq t \leq b$ sea medible, entonces la función compuesta $f(t,x(t))$ es sumable.*

Teorema 3.3 (Condiciones de Caratheodory [52]): *Sea una función $f(t,x)$ que cumple las condiciones de Caratheodory para $t_0 \leq t \leq t_0 + a$ y $|x - x_0| \leq b$. En un intervalo cerrado $[t_0, t_0 + d]$, donde $d > 0$, existe una solución a la ecuación diferencial $\dot{x} = f(x,t)$, donde $x(t_0) = x_0$. Asimismo, se puede tomar un valor arbitrario d que satisface las siguientes desigualdades:*

$$0 < d \leq a \tag{3.12a}$$

$$\varphi(t_0 + d) \leq b \tag{3.12b}$$

$$\varphi(t) \equiv \int_{t_0}^{t} m(s)ds \tag{3.12c}$$

Demostración del Teorema 3.3 (Condiciones de Caratheodory [52]): Para cualquier entero $k \geq 1$ se toma $h = \frac{d}{k}$. En los intervalos $t_0 + ih \leq t \leq t_0 + (i+1)h$, para $i = 0,1,\dots,k-1$, se construye iterativamente una solución asumiendo $x_k(t) = x_0$ para $t \leq t_0$:

$$x_k(t) = x_0 + \int_{t_0}^{t} f\big(s, x_k(s-h)\big)ds, \quad t_0 < t \leq t_0 + d \tag{3.13}$$

Debido a lo expuesto en el teorema (3.3) y el conjunto de ecuaciones (3.12) se obtiene $|x_k(t) - x_0| \leq b$. Asimismo, para valores arbitrarios α y β dentro del intervalo $[t_0, t_0 + d]$ se tiene:

$$|x_k(\beta) - x_k(\alpha)| \leq \left|\int_{\alpha}^{\beta} m(t)dt\right| = |\varphi(\beta) - \varphi(\alpha)| \tag{3.14}$$

La función $\varphi(t)$ es continua en el intervalo cerrado $[t_0, t_0 + d]$ y por consiguiente es uniformemente continua. Por tanto, para cada $\varepsilon > 0$ existe un valor $\delta > 0$ tal que para $|\beta - \alpha| < \delta$ el lado derecho de la ecuación (3.14) es menor a ε y de esta forma las funciones $x_k(t)$, $k = 1,2, \ldots$, son entonces equicontinuas y uniformemente acotadas. Si (3.15) *mediante el teorema de Arzela una subsecuencia convergente y su límite es denotado por $x(t)$, dado que:*

$$|x_k(s-h) - x_k(s)| \leq |x_k(s-h) - x_k(s)| + |x_k(s) - x(s)|$$

y el primer término en el lado derecho de la ecuación es menor a ε para $h = \frac{d}{k} < \delta$, entonces $x_k(s-h)$ tiende a $x(s)$. En virtud a la continuidad de la función $f(t,x)$ en x y el valor estimado $|f(t,x)| \leq m(t)$ se puede pasar al límite bajo el signo integral de (3.13). Por tanto, se concluye que la función límite $x(t)$ satisface la ecuación (3.11) para $x(t_0) = x_0$, siendo una solución de la ecuación diferencial $\dot{x} = f(x,t)$. Demostración finalizada.

Teorema 3.4 (Teorema de Caratheodory [52]): *Para cualquier conjunto cerrado acotado $A \in \mathbb{R}^n$ cualquier punto $x \in co\ A$ puede ser representado de acuerdo a la ecuación (3.9), donde $x_i \in A, i = 1, 2, \ldots, m$, los números α_i satisfacen la condición (3.10) y $k \leq n + 1$.*

Demostración del Teorema 3.4 (Teorema de Caratheodory [52]): *Sea $x \in co\ A$, cumpliéndose lo siguiente:*

$$x = \sum_{i=1}^{N} \alpha_i x_i, \quad [\alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i = 1] \tag{3.16}$$

Si se elige, entre todas las representaciones posibles de x que figuran en (3.16), la combinación convexa con el menor valor de N posible, entonces $N > m + 1$. Por consiguiente, el sistema de ecuaciones homogéneas de dimensión $m + 1$ de la forma:

$$\sum_{i=1}^{N} \mu_i x_i = 0 \tag{3.17a}$$

$$\sum_{i=1}^{N} \mu_i = 0 \tag{3.17b}$$

Con N parámetros desconocidos $\mu_1, \ldots, \mu_N$ posee una solución no trivial $\delta_1, \ldots, \delta_N$:

$$\sum_{i=1}^{N} \delta_i x_i = 0 \tag{3.18a}$$

$$\sum_{i=1}^{N} \delta_i = 0\,, (\delta_1, .., \delta_N) \neq 0 \tag{3.18b}$$

Demostración finalizada.

3.2.3.6 Existencia y propiedades de las soluciones

Definición 3.8 (Solución aproximada de una ecuación con discontinuidades [52]): *La función continua $y(t)$ puede ser considerada como una solución aproximada de la ecuación $\dot{x} = f(t,x)$ si es que se cumple lo siguiente en casi todos sus puntos:*

$$|\dot{y}(t) - f(t, z(t))| \leq \delta, \qquad |z(t) - y(t)| \leq \delta \tag{3.19}$$

Donde $z(t)$ es una función arbitraria y el número δ es lo suficientemente pequeño. Si se denota un vecindario δ cerrado del conjunto M como M^{δ}, se puede escribir la ecuación (3.19) de la siguiente manera:

$$\dot{y}(t) \in [f(t, y(t)^{\delta})]^{\delta} \tag{3.20}$$

Si $z(t) - y(t) = p(t)$, la ecuación (3.19) puede ser descrita de la siguiente manera:

$$\dot{y}(t) = f(t, y(t) + p(t)) + q(t), \quad |p(t)| \leq \delta, \quad |q(t)| \leq \delta \tag{3.21}$$

Donde $p(t)$ y $q(t)$ son conocidas como perturbaciones internas y externas, respectivamente. Tomando en consideración los lemas (3.7) y (3.8), es posible reemplazar la ecuación (3.19) por la siguiente forma:

$$\dot{y}(t) \in [co\ f(t, y(t)^{\delta})]^{\delta} \tag{3.22}$$

Definición 3.9 (Solución aproximada de una inclusión diferencial con exactitud δ [52]): *Una función vectorial $y(t)$ es llamada solución aproximada con exactitud δ de una inclusión diferencial $\dot{x} \in F(t,x)$, con una función F semicontinua superior en t,x, si en un intervalo dado la función $y(t)$ es absolutamente continua, cumpliéndose en casi todos sus puntos:*

$$\dot{y}(t) \in F_{\delta}(t, y(t)) \tag{3.23a}$$
$$F_{\delta}(t,y) \equiv [co\ F(t^{\delta}, y^{\delta})]^{\delta} \tag{3.23b}$$

Aquí y abajo $F(t^{\delta}, y^{\delta})$ implica la unión de los conjuntos $F(t_1, y_1)$ para todo $t_1 \in t^{\delta}, y_1 \in y^{\delta}$, esto es, para $|t_1 - t| \leq \delta, |y_1 - y| \leq \delta$.

Definición 3.10 (Condiciones básicas de $F(t,x)$ [52]): *En el dominio G una función de valor establecido $F(t,x)$ satisface las condiciones básicas si para todo $(t,x) \in G$ el conjunto $F(t,x)$ es no vacío, acotado y cerrado, convexo y la función F es semicontinua superior en (t,x).*

Lema 3.10 (Límite de una secuencia uniformemente convergente [52]): *Si $F(t,x)$ satisface las condiciones básicas en un dominio abierto G entonces el límite $x(t)$ de cualquier secuencia uniformemente convergente de soluciones con exactitud δ_k de $x_k(t)$ ($\delta_k \to 0, k = 1,2,\dots$) de la inclusión diferencial $\dot{x} \in F(t,x)$ es una solución de esta inclusión si la función limitante $x(t)$, para $a \leq t \leq b$, se encuentra dentro de G.*

Teorema 3.5 (Solución de una inclusión diferencial [52]): Sea una función $F(t,x)$ *que satisface las condiciones básicas en el dominio G. Por consiguiente, para cualquier punto $(t_0, x_0) \in G$* existirá una solución de la inclusión diferencial $\dot{x} \in F(t,x)$ *para $x(t_0) = x_0$, si el dominio G contiene un cilindro $Z(t_0 \leq t \leq t_0 + a, |x - x_0| \leq b)$ y la solución existe por lo menos en el intervalo:*

$$t_0 \leq t \leq t_0 + d,\ d = \min\left\{a; \frac{b}{m}\right\}, m = sup_z|F(t,x)| \tag{3.24}$$

Demostración del Teorema 3.5 (Solución de una inclusión diferencial [52]): *Si se considera que existen valores $a > 0$ y $b > 0$ tal que $Z \subset Q$ y existe una función F semicontinua superior en un compactum K donde para cada $p \in K$ el conjunto $F(p)$ es acotado, y por consiguiente esta función es acotada en K, entonces $m < \infty$. Para $k = 1,2,\dots$ se toma $h_k = \frac{d}{k}$, $t_{ki} = t_0 + ih_k, i = 0,1,\dots,k$. Si $x_k(t_{k0}) = x_0$ y si para valores $i \geq 0$ los valores $x_k(t_{ki}) = x_i$ se encuentran ya definidos y se cumple:*

$$|x_{ki} - x_0| \leq m|t_{ki} - t_0| \tag{3.25}$$

Por tanto, al tomar cualquier $v_{ki} \in F(t_{ki}, x_{ki})$, se define $x_k(t)$ para $t_{ki} < t \leq t_{k,i+1}$ por medio de la igualdad:

$$x_k(t) = x_{ki} + (t - t_{ki})v_{ki} \tag{3.26}$$

En virtud de (3.25) el par $(t_{ki}, x_{ki}) \in Z$, se tiene que $|v_{ki}| \leq |F(t_{ki}, x_{ki})| \leq m$, y de (3.25) y (3.26) se tiene que:

$$|x_k(t) - x_0| \le m|t - t_0|, \quad t_{ki} < t \le t_{k,i+1} \tag{3.27}$$

Entonces, el valor $x_k(t_{k,i+1}) = x_{k,i+1}$ *es definido y satisface la desigualdad obtenida en (3.25) reemplazando* i *por* $i+1$*. De esta forma,* $x_k(t)$ *es contraído sucesivamente en los intervalos* $[t_{ki}, t_{k,i+1}]$ *para* $i = 0,1,..,k-1$*; dadas las desigualdades (3.24) y (3.27) la función* $x_k(t)$ *se encuentra contenida en* Z*. Asimismo, de (3.20) la función* $x_k(t)$ *es continua y* $|\dot{x}_k(t)| \le m$ *tal que* $t \ne t_{ki}$ *para todo* $i = 1,2,...$*; Entonces la función es absolutamente continua. Dado,*

$$\dot{x}_k(t) = v_{ki} \in F(t_{ki}, x_{ki}) \tag{3.28a}$$
$$0 < t - t_{ki} < h_k \tag{3.28b}$$
$$|x_k(t) - x_{ki}| \le mh_k \tag{3.28c}$$

Entonces, $x_k(t)$ *es una solución con exactitud* δ_k *de la inclusión diferencial* $\dot{x} \in F(t,x)$ *donde:*

$$\delta_k = \max\{h_k; mh_k\} \to 0, \quad k \to \infty \tag{3.29}$$

En virtud de (3.27) y de la estimación $|\dot{x}_k(t)| \le m$*, las funciones* $x_k(t)$ *son uniformemente acotadas y equi continuas. Por medio del teorema de Arzela, se puede escoger dentro de estas funciones una sub secuencia uniformemente convergente. Por medio del lema (3.9), la función límite* $x(t)$ *es una solución de la inclusión diferencial* $\dot{x} \in F(t,x)$*. De* $x_k(t_0) = x_0$ *se tiene que* $x(t_0) = x_0$*. Demostración finalizada.*

Teorema 3.6 (Dominio de las soluciones de una inclusión diferencial [52]): *Sea la función* $F(t,x)$ *que satisface las condiciones básicas en un dominio acotado cerrado D. Entonces, cada solución de la inclusión diferencial* $\dot{x} \in F(t,x)$ *que se encuentra dentro de D se puede continuar en ambos lados hasta el límite del dominio D.*

Demostración del Teorema 3.6 (Dominio de las soluciones de una inclusión diferencial [53]): *Se considera la solución* $x(t)$ *que pasa por el punto* $p_0\ (t_0, x_0)$ *dentro del dominio* D*. Si* $\varepsilon_1 > 0$ *no es mayor que la mitad de la distancia* $\rho(p_0, \Gamma)$*, medida del punto* p_0 *al límite* Γ*. Asimismo, si* $c \le t \le d$ *se encuentra en el dominio D y dado que la función* $\varphi(t)$ *en (3.12c), para* $m(s) = const = m$*, es uniformemente continua en* $[c,d]$*, existe un* $\delta_1 > 0, \delta_1 \le \varepsilon_1$ *tal que para cualquier* α, β *existente en* $[c,d]$ *satisfaciendo la desigualdad* $|\beta - \alpha| < \delta_1$ *se tiene* $|\varphi(\beta) - \varphi(\alpha)| < \varepsilon_1$*. Entonces, el cilindro* $|t - t_0| \le \delta_1$, $|x - x_0| \le \varepsilon_1$ *está contenido en* D*. En virtud del teorema (3.5), la solución* $x(t)$ *existe al menos en el intervalo* $|t - t_0| \le \delta_1$*. Si la distancia del punto* $(t_0 + \delta_1, x(t_0 + \delta_1))$ *a* Γ *no es menor que* $2\varepsilon_1$*, la solución puede ser continuada hasta un intervalo de longitud* δ_1*, y así sucesivamente, hasta que alcance el punto* $p_1(t_1, x_1)$ *tal que* $\rho(p_1, \Gamma) < 2\varepsilon_1$*. Asumiendo* $\varepsilon_i \to 0, i = 1,2,...$*,se continua con la solución secuencialmente hasta el punto* $p_i(t_i, x_i)$ *tal que la secuencia* $t_1 < t_2 < \cdots$, $\rho(p_i, \Gamma) \to 0$ $(i \to \infty)$ *converge a un valor* t^**. Asimismo, si se considera que en un intervalo* $c \le t \le d$ *todas las soluciones de la ecuación de Caratheodory son equicontinuas y se considera el criterio de Cauchy, se observa que existe un límite* $\lim x(t) = x^*$ *para* $t \to t^*$*, donde* $(t^*, x^*) \in \Gamma$*. Asumiendo que* $x(t^*) = x^*$*, se obtiene una solución que alcanza el límite* Γ *en el punto* (t^*, x^*)*. Por último, se continua con la solución por la izquierda de la misma forma. Demostración finalizada.*

3.2.4 Sistemas que dependen linealmente del vector de control

Si se plantea el sistema dinámico discontinuo que figura en (3.2a) de la siguiente forma:

$$\dot{x} = f(x,t) + B(x,t)u(x,t) \tag{3.30}$$

donde x y f son vectores columna de dimensión n, $B(x,t)$ es una matriz de $n\ x\ m$ y u es el vector de control de dimensión m con componentes que cuentan con discontinuidades en una superficie $s_i(x) = 0$ apropiada. El vector de control u debe cumplir con las ecuaciones que figuran en (3.2b). Cabe resaltar que las funciones continuas $f(x,t)$ y $B(x,t)$ deben satisfacer la condición de Lipschitz afuera de la superficie de discontinuidad, según lo detallado por Utkin [50].

Sea el vector de superficies $s = (s_1, \dots, s_m)$ de dimensión m, el valor del vector de control equivalente u_{eq} puede ser hallado al igualar el vector de superficies $s = 0$:

$$s = Gf + GBu_{eq} = 0 \tag{3.31}$$

Donde G es una matriz de dimensión $m\,x\,n$ que tiene por filas a los vectores gradientes de las funciones $s_i(x)$. Si el determinante de la matriz GB cumple $\det(GB) \neq 0$ para cualquier x y t, se tiene:

$$u_{eq} = -(GB)^{-1}Gf \tag{3.32}$$

Reemplazando $u = u_{eq}$ en $\dot{x} = f(x,t) + B(x,t)u$ se obtiene:

$$\dot{x} = f - B(GB)^{-1}Gf \tag{3.33}$$

donde (3.33) es denominada la ecuación de modos deslizantes ideal [50]. Cabe resaltar que una de las ventajas que posee un sistema de estructura variable por modos deslizantes es que se puede efectuar el control de un sistema de orden n por medio del diseño de un sistema de control de orden reducido, de dimensión $n - m$. Esto es, el método de control equivalente permite que $\dot{s} \equiv 0$ y dadas las condiciones iniciales planteadas todas las trayectorias del sistema permanecen en el colector situado en la intersección de todas las superficies deslizantes de dimensión $n - m$.

Al introducir no idealidades al modelo se obtiene una ecuación de modos deslizantes real, dado que las trayectorias de las variables de estado del sistema tienden de forma finita a un área localizada en la Δ–vecindad del colector deslizante $s(x) = 0$. Esto puede ser representado de la siguiente forma:

$$\|s(x)\| \leq \Delta, \|s\| = \sqrt{\sum_{i=1}^{m} {s_i}^2} \tag{3.34}$$

La distancia de cualquier punto de la Δ–vecindad al colector deslizante $s(x) = 0$ es denominada $r(s,x)$, siendo estimada mediante la desigualdad $r(s,x) \leq P\Delta$, donde P es un número positivo. Según lo explicado por Utkin [50], bajo este marco conceptual, la ecuación (3.30) puede ser representada por una ecuación que incorpora las no idealidades del vector de control $u(x,t)$ de la siguiente forma:

$$\dot{x} = f(x,t) + B(x,t)\tilde{u}(x,t) \tag{3.35}$$

Teorema 3.7 (Cantidad positiva H estabilizante [50]): *Si se cumple que:*

- *En el intervalo $[0,T]$ cualquier solución $x(t)$ de la ecuación (3.35) es tal que la trayectoria de las variables de estado se encuentra en la Δ–vecindad del colector $s(x) = 0$ o la desigualdad (3.34) es válida.*
- *Para el lado derecho de la ecuación (3.33), obtenida por medio del método de control equivalente, existe una constante de Lipschitz denominada L.*
- *Las derivadas parciales de la función $B(x,t)(G(x)B(x,t))^{-1}$ respecto a todos los argumentos existen y son acotados en cualquier dominio acotado.*
- *Siendo la función $f(x,t) + B(x,t)\tilde{u}(x,t)$ el lado derecho de la ecuación de modos deslizantes real (3.35), existen cantidades positivas M y N tal que:*

$$\|f(x,t) + B(x,t)\tilde{u}(x,t)\| \leq M + N\|x\| \tag{3.36}$$

Entonces para cualquier par de soluciones de las ecuaciones (3.33) y (3.35) bajo las condiciones iniciales $\|x(0) - x^*(0)\| \leq P\Delta$ existe una cantida positiva H tal que:

$$\|x(t) - x^*(t)\| \leq H\Delta \; para \; t \in [0, T] \quad (3.37)$$

El teorema anterior muestra que siempre existirá una diferencia entre los modos deslizantes ideal y el real, siendo evidente que la derivada del colector deslizante $\dot{s}(x)$ para los modos deslizantes real será distinta de cero debido a las no idealidades que afectan al sistema. Por tanto, si para el control equivalente se cumple la ecuación (3.32) y el colector deslizante $\dot{s}(x) \neq 0$, el vector de control con no idealidades $\tilde{u}(x,t)$ será igual a:

$$\tilde{u}(x,t) = -(GB)^{-1}Gf + (GB)^{-1}\dot{s}(x) \quad (3.38)$$

Y por consiguiente la ecuación de modos deslizantes real queda representada de la siguiente forma:

$$\dot{x} = f(x,t) - B(x,t)[G(x)B(x,t)]^{-1}G(x)f(x,t) + B(x,t)[G(x)B(x,t)]^{-1}\dot{s}(x) \quad (3.39)$$

3.2.5 Teoría de estabilidad de Lyapunov

Si se considera el sistema invariante en el tiempo $\dot{x} = f(x)$ con condición inicial $x(t_0) = x_0$, donde $f: D \rightarrow \mathbb{R}^n$ es localmente continua desde el sentido de Lipschitz en D con $\sigma \in D \subseteq \mathbb{R}^n$ sujeto al origen localmente, una solución única existe. Asimismo, se debe asumir que $f(0) = 0$ sujeto a que el origen es un punto de equilibrio. Si $f(\bar{x}) = 0$ con $\bar{x} \neq \sigma$, se utiliza $\tilde{x} = x - \bar{x}$ y de esa manera $\dot{\tilde{x}} = f(x - \bar{x}) \equiv \tilde{f}(\tilde{x})$. Entonces, $\tilde{x} = 0$ es un punto de equilibrio respecto a $\dot{\tilde{x}} = \tilde{f}(\tilde{x})$ [54].

Definición 3.11 (Estabilidad de Lyapunov [54]): *Sea $\dot{x} = f(x)$, $x(t_0) = x_0$, con $f(0) = 0$ sujeto a que $\bar{x} = 0$ sea el equilibrio. Entonces, $\bar{x}$ puede ser denominado:*

1) *Estable: Si para cualquier $x_0 \in \mathbb{R}^n$, $\epsilon > 0$ y $t_0 \geq 0$ existe $\delta = \delta(\varepsilon)$ sujeto a $\|x_0\| < \delta$. Por consiguiente, $\|x(t)\| < \varepsilon$ tal que $t \geq t_0 \geq 0$.*
2) *Inestable: Si no cumple con la condición de "estable".*
3) *Asintóticamente Estable: Si cumple con la de "estable" y en adición existe $\gamma > 0$ tal que $\|x_0\| < \gamma$. Por consiguiente, $\lim_{t\to\infty} x(t) = 0$.*

Cabe resaltar que por lo general se cumple $\gamma \leq \delta \leq \varepsilon$.

Teorema 3.8 (Método directo de Lyapunov [54]): *Sea $\bar{x} = 0$ un punto de equilibrio del sistema $\dot{x} = f(x)$, $f: D \rightarrow \mathbb{R}^n$, donde D es un vecindario abierto de $\bar{x}$. Si existe una función $V = V(x), V: D \rightarrow \mathbb{R}^n$, sujeto a:*

1) $V(x) > 0$ *tal que* $x \in D - \{0\}, V(0) = 0$[10]
2) $\dot{V}(x) = \frac{\partial V}{\partial X} f(x) \leq 0$ *tal que* $x \in D$[11]

Entonces, $\bar{x}$ es localmente estable. Si en vez de la condición número 2) es válida la condición más sólida:

2a) $\dot{V}(x) = \frac{\partial V}{\partial X} f(x) < 0$ *tal que* $x \in D - \{0\}$[12]

Entonces $\bar{x}$ es localmente asintóticamente estable. Si $D = \mathbb{R}^n$ y adicionalmente la función V satisface el concepto de ilimitación radial:

[10] Si $V(x)$ satisface esta condición se le puede llamar "Función candidata de Lyapunov".
[11] Si $V(x)$, en adición a la condición 1), satisface la condición 2) se le puede llamar "Función de Lyapunov débil".
[12] Si V(x), en adición a la condición 1), satisface la condición 2a) se le puede llamar "Función de Lyapunov estricta".

3) $\lim_{\|x\|\to\infty} V(x) \to \infty, \|x\| \to \infty$[13]

Entonces, $\bar{x}$ es globalmente asintóticamente estable

3.2.6 Segundo Método de Lyapunov para determinar el dominio de un modo deslizante

El problema de hallar condiciones suficientes para que un dominio en las intersecciones de superficies deslizantes o sobre una superficie deslizante sea considerado un "dominio deslizante" se traduce al conocido problema de hallar una solución estabilizante para un sistema no lineal; las discontinuidades del lado derecho de la ecuación diferencial (3.2a) obligan a establecer algún tipo de método que demuestre la estabilidad del sistema.

Teorema 3.9 (Existencia de un dominio deslizante S en la intersección de superficies de discontinuidad [50]): *Para que el dominio $S(x,t)$, ubicado en la intersección de las superficies de discontinuidad del sistema (3.2), sea considerado un dominio deslizante es suficiente que para todos los valores de x que pertenecen a este dominio exista, en un área Ω del subespacio $s_1, \dots, s_m$ que contenga el origen, una función de Lyapunov $V(s,x,t)$ continuamente diferenciable respecto a todos sus argumentos tal que las siguientes condiciones se cumplan:*

1) *La función debe ser positiva definida respecto a s. Esto es, $V(s,x,t) > 0$ para $s \neq 0$ y valores arbitrarios de x y t y $V(0,x,t) \equiv 0$. Asimismo, en la esfera $\|s\| \leq R$ para todo x que este en la región y cualquier t las relaciones $\inf_{\|s\|=R} v = h_R$, $\sup_{\|s\|=R} v = H_R, R \neq 0$, donde h_R y H_R son positivas dependiendo solo del valor de R.*
2) *La derivada total en el tiempo de la función de Lyapunov $V(s,x,t)$:*

$$\dot{V}(s,x,t) = \frac{\partial V}{\partial s}\dot{s} + \frac{\partial V}{\partial x}\dot{x} + \frac{\partial V}{\partial t} \tag{3.40}$$

$$\dot{V}(s,x,t) = \frac{\partial V}{\partial s}(Gf + GBu) + \frac{\partial V}{\partial x}(f + Bu) + \frac{\partial V}{\partial t}$$

$$\frac{\partial V}{\partial s} = \left(\frac{\partial V}{\partial s_1}, \frac{\partial V}{\partial s_2}, \dots, \frac{\partial V}{\partial s_m}\right)$$

$$\frac{\partial V}{\partial x} = \left(\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \dots, \frac{\partial V}{\partial x_m}\right)$$

Donde (3.40) debe ser negativa en todas partes salvo sobre las superficies de discontinuidad, dado que la función no se encuentra definida en esas áreas. Asimismo, esta función en toda la superficie de la esfera $\|s\| = R$, a excepción de los puntos de discontinuidad, cumple lo siguiente:

$$\sup_{\|s\|=R} \dot{v} = -m_R \tag{3.41a}$$

$$m_R = const > 0 \tag{3.41b}$$

Si (3.41a) y (3.41b) son verdaderas entonces se puede decir que la función $\dot{V}(s,x,t)$ es positiva definida.

Demostración del Teorema 3.9 (Existencia de un dominio deslizante S en la intersección de superficies de discontinuidad [49]): *La ecuación (3.40a) puede ser representada de la siguiente forma:*

$$\dot{V}(s,x,t) = G_0(s,x,t) + \sum_{i=1}^{m} G_i(s,x,t)u_i \tag{3.42}$$

Donde $G_0, G_1, \dots, G_m$ son funciones continuas respecto a todos los argumentos y dependientes de $f, B, G, \frac{\partial V}{\partial s}, \frac{\partial V}{\partial x}$ y $\frac{\partial V}{\partial t}$. Para demostrar el teorema se debe efectuar lo siguiente:

[13] La condición 3) garantiza que los conjuntos de nivel sean acotados.

1) *Tomar un punto arbitrario x_0 que se encuentre sobre las superficies $s_i = 0$ para $i = 1,2,\dots,k$ tal que k < m y sobre la esfera $\|s\| < R$, y la secuencia i = 1,2,…m del segundo término de la ecuación (3.42) es dividida en dos subsecuencias de tal manera que la primera subsecuencia es i= 1,2,..,k y la segunda es i=k+1,…,m, las funciones de control u_i pueden tomar los valores límite ${u_i}^+$ y ${u_i}^-$ en la primera subsecuencia[14]. Por tanto, la ecuación (3.42) queda representada de la siguiente manera:*

$$\dot{V}(x_0) = G_0(s, x_0, t) + \sum_{i=1}^{k} G_i(s, x_0, t){u_i}^{\pm} + \sum_{i=k+1}^{m} G_i(s, x_0, t){u_i}^{15} \qquad (3.43)$$

2) *Considerar a la función $\dot{V}(s, x, t)$ a lo largo de trayectorias de las variables del estado del sistema con no idealidades que escapan de la Δ_0-vecindad de las superficies de discontinuidad, y asumir que el control real es $\tilde{u}_i = u_i$ si se cumple $|s_i| \geq \Delta_0$, entonces se puede considerar que los valores de $\dot{V}(s, x, t)$ para un sistema con no idealidades y un sistema ideal son iguales.*

3) *La esfera $\|s\| < R$, fuera de los puntos de discontinuidad que puedan existir, el valor de la función $\dot{V}(s, x, t)$ no excede m_R debido a la condición número dos del teorema. Por otro lado, en la Δ_0-vecindad de las intersecciones de las superficies $s_i = 0$ para valores de i = 1,2,…,k la función $\dot{V}(s, x, t)$ puede ser representada de la siguiente forma:*

$$\dot{V}(x) = \dot{V}_0 + O(\Delta_0) \qquad (3.44)$$

Donde $\dot{V}_0 = G_0(s, x_0, t) + \sum_{i=1}^{k} G_i(s, x_0, t)\tilde{u}_i + \sum_{i=k+1}^{m} G_i(s, x_0, t)u_i$, x_0 es el punto de intersección de las superficies $s_i = 0$, Δ_0 es igual a la distancia entre los puntos x y x_0 y $O(\Delta_0)$ es un valor infinitesimal respecto a Δ_0.

4) *Se halla el valor de $\dot{V}(x_0)$ en cualquier instante de tiempo y su máximo valor en el punto x_0 por medio de la desigualdad $\dot{V}(x_0) \leq -m_R$ y al elegir valores apropiados de los límites ${u_i}^{\pm}$, sujeto al signo de la función G_i , respectivamente.*

5) *Se deduce que el valor máximo de la función $\dot{V}_0$ en relación al control $\tilde{u}_i$ coincide con el valor máximo de la función $\dot{V}(x_0)$ en relación al control u_i, debido a que el valor de $\tilde{u}_i$ se encuentra necesariamente entre los límites de control ${u_i}^+ y {u_i}^-$. Consecuentemente, $\dot{V}_0$ es también más pequeño que el valor $-m_R$ y si $\dot{V}_0 = G_0(s, x_0, t) + \sum_{i=1}^{k} G_i(s, x_0, t)\tilde{u}_i + \sum_{i=k+1}^{m} G_i(s, x_0, t)u_i$ para un valor fijo de R, es posible seleccionar siempre un valor de Δ_0 tal que en la esfera de radio R la función $\dot{V}(x)$ siempre sea negativa. Demostración finalizada.*

Teorema 3.10 (Dominio deslizante por medio de formas cuadráticas [50]): *Sea la función,*

$$\dot{s} = G(x)f(x, t) + G(x)B(x, t)u \qquad (3.45)$$

Que describe el movimiento deslizante de un sistema discontinuo, donde G es una matriz de dimensión $m\ x\ n$ que tiene por filas las gradientes de las superficies discontinuas $s_1(x), \dots, s_m(x)$ y $x, f(x, t), u\ y\ la\ matriz\ B(x, t)$ determinan el movimiento del sistema; el vector de control u es representado de la siguiente forma:

$$u = u^0 + Usign(s) \qquad (3.46)$$

[14] Cabe resaltar que a pesar de que la función $\dot{V}(s, x, t)$ no se encuentra definida en el punto x_0, para efectos de demostración del teorema se realiza esto.

[15] Si cumplen las condiciones (3.41) para la función $\dot{V}(x_0)$ y se cumple que las funciones $G_i, {u_i}^+ y\ {u_i}^-$ sean continuas, se puede estimar a $\dot{V}(x_0)$ por medio de la desigualdad $\dot{V}(x_0) \leq -m_R$ para cualquier combinación de valores u_i.

Donde $u^0 = \frac{1}{2}(u^+ + u^-)$, u^+ *y* u^- *son los límites máximos del vector de control* u *para* $i = 1, \dots, m$, U *es una matriz diagonal de dimensión* m *con elementos* $U_i = \frac{1}{2}(u_i^{\ +} - u_i^{-})$ *y* $sign(s)$ *es un vector de dimensión* m *con elementos* $sign(s_i)$[16]. *Al respecto, la función (3.45) puede ser descrita de la siguiente forma:*

$$\dot{s} = D(x,t)sign(s) + d(x,t) \tag{3.47}$$

Donde $D(x,t) = G(x)B(x,t)u$ *y* $d(x,t) = G(x)f(x,t) + G(x)B(x,t)u^0$. *Por tanto, para que un dominio* S *localizado en la intersección de superficies de discontinuidad* s_i *en cualquier instante* t *sea considerado un dominio deslizante, deberán existir las matrices* $W(x,t)$ *y* $L(x,t)$ *y el vector* $l(x,t)$ *tal que* $L = -W(x,t)D(x,t)$ *y* $l = W(x,t)d(x,t)$ *y deberán cumplirse las siguientes condiciones:*

1) *La matriz* W *es simétrica (*$W = W^T$*), siendo todas sus menores principales son positivas y tienen límites superior e inferior.*
2) *Los elementos de la matriz* L *y el vector* l *deben cumplir la condición:*

$$\inf_{x \in S, t \in [0,\infty]} \left(l_{kk}(x,W,t) - \sum_{\substack{i=1 \\ i \neq k}}^{m} |l_{ki}(x,W,t)| - |l_k(x,W,t)|\right) > \Delta l_k > 0 \tag{3.48}$$

donde $\Delta l_k = constante$ *para* $k = 1,2,\dots m$.

3) *La norma de la derivada en el tiempo de* W *es acotada por un número* M.

Demostración del Teorema 3.10 (Dominio deslizante por medio de formas cuadráticas [50]): *De la condición número uno, para la forma cuadrática* $V = \frac{1}{2}s^T W(x,t)s$ *las relaciones* $\inf_{\|s\|=R} v = h_R$, $\sup_{\|s\|=R} v = H_R, R \neq 0$ *del teorema 3.8 (Existencia de un dominio deslizante* S *en la intersección de superficies de discontinuidad) cumplen. Asimismo, la segunda condición implica que la función* $V_0 = -s^T Lsign(s) + s^T l$ *sea negativa definida en la superficie de cualquier esfera de radio* $\|s\|$ *y su límite superior igual a* $\sup_{\|s\| = const} V_0 = -\|s\|\Delta l_r$. *Por tanto, si se computa la derivada de la función* V, *tomando en consideración que* $L = -WD$ *y* $l = -Wd$, *se tiene:*

$$\dot{V} = -V_0 + s^T \dot{W} s \tag{3.49}$$

En cumplimiento a la tercera condición del teorema y la relación $\sup_{\|s\| = const} V_0 = -\|s\|\Delta l_r$, *la estimación del límite superior de la función (3.49) en la esfera de radio* $\|s\|$ *es:*

$$\sup_{\|s\|=const} \dot{v} \leq -\|s\|\Delta l_r + \|s\|^2 M \tag{3.50}$$

Dado que el primer y segundo término de (3.50) son proporcionales a $\|s\|$ *y* $\|s\|^2$ *siempre existirá una vecindad del origen de las coordenadas dentro de la cual, en cualquier esfera, el límite inferior de la función* V *es infinita. Por tanto, la condición (3.41) del teorema (3.8) se cumple y por consiguiente el dominio* S *puede ser considerado un dominio deslizante. Demostración finalizada.*

3.2.7 Principio de Invariancia para sistemas discontinuos por el método de Velichenko

Dado que los valores de las variables que implementan el colector o superficie deslizante $s(t)$ son cercanas a cero y la salida discontinua promedio de $u(t)$ toma valores finitos, el sistema de estructura variable por modos deslizantes planteado en el apartado anterior permite implementar un sistema de alta ganancia, teóricamente de ganancia infinita, siendo capaz de rechazar

16 La función $sign(s_i)$ está definida en cualquier punto a excepción de $s_i = 0$ y es igual a +1 cuando $s_i > 0$ y -1 para $s_i < 0$.

perturbaciones y otras incertidumbres por medio de acciones de control finitas [51]; esta propiedad es conocida como la propiedad de invariancia.

Se conoce que para sistemas continuos invariantes en el tiempo (LTI) se utiliza el principio de invariancia de La Salle [55] con la finalidad de comprobar su estabilidad. Sin embargo, este principio fue adaptado por Byrnes y Martin [56] para sistemas no lineales estableciendo un principio de invariancia integral y posteriormente por Ryan et al. [57] fue extendido para inclusiones diferenciales. Asimismo, tal y como se describe en [58] existen diversos autores que desarrollan extensiones del principio de invariancia de La Salle para ecuaciones diferenciales con lado derecho discontinuo para soluciones de Fillipov y otros para soluciones de Caratheodory, y otros autores tales como Velichenko [59] que desarrollan el principio de invariancia para sistemas no lineales desde el enfoque variacional de Rozonoer [60].

Teorema 3.11 (Principio de invariancia de Velichenko [59]): *Sea el sistema de ecuaciones que describe a un sistema de dimensión n:*

$$\dot{x}(t) = f(x, u, t) \tag{3.51}$$

donde x es el vector de estados y u es un vector de perturbaciones externas. Se establece el funcional,

$$J(x, u) = \Phi[x(T), T] \tag{3.52}$$

como el criterio de control, representado por la función $\Phi(x, t)$ (3.53) dependiente de las coordenadas del sistema (x(t), t) en el instante de muestreo T, este último definido por la condición de que la trayectoria ($x(t)$,t) del sistema alcanza la hiper superficie M definida por,

$$M(x, t) = 0 \tag{3.54}$$

Asimismo, las perturbaciones $u(t)$ deben ser restringidas tal que $u(t)$ no ocasione que los puntos $\{x, u, t\}$ abandonen la región discontinua G donde los elementos discontinuos del lado derecho de (3.51) existen. Si se establece la región en el espacio A de dimensión $n + 1$, con un vector $u(t)$ de perturbaciones permisible que genere trayectorias (x(t), t) que inicien y se encuentren contenidas en A, cada punto $\{x(t), u(t), t\}$ pertenece a la región discontinua G y se asume que (3.51), (3.53) y (3.54) y sus primeras y segundas derivadas con respecto a todos sus argumentos son funciones continuas, se dice que el sistema (3.51) es Φ-invariante en el conjunto M respecto a u(t) si en sus trayectorias, que inician en los puntos {x, t} ε A, el valor del funcional (3.52) no depende de la perturbación $u(t)$.

Definición 3.12 (Campo de apoyo [59]): *Sea la función vectorial $u(t)$ elegida como el control, siendo su valor conocido en cualquier instante de tiempo a requerimiento, y una función de soporte de control $\tilde{u}(t)$ con todas sus trayectorias $\tilde{x}(t), t$, iniciando en los puntos $\{x, \tau\}$ ε A y correspondiendo al control $u(t) = \tilde{u}(t)$. La trayectoria $\tilde{x}(t), t$ alcanza M en el instante $t = \tilde{T}$, definiéndose a lo largo de esta una función vectorial $\tilde{\rho}(t)$ por medio de la ecuación:*

$$\dot{\tilde{\rho}}(t) = -\nabla_x H(x, \tilde{\rho}, \tilde{u}, t) \tag{3.55a}$$
$$x(t) \equiv \tilde{x}(t) \tag{3.55b}$$
$$H \equiv (\rho, f(x, u, t)) \tag{3.55c}$$

Con la condición de frontera:

$$\tilde{\rho}(\tilde{T}) = \left[-\nabla_x \Phi(x, t) + \left(\frac{d\Phi(x, t)}{dt} / \frac{dM(x, t)}{dt}\right) \nabla_x M(x, t) \right]_{t=\tilde{T}, x=\tilde{x}(\tilde{T})} \tag{3.56}$$

De lo anteriormente expuesto, el grupo de ecuaciones (3.55) coloca al vector $\tilde{\rho} = \tilde{\rho}(x, \tau)$ en correspondencia con cada punto {x, τ } ε A, por lo que se dice que el campo vectorial $\tilde{\rho}(x, \tau)$ es un campo de apoyo.

Definición 3.13 (Función de apoyo [59]): *La trayectoria $\tilde{x}(t), t$ pasando por el punto $\{x, \tau\}$ coloca al valor del funcional (3.4) en correspondencia única con este punto, lo que hace que un valor del funcional (3.52) sea una función del punto $\{x, \tau\}$. Entonces, se dice que:*

$$\tilde{V}(x, \tau) \equiv \Phi[\tilde{x}(\tilde{T}), \tilde{T}] \tag{3.57}$$

es una función de apoyo; por medio de esta función de apoyo se define la diferencia entre el valor del funcional en la trayectoria $\hat{x}(t), t$ y la trayectoria $\tilde{x}(t), t$, pasando ambas trayectorias por el punto $\{x_o, t_o\} \in A$ y correspondiendo a los controles $\hat{u}(t)$ y $\tilde{u}(t)$, de la siguiente manera:

$$\Delta J = \Phi[\hat{x}(\hat{T}), \hat{T}] - \Phi[\tilde{x}(\tilde{T}), \tilde{T}] = \tilde{V}[\hat{x}(\hat{T}), \hat{T}] - \tilde{V}[x_o, t_o] = \int_{t_o}^{\hat{T}} \frac{d\tilde{V}[\hat{x}(t), t]}{dt} dt \tag{3.58}$$

Para valores pequeños de dt, el valor de la integral (3.58) respecto del tiempo a lo largo de la trayectoria $\hat{x}(t), t$ es,

$$\Delta J = \int_{t_o}^{\hat{T}} [H(\hat{x}, \tilde{p}(\hat{x}, t), \hat{u}, t)] - [H(\tilde{x}, \tilde{p}(\tilde{x}, t), \tilde{x}, t)]\, dt \tag{3.59}$$

Por tanto, el uso de la integral (3.59) permite establecer la condición necesaria y suficiente de invariancia del sistema en estudio.

Teorema 3.12 (Principio de independencia [59]): *Una condición necesaria y suficiente para que se cumpla el teorema (3.10) es que la función,*

$$H[x, \tilde{p}(x, \tau), u, \tau] \equiv (\tilde{p}(x, \tau), f(x, u, \tau)) \tag{3.60}$$

Correspondiente a la función de apoyo de control $\tilde{u}(t)$, sea independiente de u. Asimismo, si esta condición es válida para una función de apoyo entonces se dice que es válida también para cualquier función de apoyo, dado que los vectores del campo de apoyo y el Hamiltonian $H[x, \tilde{p}(x, \tau), \tilde{u}, \tau] = \tilde{H}(x, \tau)$ presentan la propiedad de invariancia respecto a $\tilde{u}(t)$. Esto significa que los valores de los vectores $\tilde{p}(x, \tau)$ y $\tilde{H}(x, \tau)$ solo dependen del punto x, τ y no dependen de la función de control $\tilde{u}(t)$.

Teorema 3.13 (Principio de independencia [59]: *Para que se cumpla el teorema (3.12) es una condición necesaria y suficiente que la función de apoyo, correspondiente a cualquier función de apoyo de control $\tilde{u}(t)$, satisfaga la siguiente ecuación diferencial parcial:*

$$\frac{\partial \tilde{V}}{\partial \tau} = -(\nabla_x \tilde{V}, \mathrm{f(x, u, t)}) \tag{3.61}$$

Teorema 3.14 (Principio de invariancia respecto a condiciones iniciales [59]: *Para que el sistema (3.51) sea Φ −invariante respecto a las condiciones iniciales en L, es necesario y suficiente que las condiciones del teorema (3.10) se cumplan y que en cualquier punto $\{x, \tau\} \in L$ se cumpla:*

$$(\tilde{P}(x, \tau), L(x, \tau)) = 0 \tag{3.62}$$

A partir de los teoremas estudiados se puede establecer el problema de síntesis de un sistema invariante, el cual se traduce a encontrar un término correctivo que asegure la invariancia del sistema. Por tanto, el problema es formulado por Velichenko [59] de la siguiente forma:

Sea el sistema de control descrito por el siguiente sistema de ecuaciones,

$$\dot{x}(t) = f(x, u, v, t) \tag{3.63}$$

Donde x es la variable de estado, u es considerada una perturbación escalar y v una función escalar de control que representa el término correctivo que la parte variante del sistema. Por tanto, la función escalar de control debe ser elegida como $v = v(x, u, t)$ tal que el sistema (3.63)

sea Φ −invariante en M respecto a u. Asimismo, se define al campo de apoyo $\tilde{p}(x,t)$ correspondiente a $\tilde{u}(t) \equiv 0$ y la siguiente ecuación

$$H[x,\tilde{p}(x,t),u,v,t] = H[x,\tilde{p}(x,t),0,0,t] \tag{3.64}$$

que define a la función v en términos de $x,u\ y\ t$; *para que la función* v sea considerada correctiva, de manera necesaria y suficiente, debe ser una solución de la ecuación (3.64). Cabe resaltar que utilizando la función de apoyo (3.57), la ecuación (3.64) puede ser escrita de la siguiente forma:

$$-(\nabla_x \tilde{V}(x,t), f(x,u,v,t)) = \frac{\partial \tilde{V}(x,t)}{\partial t} = -(\nabla_x \tilde{V}(x,t), f(x,0,0,t)) \tag{3.65a}$$

La solución $\tilde{x}(t) = \tilde{\varphi}(x,\tau,t)$ a la ecuación (3.65a) representa la solución completa del sistema (3.63) para $u \equiv 0$ y $v \equiv 0$, donde la condición inicial es el punto $\{x,\tau\}$. Si se reemplaza $\tilde{x}(t)$ en (3.54) y se resuelve respecto a $\tilde{T}$ se obtiene:

$$\tilde{x}\,(\tilde{T}) = \tilde{\varphi}[x,\tau,\tilde{T}(x,\tau)] = \varphi(x,\tau) \tag{3.65b}$$
$$\tilde{V}(x,\tau) \equiv \Phi[\tilde{x}(\tilde{T}),\tilde{T}] = \Phi[\varphi(x,\tau),\tilde{T}(x,\tau)] \tag{3.65c}$$

Y su condición de controlabilidad respecto a v existe de acuerdo al cumplimiento de la siguiente desigualdad para todos los valores de $x,t,u\ y\ v$:

$$\frac{\partial}{\partial v} H[x,\tilde{p}(x,t),u,v,t] \neq 0 \tag{3.66}$$

De acuerdo a la ecuación (3.58), se determina que la condición necesaria y suficiente para que el sistema sea invariante debe tener la siguiente forma:

$$\Delta J(t) = -\int_{t_o}^{T} ([H(x,\tilde{\rho}(x,t),u,v,t)] - [H(x,\tilde{\rho}(x,t),0,0,t)])\,dt = 0 \tag{3.67}$$

Siendo esta condición válida para cualquier trayectoria $\mathrm{x(t)}, \mathrm{t}$ con punto inicial en $\{\mathrm{x_0}, \mathrm{t_0}\}$. Si se da la restricción de que $v \in W$, *se puede satisfacer la ecuación (3.67) por medio de lo siguiente:*

$$H(x,\tilde{\rho}(x,t),u,v,t) = \sup_{w \in W} H(x,\tilde{\rho}(x,t),u,w,t), \quad para\ \Delta J(t) > 0 \tag{3.68a}$$
$$H(x,\tilde{\rho}(x,t),u,v,t) = \inf_{w \in W} H(x,\tilde{\rho}(x,t),u,w,t), \quad para\ \Delta J(t) < 0 \tag{3.68b}$$

3.3 Criterios de evaluación de desempeño de los sistemas de estructura variable para la observación de estados

Para efectuar la comparación de los observadores a ser diseñados en el presente capítulo, es indispensable establecer criterios relacionados al desempeño de estos ante incertidumbres, ruido, entre otros.

3.3.1 Nivel de robustez de la estimación ($\mathbf{N_\rho}$)

Para la presente tesis se busca evaluar el desempeño del observador en un ambiente que cuente con perturbaciones o no linealidades no contempladas en el desarrollo del modelo del sistema, denominadas ***incertidumbres no estructuradas,*** y variaciones de parámetros de la planta o proceso, denominadas ***incertidumbres paramétricas*** por Zhang et al. [48]. Para tal efecto, el criterio de desempeño de ***robustez de la estimación*** permite identificar si el observador mantiene la exactitud de la estimación a pesar de que el modelo cuente con este tipo de incertidumbres, calificando su desempeño como bajo, medio o alto.

3.3.2 Nivel de sensibilidad al ruido de medición ($\mathbf{N_{ruido}}$)

Busca evaluar el nivel de corrupción que presentan las variables de estado estimadas cuando se proveen mediciones de las variables de estado reales con ruido angular, el cual ha sido modelado

para la presente tesis como una distribución mixta gaussiana (GMM). Por tanto, se calificará la sensibilidad al ruido de medición como baja, media o alta.

3.3.3 Número de variables de estado medibles empleadas ($\mathbf{N_{var}}$)

Se conoce que muchas aplicaciones de control utilizan observadores de variables de estado para poder acceder a las variables de estado del modelo que no son medibles y así poder elaborar una señal apropiada para lograr el objetivo de control. Sin embargo, existen algunos observadores de modos deslizantes que deben contar con las mediciones de las variables de estado de posición y velocidad para poder estimar las variables de estado de aceleración. Asimismo, para poder estimar las variables de estado de sobre aceleración es necesario medir las variables de estado de posición, velocidad y aceleración.

3.3.4 Número de variables de estado estimadas ($\mathbf{N_{est}}$)

Este criterio involucra la cantidad de variables de estado reales del modelo del blanco aéreo que pueden ser estimadas por el observador.

3.3.5 Nivel de castañeo o "chattering" (N_δ)

El castañeo o "chattering" excesivo afecta el desempeño del observador para la aplicación en estudio por dos motivos:

(1) Las estimaciones efectuadas por el observador son suministradas al controlador de posición de antena a fin de reposicionarla a la posición angular estimada del blanco, por lo que si existe excesivo castañeo o "chattering" esto es perjudicial para los servo-sistemas involucrados del sistema de seguimiento automático.

(2) Al implementar el observador de modos deslizantes en su forma discreta, según lo señalado por Sabanovic et al. [51], existen restricciones de implementación de la acción de conmutación de la inyección no lineal v si es que la frecuencia de esta excede el período de muestreo utilizado por el microprocesador.

Por ello, en este criterio de desempeño se evalúa si el observador genera un nivel bajo, medio o alto de castañeo, así como si la señal de control discontinua está siendo muestreada a un tiempo de muestreo por lo menos dos veces mayor que la componente de frecuencia mas alta que posea, según el teorema de Nyquist.

3.3.6 Exactitud de la estimación durante maniobra terminal ($RMSE$)

El observador diseñado debe obtener la mejor exactitud posible durante la maniobra terminal del misil, a fin de permitir que el arma antiaérea afectada pueda lograr la mayor probabilidad de destrucción. Por ello, este criterio de desempeño será calculado por medio del error medio cuadrático (RMSE) entre la variable de estado del sistema x y la variable de estado estimada por el observador $\hat{x}$, aplicando la siguiente formula:

$$RMSE = \frac{1}{nt}\sqrt{\sum_{k=0}^{nt}(x_k - \hat{x}_k)^2}$$

donde x_k y $\hat{x}_k$ son el vector de estados del sistema y el vector de estados estimados del sistema en el instante k de muestreo y nt es la cantidad total de muestras. Asimismo, es necesario establecer límites máximos de errores polares en distancia, acimut y elevación para verificar la precisión de la estimación durante la maniobra terminal del misil. Por ello, se dedujeron estos errores máximos en base al siguiente razonamiento:

- **Error polar en distancia:** Durante la fase de adquisición del blanco aéreo el radar de control de tiro habilita una compuerta de traqueo en distancia de aproximadamente 1,000 yds a fin de centrar el pulso del eco de retorno mediante el uso de un algoritmo denominado Early-Late

Gate. Posterior a la adquisición, el radar reduce esta compuerta de traqueo en distancia a aproximadamente el ancho del pulso de radar. Si se determinó una frecuencia de transmisión en banda X y un ancho de pulso de 0.5 useg, el ancho del pulso de radar en metros será igual a $\mathrm{PW_m} = \frac{\mathrm{PW_{seg}} * \mathrm{c}}{2}$, donde $\mathrm{PW_{seg}}$ es el ancho de pulso en segundos y $\mathrm{c} = 3 * 10^8\ \mathrm{m/s}$ es la velocidad de la luz. Efectuando la operación se obtiene un ancho de pulso de 75 metros. Por tanto, el observador debería mantenerse en un rango de $\pm\, 37.5\ \mathrm{metros}$ respecto a la posición real del blanco, la cual se ve afectada por ruido de medición en un caso real.

- **Errores polares en acimut y elevación:** Cuando el misil se aproxima al objetivo, el sistema de armas del buque antiaéreo podrá hacer uso de artillería antiaérea, para el caso de utilizar un arma de 40mm, a una distancia máxima aproximada de 2400 metros con la finalidad de batir al misil. La carga de combate de las municiones disparadas por el buque antiaéreo usualmente viene pre fragmentada en su interior con esferas de tungsteno con la finalidad de que estas impacten al blanco al efectuar la activación de la espoleta de proximidad por medio del efecto Doppler. Es usual que la distancia de activación de la espoleta sea menor a los 10 mts, a fin de elevar la probabilidad de destrucción del blanco. Si se establece la distancia de activación de la espoleta de proximidad de la munición antiaérea en 10 mts, el máximo error de observación permisible en acimut y elevación deberá ser de 10 mts. Asimismo, a un alcance máximo del arma antiaérea de 2400 mts, 1 mrad equivale a 2.4 mts y 10 mts equivaldrá a 4.1667 mrad. Por tanto, los límites máximos permisibles de traqueo son establecidos en $\pm\, 2\ mrad$.

3.3.7 Tiempo de establecimiento en el colector deslizante ($\mathbf{T_{es}}$)

Mide el tiempo que tarda el observador en establecerse en el colector deslizante, es decir, en llevar el error a cero o cerca de este a una solución compacta. Por ello, se define el criterio de tiempo de establecimiento en el colector deslizante T_s, en segundos.

3.3.8 Tiempo consumido en el cómputo del algoritmo ($\mathbf{T_{cpu}}$)

Se define como la demanda computacional que presenta el observador diseñado en base al tiempo que demora en ejecutar el algoritmo de observación, calculado desde que se reciben las entradas al sistema hasta obtener las salidas de estimación.

3.3.9 Estimación de perturbaciones ($\boldsymbol{\xi_{est}}$)

Permite evaluar si el observador cuenta con la capacidad de estimar o reconstruir las perturbaciones al modelo del blanco aéreo de alta maniobrabilidad.

3.4 Introducción al diseño de observadores de estado de modos deslizantes

A partir de los fundamentos teóricos presentados en el apartado anterior, en esta parte del capítulo se efectúa la introducción al diseño de observadores de estructura variable por modos deslizantes.

Se puede considerar que el primer observador robusto es el propuesto por el Dr. Vadim Utkin, en la década de los sesentas. A continuación, se presenta el diseño de este observador, descrito ampliamente en [50] y [51], el cual sirve como base para el diseño de otros observadores de modos deslizantes más complejos.

Sea el sistema dinámico lineal invariante en el tiempo,

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad (3.69a)$$
$$y(t) = Cx(t) \qquad (3.69b)$$

Donde $A \in \mathbb{R}^{nxn}, B \in \mathbb{R}^{nxm}, C \in \mathbb{R}^{pxm}$ son matrices del sistema, $x(t)$ es el vector de estados del sistema de dimensión $n\,x\,1$, $\dot{x}(t)$ es la derivada del vector de estados del sistema de dimensión $n\,x\,1$, $u(t)$ es el vector de entrada del sistema de dimensión $m\,x\,1$, $y(t)$ es el vector

de mediciones del sistema de dimensión $p\ x\ 1$. Asimismo, de cumplir que el $rango(C) = p$, $p \geq m$ y el par A, C debe ser observable.

El observador de modos deslizantes lineal de Utkin está conformado por un observador lineal de Luenberger y una función vectorial discontinua que lleva de forma finita al error del sistema a cero. Al respecto, el observador de Luenberger es diseñado con la misma estructura que el sistema dinámico original pero agregando una retroalimentación del error, de acuerdo al siguiente detalle:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L(C\hat{x}(t) - y(t)) \tag{3.70}$$

Donde L es la ganancia de Luenberger de dimensión $n\ x\ p$, la cual puede ser hallada por medio de la solución de la ecuación algebraica de Riccati, $\dot{\hat{x}}$ es la derivada del vector de estados estimados y $\hat{x}$ es el vector de estados estimados. Asimismo, la dinámica del error está definida por la ecuación $\dot{e}_x(t) = (A + LC)e_x(t)$, siendo los autovalores de la matriz $(A + LC)$ dependientes de la apropiada selección de la matriz *L*. Cabe resaltar que el diseño de autovalores apropiados para la matriz *L* se puede efectuar al resolver la siguiente ecuación algebraica de Riccati:

$$PA - PBR^{-1}B^TP + A^TP + Q = 0 \tag{3.71a}$$
$$L = R^{-1}B^TP \tag{3.71b}$$

Donde P es una matriz simétrica positiva de dimensión $n\ x\ n$ y Q es una matriz de pesos de dimensión $n\ x\ n$, elegidos a criterio del diseñador.

De acuerdo a la teoría de diseño de los observadores de modos deslizantes, el sistema en estudio debe ser descompuesto en su forma canónica, a fin de separar las variables de estado medibles de las no medibles. Para tal efecto, es necesario realizar una transformación de coordenadas por medio de la matriz de transformación $T_c = \begin{bmatrix} N_c{}^T \\ C \end{bmatrix}$, donde $N_c \in \mathbb{R}^{n\ x\ (n-p)}$ abarca el espacio nulo de la matriz C; efectuando la transformación se obtienen las matrices del sistema en espacio de estados en el nuevo sistema de coordenadas:

$$\bar{A} = T_cAT_c{}^{-1} \tag{3.72a}$$
$$\bar{B} = T_cB \tag{3.72b}$$
$$\bar{C} = CT_c{}^{-1} \tag{3.73b}$$

donde $\bar{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \bar{B} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \bar{C} = [0 \quad I]$ y $I \in \mathbb{R}^{p\ x\ p}$ es una matriz de identidad.

De acuerdo a las transformaciones que figuran en (3.72), el sistema en espacio de estados de forma canónica puede ser representado de la siguiente forma:

$$\dot{x}_1(t) = A_{11}x_1(t) + A_{12}y(t) + B_1u(t) \tag{3.73a}$$
$$\dot{y}(t) = A_{21}x_1(t) + A_{22}y(t) + B_2u(t) \tag{3.73b}$$

Donde $x_1(t)$ es el vector de estados no medibles y $y(t)$ es el vector de estados medibles. A partir de la representación del sistema en forma canónica se diseña el observador de modos deslizantes de Utkin con la siguiente estructura:

$$\dot{\hat{x}}_1(t) = A_{11}\hat{x}_1(t) + A_{12}\hat{y}(t) + B_1u(t) + Lv \tag{3.74a}$$
$$\dot{\hat{y}}(t) = A_{21}\hat{x}_1(t) + A_{22}\hat{y}(t) + B_2u(t) - v \tag{3.74b}$$

Donde $\hat{x}_1(t)$ y $\hat{y}(t)$ son estimaciones de los vectores de estado medibles y no medibles del sistema, respectivamente, L es la matriz de Luenberger de dimensión $(n - p)\ x\ p$ y v es la función de control discontinua dlos modos deslizantes que llevará el error de las variables de estado idealmente a cero. Por otro lado, si se define el error de estados no medibles como $e_1 = \hat{x}_1 - x_1$ y el error de estados medibles como $e_y = \hat{y} - y$, el sistema de error según [51] está dado de la siguiente forma:

$$\dot{e}_1(t) = A_{11}e_1(t) + A_{12}e_y(t) + Lv \quad (3.75a)$$
$$\dot{e}_y(t) = A_{21}e_1(t) + A_{22}e_y(t) - v \quad (3.75b)$$

Dado que el par matricial A, C es observable, entonces el par A_{11}, A_{21} también es observable. Por ello, la matriz de observación L debe ser diseñada de tal forma que los autovalores de la matriz $A_{11} + LA_{21}$ cuenten con parte real negativa. Sin embargo, si se introduce una perturbación $\xi(t, y, u)$ de dimensión $h\ x\ 1$ a través de la matriz de distribución D al sistema lineal planteado en (3.69),

$$\dot{x}(t) = Ax(t) + Bu(t) + D\xi(t, y, u) \quad (3.76a)$$
$$y(t) = Cx(t) \quad (3.76b)$$

El concepto de rechazar la perturbación, aprovechando las propiedades de robustez de los observadores de modos deslizantes, se establece a partir del diseño de una ganancia de observación que guarde relación con la matriz de distribución de perturbaciones D [50] de la siguiente forma:

$$D = \begin{bmatrix} L \\ -I \end{bmatrix} X \quad (3.77)$$

Donde X es de dimensión p x h. Asimismo, el sistema de error planteado en (3.75), a partir del considerando que figura en (3.77), se da de la siguiente forma:

$$\dot{e}_1(t) = A_{11}e_1(t) + A_{12}e_y(t) + Lv - LX\xi(t, y, u) \quad (3.78a)$$
$$\dot{e}_y(t) = A_{21}e_1(t) + A_{22}e_y(t) - v + X\xi(t, y, u) \quad (3.78b)$$

A partir de esta introducción al diseño de observadores de modos deslizantes, se efectúa el diseño del observador propuesto por Edwards-Spurgeon.

3.5 Observador de estados de modos deslizantes de Edwards-Spurgeon (ESSMO)

Para el diseño del observador de modos deslizantes de Edwards-Spurgeon (ESSMO), presentado por Sarah Spurgeon y Christopher Edwards en [61], se sintetiza las ganancias de observación lineal y la ganancia de observación no lineal o de inyección discontinua por medio de transformaciones matriciales canónicas, canalizando las perturbaciones hacia los canales medibles del espacio de estados de la planta o proceso. Asimismo, este método evita el uso de otros métodos de síntesis tales como desigualdades lineales matriciales (LMI) o los métodos de diseño por funciones de Lyapunov. Cabe resaltar que las transformaciones lineales matriciales efectuadas por este algoritmo son numerosas, pero permiten efectuar una correcta estimación de las variables de estado reales y efectuar una reconstrucción apropiada de las perturbaciones, siendo esto último una ventaja sustancial respecto a otras soluciones de observación. Cabe resaltar que para la síntesis de las citadas ganancias se utilizó como referencia, en adición a lo descrito en Edwards y Spurgeon [61], el procedimiento indicado en Spurgeon [62,63].

3.5.1 Diseño del Observador de Edwards-Spurgeon

Mediante *el* ***programa del apéndice 2.1*** se efectúa la síntesis de las ganancias lineales y no lineales por medio de transformaciones matriciales canónicas del observador, de acuerdo al procedimiento que se detalla a continuación.

Sea el sistema dinámico incierto en el tiempo en espacio de estados,

$$\dot{x}(t) = Ax(t) + Bu(t) + D\xi(t, y, u) \quad (3.79a)$$
$$y(t) = Cx(t) \quad (3.79b)$$

Donde $x(t)$ es el vector de estados del sistema de dimensión $n\ x\ 1$, $\dot{x}(t)$ es la derivada del vector de estados del sistema de dimensión $n\ x\ 1$, $u(t)$ es la entrada al sistema de dimensión $m\ x\ 1$, $y(t)$ es el vector de mediciones del sistema de dimensión $p\ x\ 1$, $A \in \mathbb{R}^{nxn}, B \in \mathbb{R}^{nxm}, C \in$

$\mathbb{R}^{pxn}, D \in \mathbb{R}^{nxq}$, donde $p \geq q$. Asimismo, $\xi(t, y, u)$ representa incertidumbres tales como no linealidades y desajustes del modelo del sistema no considerados durante la etapa de modelamiento que son desconocidas pero acotadas. La versión digital de este sistema es el modelo lineal incierto 2.23 del capítulo II propuesto,

$$x_{k+1} = F_k x_k + G_k u_k + D_k \xi_k + w_k$$
$$y_k = H_k x_k + v_k$$

Si se considera que el ruido de proceso es $w_k = 0$, la correlación en el tiempo de la sobre aceleración $\alpha = 0$ (para un blanco aéreo altamente maniobrable) y un tiempo de muestreo T = 0.02s, las matrices digitales que figuran en 2.24, 2.25, 2.26 y 2.29 quedan detalladas de la siguiente manera:

$$F_k = \begin{bmatrix} 1 & 0.0200 & 0.0002 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.0200 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.0200 & 0.0002 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.0200 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.0200 & 0.0002 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.0200 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, D_k = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

$$G_k = \begin{bmatrix} 1.33e-06 & 0 & 0 \\ 0.0002 & 0 & 0 \\ 0.0200 & 0 & 0 \\ 0 & 1.33e-06 & 0 \\ 0 & 0.0002 & 0 \\ 0 & 0.0200 & 0 \\ 0 & 0 & 1.33e-06 \\ 0 & 0 & 0.0002 \\ 0 & 0 & 0.0200 \end{bmatrix}, H_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Cabe resaltar que para el diseño del presente observador es necesario que la matriz H_k cuente con capacidad de medir tanto la posición como la velocidad debido a la restricción $p \geq q$ que existe del observador. Por otro lado, según lo descrito en [61], el observador de modos deslizantes para el sistema dinámico incierto puede ser representado de la siguiente manera:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) - G_l \mathrm{Ce(t)} + G_n \mathrm{v}^{17} \qquad (3.80)$$

donde $e(t) = \hat{x}(t) - x(t)$ es el error entre el vector de estados estimados y el vector de estados del sistema, G_l es la matriz de observación lineal de dimensión $n\ x\ p$, G_n es la matriz no lineal de inyección discontinua de dimensión $n\ x\ p$, v es la función de inyección discontinua respecto al colector o superficie deslizante $S_o = \{e \in \mathbb{R}^n : Ce = 0\}$. Cabe resaltar que la versión digital del observador es la siguiente:

$$\hat{x}_{k+1} = F_k \hat{x}_k + G_k u_k - G_l \mathrm{H}_k \mathrm{e}_k + G_n \mathrm{v}$$

Para efectos de mantener la misma nomenclatura que en [61], considerar para el diseño $F_k = A$, $G_k = B$, $D_k = D$ y $H_k = C$. Al respecto, para la síntesis de este observador se deben seguir los siguientes pasos:

1) Se aplica una primera transformación de coordenadas por medio de la matriz de transformación $T_c = \begin{bmatrix} N_c{}^T \\ C \end{bmatrix}$, donde $N_c \in \mathbb{R}^{nx(n-p)}$ abarca el espacio nulo de C, obteniéndose las siguientes matrices del sistema en espacio de estados en el nuevo sistema de coordenadas:

$$\bar{A} = T_c A T_c{}^{-1} \qquad (3.81a)$$
$$\bar{D} = T_c D \qquad (3.82b)$$

[17] El observador de estados existe si y solo sí se cumple $rango(CD) = q$ y que la tripleta matricial *A,D,C* sea de grado relativo uno y de fase mínima.

$$\bar{C} = CT_c{}^{-1} \tag{3.81c}$$

Para este paso, la transformación elegida fue $T_c = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

2) Se particiona la matriz de distribución de entradas en el nuevo sistema de coordenadas $\bar{D} = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix}$, donde $D_1[1{:}\,\mathrm{n} - \mathrm{p}][m]$ y $D_2[n\ -\ p + 1{:}\,n][m]$.

3) Se halla la transformación T_b a fin de canalizar las perturbaciones del sistema hacia los canales de salida $y(t)$ del sistema (hacia las variables de estado medibles) mediante una descomposición QR, de la siguiente manera:

- Aplicar descomposición QR a la matriz D_2 tal que $D_2 = QR$, donde R es una matriz triangular superior, de dimensión $[n\ -\ p + 1{:}\,n][m]$, y Q es una matriz ortogonal, de dimensión $[n\ -\ p + 1{:}\,n][n\ -\ p + 1{:}\,n]$.
- Cambiar el orden de los elementos filas de la matriz Q^T en la dirección arriba-abajo y aplicar la transpuesta del resultado para hallar la matriz QT.
- Hallar la transformación $T_b = \begin{bmatrix} I_{n-p} & -D_1({D_2}^T D_2){D_2}^T \\ 0_{P\ x\ n-p} & QT^T \end{bmatrix}$

4) Se aplica la transformación a las matrices $\bar{A}, \bar{D}\ y\ \bar{C}$ de la siguiente manera:

$$\bar{\bar{A}} = T_b \bar{A} {T_b}^{-1} \tag{3.82a}$$
$$\bar{D} = T_b \bar{D} \tag{3.82b}$$
$$\bar{\bar{C}} = \bar{C} {T_b}^{-1} \tag{3.82c}$$

Cabe resaltar que en este paso es importante verificar si el sistema cuenta con ceros de transmisión.

5) A partir de la transformación realizada en el paso anterior, se particiona a la matriz $\bar{\bar{A}} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$. La sub matriz $A_{11} = \begin{bmatrix} A_{11}^0 & A_{12}^0 \\ 0 & A_{22}^0 \end{bmatrix}$ es de dimensión $(n-p)\ x\ (n-p)$, siendo A_{11}^0 de dimensión $r\ x\ r$, para un $r \geq 0$ dependiente de la dimensión del sub espacio no observable o el número de ceros de transmisión, la sub matriz $A_{21} = \begin{bmatrix} A_{211} \\ A_{212} \end{bmatrix}$ es particionada de tal forma que contiene a $A_{211} = [0 \quad A_{21}^0]$ de dimensión $(p-q)\ x\ (n-p-r)$. Cabe resaltar que el par de matrices A_{21}^0, A_{22}^0 debe ser completamente observable. Asimismo, los autovalores de A_{11}^0 son los ceros invariantes de la tripleta A, D, C, por lo que, si se asume que estos son estables, entonces la matriz A_{11}^0 es estable y existe en consecuencia una matriz de observación lineal L de dimensión $(n-p)\ x\ (p-q)$ tal que la matriz $A_{11} + LA_{211}$ es estable.

6) Posteriormente, se establece la tripleta matricial A_{11}, $0_{n-p\ x\ 1}$ y A_{211}, y se coloca en forma canónica observable de la siguiente manera:

- Si la matriz de observación del par (A_{11}, A_{211}) tiene rango $r \leq n\ -\ p$, donde $n\ -\ p$ es la dimensión de A_{11}, existe una transformación matricial tal que: $\bar{A}_{11} = T_{obs} A_{11} {T_{obs}}^T$, $\bar{0}_{n-p\ x\ 1} = T_{obs} 0_{n-p\ x\ 1}$ y $\bar{A}_{211} = A_{211} {T_{obs}}^T$, donde T_{obs} es unitario y el sistema transformado posee una forma escalonada, con los modos no observables en la esquina superior izquierda.

$$\bar{A}_{11} = \begin{bmatrix} A_{no} & A_{12} \\ 0 & A_o \end{bmatrix} \quad (3.83a)$$

$$\bar{0}_{n-p\,x\,1} = \begin{bmatrix} B_{no} \\ B_o \end{bmatrix} \quad (3.83b)$$

$$\bar{A}_{211} = [0 \quad C_o] \quad (3.83c)$$

Donde, el par matricial (C_0, A_0) es observable y los autovalores de la matriz A_{no} son los modos no observables.

- Se halla la dimensión del espacio no observable o el número de ceros invariantes mediante la siguiente ecuación: $dim_{no} = n - p - dim(A_0)$

7) Por medio de la matriz T_{obs} se forma la matriz de transformación T_f y se efectúa una transformación de coordenadas a la tripleta matricial (3.82) de la siguiente manera:

$$T_f = \begin{bmatrix} T_{obs} & 0_{n-p\,x\,p} \\ 0_{p\,x\,n-p} & I_p \end{bmatrix} \quad (3.84)$$

$$\bar{\bar{A}} = T_f \bar{A} {T_f}^{-1} \quad (3.85a)$$

$$\bar{\bar{D}} = T_f \bar{D} \quad (3.85b)$$

$$\bar{\bar{C}} = \bar{C} {T_f}^{-1} \quad (3.85c)$$

8) A partir de estos resultados, se debe verificar que la transformación se ha efectuado correctamente si es que el sistema no cuente con ceros invariantes en el plano derecho del locus o lugar geométrico de las raíces; si se cumple $dim_{no} > 0$, se extrae la partición $A_{11_o} = \bar{\bar{A}}[1:r][1:r]$ y se verifica que sus autovalores no se encuentren en el plano derecho del locus. Si se encuentra en el plano izquierdo, entonces se ha efectuado la transformación correcta. De lo contrario, se cuenta con ceros invariantes inestables y la transformación T_{obs} debe ser reformulada.

9) Se computa la transformación parcial $T_o = T_f T_b T_c$ con la finalidad de verificar que la forma canónica obtenida hasta el momento es factible.

10) Se procede a ubicar los polos de los modos deslizantes bajo las siguientes condiciones:

- Si $n - p - dim_{no} > 0$, se hallan las particiones $\mathrm{A_{22_o}} = \bar{\bar{\mathrm{A}}}[\mathrm{r}+1{:}\,\mathrm{n}-\mathrm{p}][\mathrm{r}+1{:}\,\mathrm{n}-\mathrm{p}]$ y $\mathrm{A_{21_o}} = \bar{\bar{\mathrm{A}}}[\mathrm{n}-\mathrm{p}+1{:}\,\mathrm{n}-\mathrm{q}][\mathrm{r}+1{:}\,\mathrm{n}-\mathrm{p}]$. Posteriormente, se elige el vector de los polos de la ganancia lineal, $p_l = [p_1\, p_2 \,..\, p_m]$, y se efectúa el procedimiento de ubicación de polos utilizando las matrices $\mathrm{A_{22_o}}^T$ y $\mathrm{A_{21_o}}^T$ y el vector de polos p_l, resultando el vector de ganancias L_T. Los polos seleccionados fueron $p_l = [-0.92 \; -0.99 \; -0.97]$.
- Si $dim_{no} > 0$, el vector de ganancias $L = -\begin{bmatrix} 0_{r\,x\,p-q} \\ L_T \end{bmatrix}$. De lo contrario, $L = -L_T$.
- Si $n - p - dim_{no} \leq 0$, $L = 0_{n-p\,x\,n-p}$

Finalmente, se halla la matriz de ganancias lineales $\bar{L} = [L \quad 0_{n-p\,x\,q}]$ siendo esta igual a $\bar{L} = \begin{bmatrix} 67.8465 & 67.8465 & -95.9495 & 0 & 0 & 0 \\ 70.3464 & 70.3464 & 99.4848 & 0 & 0 & 0 \\ -98.4747 & 98.4747 & 0 & 0 & 0 & 0 \end{bmatrix}$.

11) Del procedimiento de ubicación de polos anterior y cómputo de la matriz $\bar{L}$, se efectúa una transformación que permite que la matriz $\bar{\bar{A}}$ posea una sub matriz A_{11} con autovalores estables. Asimismo, la transformación permite que las matrices $\bar{\bar{C}}$ y $\bar{\bar{D}}$ tomen la forma $\bar{\bar{C}} = [0 \quad I]$ y $\bar{\bar{D}} = [0 \quad {D_2}^T]$. La transformación y matrices transformadas se encuentran detalladas a continuación:

$$T = \bar{\bar{\bar{C}}}[:][n-p+1:n]$$
$$T_L = \begin{bmatrix} I_{n-p} & \bar{L} \\ 0_{p\,x\,n-p} & T \end{bmatrix}$$
$$\bar{\bar{\bar{A}}} = T_L \bar{\bar{\bar{A}}} {T_L}^{-1}$$
$$\bar{\bar{\bar{D}}} = T_L \bar{\bar{\bar{D}}} \quad (3.86)$$
$$\bar{\bar{\bar{C}}} = \bar{\bar{\bar{C}}} {T_L}^{-1} \quad (3.87)$$
$$T_T = T_L T_o$$

12) A partir de la transformación realizada, se particiona la matriz $\bar{\bar{\bar{A}}}$ con la finalidad de obtener las matrices que conforman la ganancia lineal de Luenberger del observador:

$$A_{12} = \bar{\bar{\bar{C}}}[1:n-p][n-p+1:n] \quad (3.88a)$$
$$A_{22} = \bar{\bar{\bar{C}}}[n-p+1:n][n-p+1:n] \quad (3.88b)$$

13) Posteriormente, se eligen los polos deseados correspondientes a la ganancia no lineal del observador para formar la matriz diagonal A_{22}^s, que tiene por diagonal los elementos del vector de polos $p_{nl} = [p_1, \ldots, p_{n-p}]$, siendo este vector seleccionado como $p_{nl} = [-0.35 \; -0.35 \; -0.35 \; -0.35 \; -0.35 \; -0.35]$. Bajo los criterios de estabilidad de Lyapunov descritos en el apartado (3.2.5), se definen las matrices de pesos Q_1 y Q_2 de dimensiones $n-p\,x\,n-p$ y $p\,x\,p$, respectivamente, así como la matriz simétrica positiva definida P_2, de dimensión $p\,x\,p$; esta última debe ser considerada como la solución única a la siguiente ecuación de Lyapunov:

$$P_2 A_{22}^s + {A_{22}^s}^T P_2 = -Q_2 \quad (3.90)$$

Y mediante la siguiente ecuación de Lyapunov se halla la matriz simétrica positiva definida P_1, de dimensión $n-p\,x\,n-p$, como la única solución a la siguiente ecuación de Lyapunov:

$$P_1 A_{11} + {A_{11}}^T P_1 = -({A_{21}}^T P_2 {Q_2}^{-1} P_2 A_{21} + Q_1) \quad (3.91)$$

14) Finalmente, las ganancias lineales y no lineales del observador se computan de la siguiente forma:

$$G_l = {T_T}^{-1} \begin{bmatrix} A_{12} \\ A_{22} - A_{22_s} \end{bmatrix} \quad (3.92a)$$
$$G_n = \|D_2\| {T_T}^{-1} \begin{bmatrix} 0_{n-p\,x\,p} \\ I_p \end{bmatrix} \quad (3.92b)$$

Cabe resaltar que para la implementación del observador mediante las ganancias G_l y G_n se trabaja con las matrices originales del sistema.

Después de efectuar todas las transformaciones, la tripleta del sistema (3.88) permite colocar al espacio de estados del sistema en forma canónica, separando los estados medibles de los no medibles y canalizando las perturbaciones del sistema hacia los estados medibles de este, siendo esta la ventaja sustancial de trabajar con una forma canónica.

$$\dot{x}_1(t) = A_{11} x_1(t) + A_{12} y(t) + B_1 u(t) \quad (3.93a)$$
$$\dot{y}(t) = A_{21} x_1(t) + A_{22} y(t) + B_2 u(t) + D_2 \xi(t, y, u) \quad (3.93b)$$

Donde $x_1(t)$ es el vector de estados no medibles de dimensión $n - p\,x\,1$, $y(t)$ es el vector de estados medibles de dimensión $p\;x\,1$, $u(t)$ es la señal de entrada al sistema de dimensión $m\,x\,1$ y la matriz A_{11} es estable de acuerdo a lo anteriormente expuesto. Por consiguiente, del modelo (3.93) se obtiene el observador de modos deslizantes de Edwards-Spurgeon:

$$\dot{\hat{x}}_1(t) = A_{11} \hat{x}_1(t) + A_{12} \hat{y}(t) + B_1 u(t) - A_{12} e_y(t) \quad (3.94a)$$
$$\dot{\hat{y}}(t) = A_{21} \hat{x}_1(t) + A_{22} \hat{y}(t) + B_2 u(t) - (A_{22} - A_{22}^s) e_y(t) + v \quad (3.94b)$$

Y el sistema dinámico del error:

$$\dot{e}_1(t) = A_{11}e_1(t) \quad (3.95a)$$
$$\dot{e}_y(t) = A_{21}e_1(t) + A_{22}^s e_y(t) + v - D_2\xi(t,y,u) \quad (3.95b)$$

Donde $e_y(t) = Ce(t) = \hat{y}(t) - y(t)$ es el error de estados medibles, $e_1(t) = \hat{x}_1(t) - x_1(t)$ y A_{22}^s es una matriz diseñada de tal forma que los elementos de su diagonal contengan los polos deseados de la matriz lineal G_l, los cuales se recomienda que se encuentren a una distancia entre tres a cinco veces a la izquierda de los polos del sistema.

Al contar con las matrices P_1 y P_2, se puede aplicar el método directo de estabilidad de Lyapunov descrito en el apartado 3.2.5, definiendo a la siguiente función candidata de Lyapunov:

$$V(e_1, e_y) = \frac{1}{2}e_1(t)^T P_1 e_1(t) + \frac{1}{2}e_y(t)^T P_2 e_y(t) \quad (3.96)$$

Se verifica que $V(e_1, e_y) > 0 \; \forall \; e_1(t), e_y(t) \in D - \{0\}, V(0) = 0$ dado que P_1 y P_2 son matrices simétricas positivas definidas. Asimismo, la derivada de la función candidata de Lyapunov:

$$\dot{V}(e_1, e_y) = \dot{e}_1(t)^T P_1 e_1(t) + e_1(t)^T P_1 \dot{e}_1(t) + \dot{e}_y(t)^T P_2 e_y(t) + e_y(t)^T P_2 \dot{e}_y(t) \quad (3.97)$$

Reemplazando (3.95) en (3.97) se obtiene:

$$\dot{V}(e_1, e_y) = 2e_1(t)^T P_1 A_{11} e_1(t) + 2e_y(t)^T P_2 \left(A_{21}e_1(t) + A_{22}^s e_y(t) + v - D_2\xi(t,y,u)\right)$$

A fin de que $\dot{V}(e_1, e_y) < 0$ y se cumpla la condición de que e_1, e_y sean localmente asintóticamente estables, se define el sistema de estructura variable o función discontinua v de la siguiente forma:

$$v = \begin{cases} -\rho(t,y,u)\|D_2\| \dfrac{P_2 e_y}{\|P_2 e_y\|}, & para \; e_y \neq 0 \\ 0, & para \; e_y = 0 \end{cases}$$

Donde la función escalar $\rho(t,y,u) \geq r_1\|u(t)\| + \alpha(t,y) + \gamma_0$ cuenta con las constantes r_1, y γ_0 y la función $\alpha(t,y)$ elegidas a criterio del diseñador.

Una de las capacidades más resaltantes del observador de modos deslizantes de Edward-Spurgeon es la de estimar la perturbación o no linealidad del sistema a partir del control equivalente de la función discontinua v.

Según lo descrito en [61], si se considera el sistema dinámico del error que figura en (3.95) se tiene:

$$\dot{e}_1(t) = A_{11}e_1(t) \quad (3.98a)$$
$$\dot{e}_y(t) = A_{21}e_1(t) + A_{22}^s e_y(t) + v - D_2 f_i(t) \quad (3.98b)$$

Si $f_0(t)$ es una perturbación del vector de medición del sistema y se considera que $f_0(0) = 0$ y que se ha establecido modo deslizante apropiado tal que $e_y = 0$ y $\dot{e}_y = 0$, del sistema establecido en (3.98) se tiene:

$$\dot{e}_1(t) = A_{11}e_1(t)$$
$$0 = A_{21}e_1(t) + v_{eq} - D_2 f_i(t)$$

Donde v_{eq} es el denominado control equivalente aplicado durante los modos deslizantes del sistema. Por tanto, de acuerdo a lo descrito en [61] si $e_1(t) \to 0$ entonces $v_{eq} = D_2 f_i(t)$ y al reemplazar el control discontinuo por una aproximación continua se tiene:

$$v_\delta = -\rho(t,y,u)\|D_2\|\frac{P_2 e_y}{\|P_2 e_y\| + \delta} \tag{3.99}$$

donde δ es un escalar positivo pequeño. Por tanto, el control equivalente puede ser aproximado con un error de aproximación proporcional al escalar δ elegido. Asimismo, la reconstrucción de la señal de perturbación se puede efectuar de la siguiente manera:

$$f_i(t) = -\rho(t,y,u)\|D_2\|(D_2{}^T D_2)^{-1} D_2{}^T \frac{P_2 e_y}{\|P_2 e_y\| + \delta} \tag{3.100}$$

Por otro lado, si se considera a la perturbación de entrada al sistema $f_i(t) = 0$ y se estudia el efecto de $f_o(t)$, dado que $y(t) = Cx(t) + f_o(t)$ entonces $e_y(t) = Ce(t) - f_o(t)$. Por tanto, el sistema dinámico del error está representado de la siguiente manera:

$$\dot{e}_1(t) = A_{11}e_1(t) + A_{12}f_o(t)$$
$$\dot{e}_y(t) = A_{21}e_1(t) + A_{22}^s e_y(t) - \dot{f}_o(t) + A_{22}f_o(t) + v$$

Durante los modos deslizantes se cumple que $0 = A_{21}e_1(t) - \dot{f}_o(t) + A_{22}f_o(t) + v_{eq}$ y por consiguiente, el control equivalente viene dado de la siguiente forma:

$$v_{eq} \cong -(A_{22} - A_{21}A_{11}{}^{-1}A_{12})f_0 \tag{3.101}$$

Al ejecutar el programa antes mencionado se logró la síntesis de las siguientes matrices:

$$G_l = \begin{bmatrix} 1.3310 & 0.0394 & 0 & 0 & 0 & 0 \\ -1.8990 & 3.2890 & 0 & 0 & 0 & 0 \\ -128.1818 & 130.8818 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.3303 & 0.0401 & 0 & 0 \\ 0 & 0 & -1.9697 & 3.3597 & 0 & 0 \\ 0 & 0 & -132.9545 & 135.6545 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.3305 & 0.0399 \\ 0 & 0 & 0 & 0 & -1.9495 & 3.3395 \\ 0 & 0 & 0 & 0 & -131.5909 & 134.2909 \end{bmatrix}$$

$$G_n = \begin{bmatrix} 1.4142 & 0.0000 & 0 & 0 & 0 & 0 \\ 0.0000 & 1.4142 & 0 & 0 & 0 & 0 \\ -134.2789 & 137.1073 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.4142 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.4142 & 0 & 0 \\ 0 & 0 & -139.2786 & 142.1070 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.4142 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.4142 \\ 0 & 0 & 0 & 0 & -137.8501 & 140.6785 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 1.43e-04 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.43e-04 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.43e-04 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.43e-04 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.43e-04 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.43e-04 \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

3.5.2 Algoritmo del observador de Edwards-Spurgeon

A continuación, se muestra el pseudocódigo del programa de MATLAB® del apéndice 2.2 que permite implementar el observador de modos deslizantes de Edward-Spurgeon (ESSMO) mediante una función:

Algoritmo 3.1: Observador de Edwards-Spurgeon (ESSMO)

Entradas: Matrices $\mathbf{G_l}$, $\mathbf{G_n}$, $\mathbf{P_2}$, $\mathbf{D_2}$,$\mathbf{F_k}$ y $\mathbf{G_k}$,vectores $\mathbf{y_k}$, $\mathbf{u_k}$ y $\mathbf{x_k}$, constante **g** y muestra actual **k.**
Salidas: Vector discontinuo **v,** vector de perturbaciones estimadas **fi** y variables de estado (VE) estimadas $\boldsymbol{xh_k}$**.**
Inicio del Programa: Invocado por el programa **sim_ESSMO.m**
1: Declara e inicializa variable persistente **xhatk** y **yhatk**
2: Halla error de estados medibles **ey=yhatk-yk**
3: Declara **ro=[0.005*g;0.005*g;0.005*g;0.005*g;0.005*g;0.005*g]**
4: Calcula la función discontinua **v=-ro.*(norm(D2)*P2*ey./norm(P2*ey))**
5: Halla VE estimadas futuras **xhatk_1= Fk*xhatk + Gk*uk -Gl*ey + Gn*v**
6: Declara delta = 0.0001
7: Reconstruye perturbación (control equivalente**) fi = -[1.0*g; 0.85*g; 0.9*g].*(norm(D2)*inv(D2'*D2)*D2'*(P2*ey./(norm(P2*ey)+delta)))**
8: Almacena **yhatk**
9: Guardar variables para siguiente iteración
10: Fin de programa

3.5.3 Simulación de la trayectoria del misil

Mediante el programa del apéndice 2.3 se contrastan las variables de estado reales y medidas del modelo con las estimaciones del observador diseñado, tomando como referencia las simulaciones efectuadas en el capítulo II relacionadas al vector de entrada de control, vector de perturbaciones, distribución gaussiana mixta y simulación de la trayectoria del misil. Por ello, se presenta su algoritmo a continuación:

Algoritmo 3.2: Simulación de trayectoria del blanco con observador ESSMO

Entradas: Señal de entrada de control $\boldsymbol{\mu_k}$**,** perturbaciones $\boldsymbol{\xi_k}$ y modelo mixto gaussiano (GMM)
Salidas: Vector de variables de estado (VE) reales del modelo $\boldsymbol{x_k}$, VE medidas del modelo $\boldsymbol{y_k}$**,** VE estimadas $\boldsymbol{xh_k}$**,** perturbaciones reconstruidas $\boldsymbol{ph_k}$, errores cartesianos $\boldsymbol{error\_c_k}$, errores polares $\boldsymbol{error\_c_k}$**,** error **RMSE** y gráficos.
Inicio del programa:
1: Declara tiempo de muestreo $\boldsymbol{T}$, gravedad $\boldsymbol{g}$**,** constante de mach **M**, sobre aceleración máxima ***load_factor*** tiempo inicial de simulación $\boldsymbol{t_i}$**,** tiempo final de simulación t_{ff} y número total de muestras $\boldsymbol{nt}$**.**
2: Declara condiciones iniciales de traqueo al blanco: distancia $\boldsymbol{Ad}$**,** acimut verdadero $\boldsymbol{Bdn}$, elevación $\boldsymbol{Ed}$**,** velocidad $\boldsymbol{Vm}$ y rumbo verdadero $\boldsymbol{Rv}$**.**
3: Convierte posición de coordenadas polares a cartesianas usando $\boldsymbol{Ad, Bdn\ y\ Ed}$
4: Declara el vector de estados inicial $\boldsymbol{x_k(0)}$
5: Declara matrices del espacio de estados $\boldsymbol{F_k}$, $\boldsymbol{G_k}$, $\boldsymbol{D_k}$ y $\boldsymbol{C_k}$.
6: Obtén $\boldsymbol{u_k}$ (invoca a función ***gen_jerk_sea_skimming3.m*** **(programa del apéndice 1.5))**
7: Obtén $\boldsymbol{\xi_k}$ **(invoca** a función ***perturb_1.m*** **(programa del apéndice 1.6))**
8: Cargar ruido angular ***glint_puntos.mat*** **(datos guardados del programa del apéndice 1.7)**
9: Invoca **ESSMO_sintesis.m** para cargar matrices del observador
10: Ingrese "1" para simulación con ruido y "0" para simulación sin ruido.
11: Para tt = ti:T:tff **Hacer**
12: Ejecuta el modelo de transición de estados para obtener $\boldsymbol{x_k}$ futuro.
13: Invoca ***c2p.m*** para convertir $\boldsymbol{x_k}$ a coordenadas polares.
14: Invoca ***yk_noise.m*** para obtener el vector de mediciones y_k
15: Invoca **ESSMO.m** para obtener VE estimadas $\boldsymbol{xh_k}$ y las perturbaciones estimadas **fi**
16: Invoca **c*2p.m*** para convertir $\boldsymbol{xh_k}$ a coordenadas polares.
17: Extrae errores polares, errores cartesianos y RMSE
18: Almacena vectores para gráficas
19: Fin Para
20: Conversión de unidades del vector de estados a Mn, Mach y g's.
21: Graficar
22: Fin de programa

Al efectuar la simulación se observó que el ESSMO en la ausencia de ruido posee buena robustez y exactitud en la estimación de las variables de estado (figuras 3.5,3.7 y 3.9), sin embargo, en la simulación con ruido (figuras 3.6, 3.8 y 3.10) se observó una alta sensibilidad al ruido angular; se intentó buscar una configuración de ganancias que permita obtener una relación aceptable de robustez-ruido, sin embargo, para todas las configuraciones el observador se mostró bastante sensible al ruido. Por otro lado, en las figuras 3.11 y 3.12 se observa un alto nivel de chattering o castañeo.

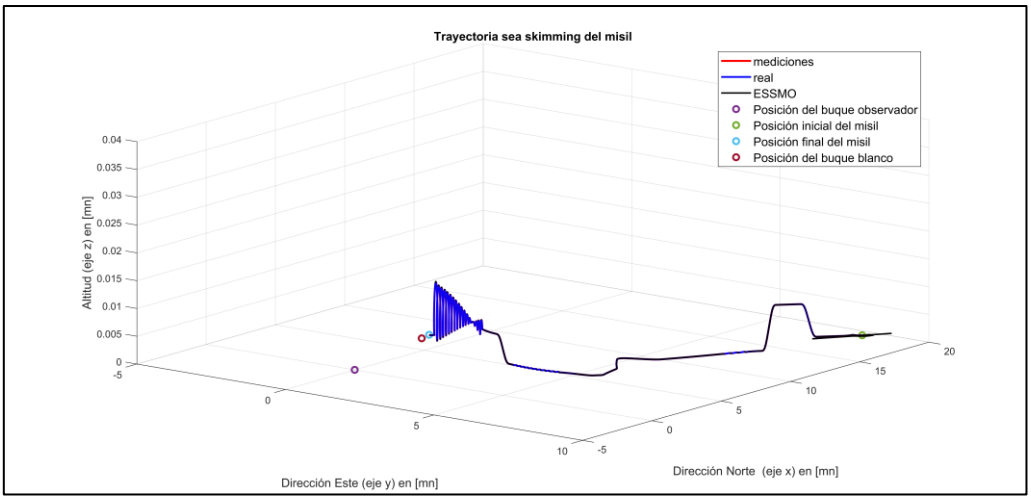

Figura 3.5.: Trayectoria en tres dimensiones sin ruido (ESSMO).
Fuente: Elaboración propia.

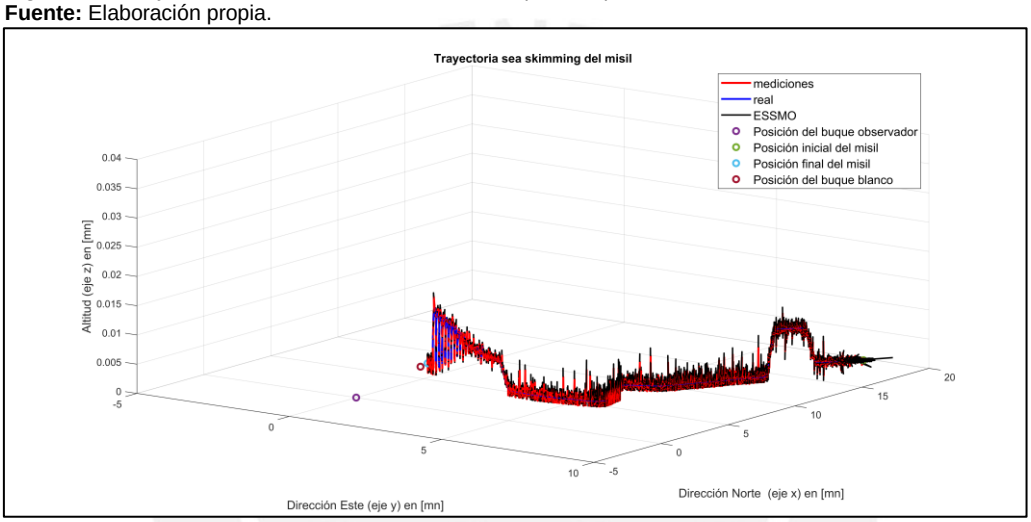

Figura 3.6.: Trayectoria en tres dimensiones con ruido (ESSMO).
Fuente: Elaboración propia.

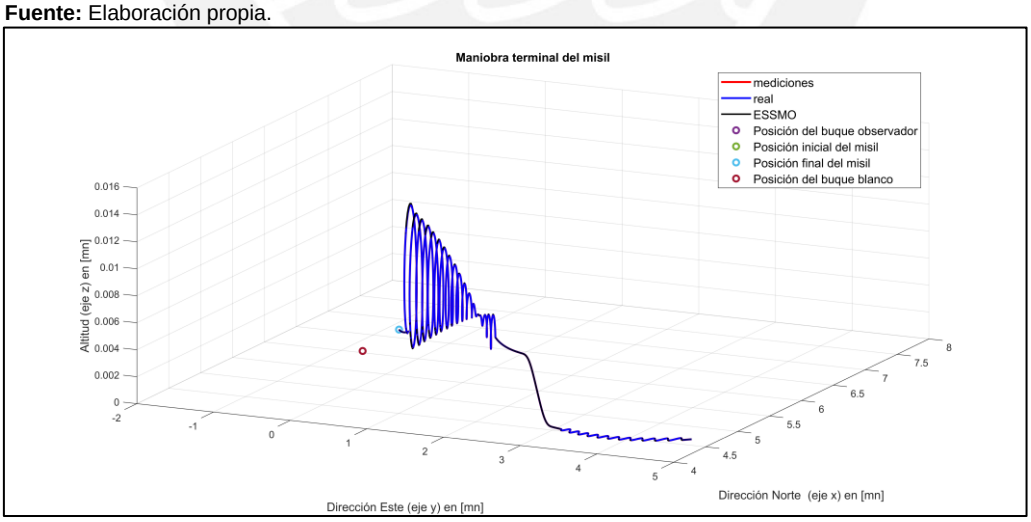

Figura 3.7.: Maniobra terminal del misil sin ruido (ESSMO).
Fuente: Elaboración propia.

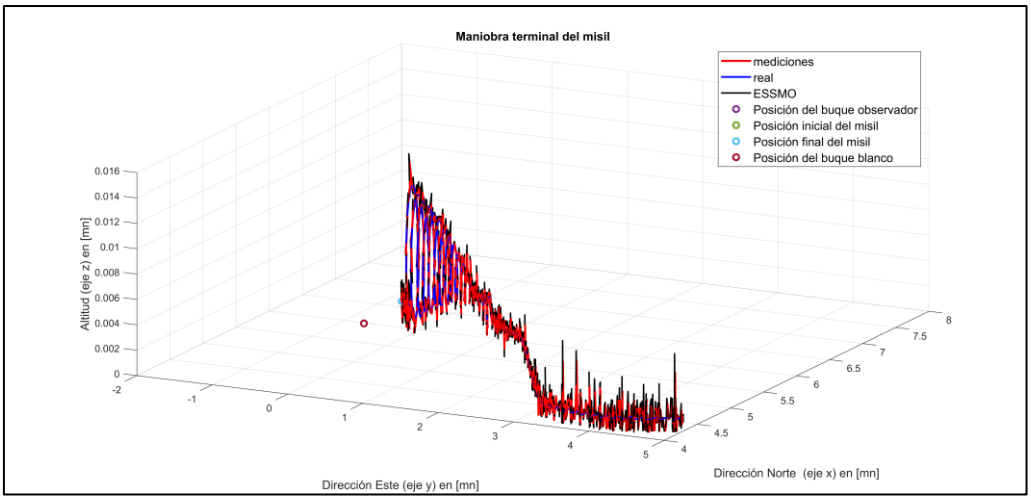

Figura 3.8.: Maniobra terminal del misil con ruido (ESSMO).
Fuente: Elaboración propia.

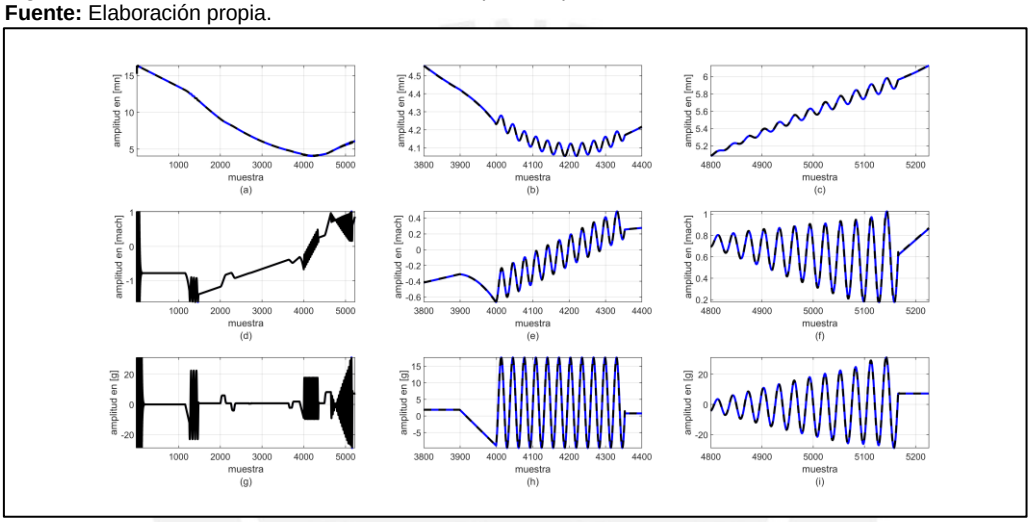

Figura 3.9.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "x" en ausencia de ruido (ESSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [3800 4400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [3800 4400]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

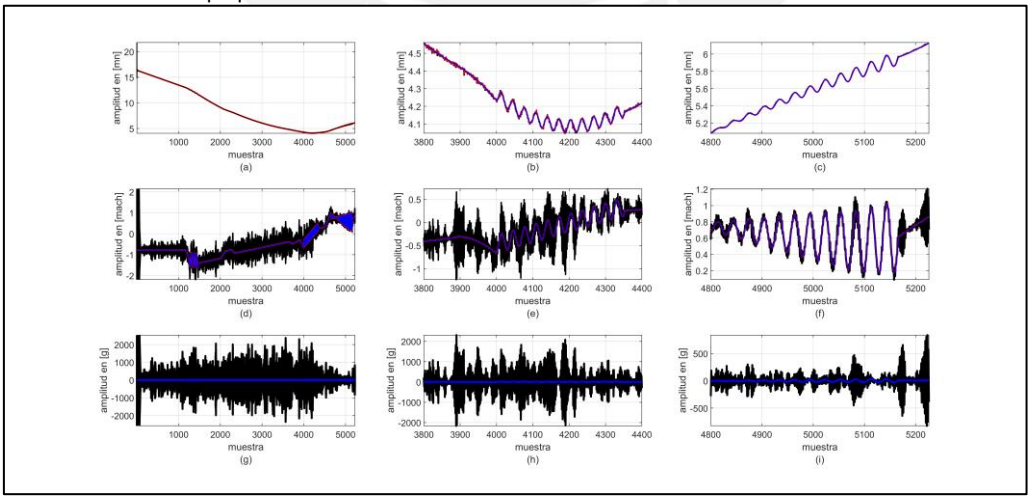

Figura 3.10.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "x" con ruido (ESSMO). Variable de estado real (línea azul). Variable de estado medida (línea roja). Variable de estado estimada (línea negra) (a) Posición (k = [0 5226]). (b) Posición (k = [3800 4400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [3800 4400]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

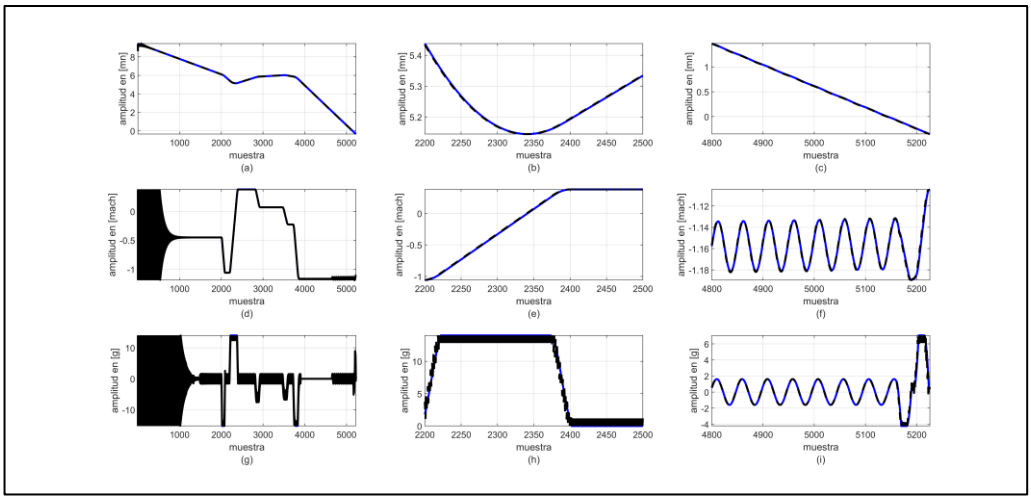

Figura 3.11.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada “y” en la ausencia de ruido (ESSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [2200 2500]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [2200 2500]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [2200 2500]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

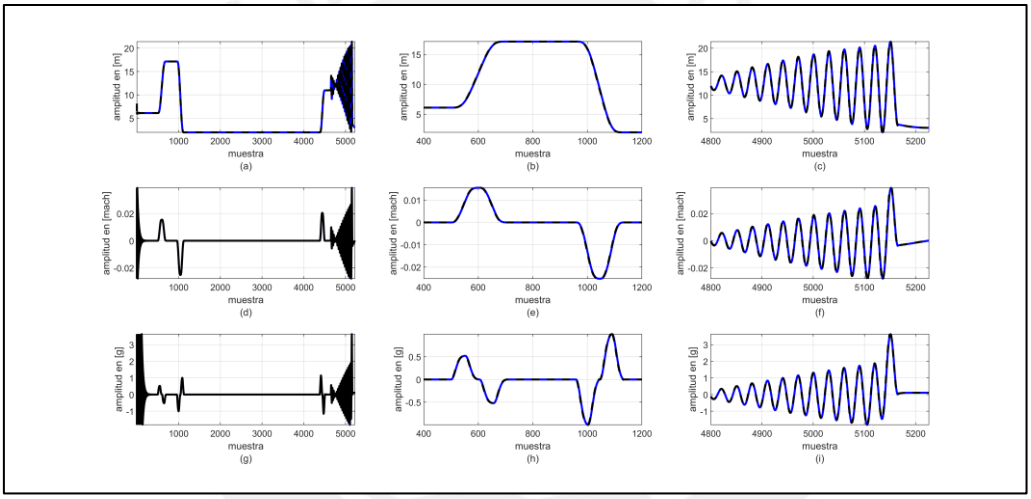

Figura 3.12.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada “z” en la ausencia de ruido (ESSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [400 1200]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [400 1200]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

En la figura 3.13 se observa que los errores cartesianos aumentan durante la maniobra terminal, sin embargo, al convertir los errores cartesianos a errores polares se verifica en la figura 3.14 que en la ausencia de ruido los errores cartesianos se mantienen dentro de los valores máximos permisibles. Por otro lado, en la figura 3.15 se observa que el ESSMO efectúa una reconstrucción adecuada de las perturbaciones al modelo, por medio del método de control equivalente.

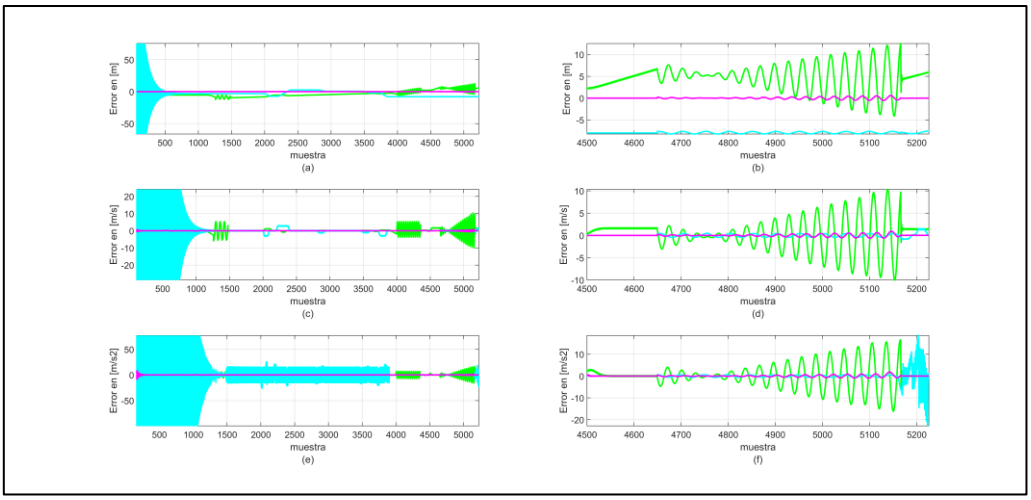

Figura 3.13.: Errores cartesianos (ESSMO). Coordenada “x” (línea verde). Coordenada “y” (línea cyan). Coordenada “z” (línea magenta). (a) Posición k = [0 5226] (b) Posición k = [4800 5226] (c) Velocidad k = [0 5226] (d) Velocidad k = [4800 5226] (e) Aceleración k = [0 5226] (f) Aceleración k = [4800 5226].
Fuente: Elaboración propia.

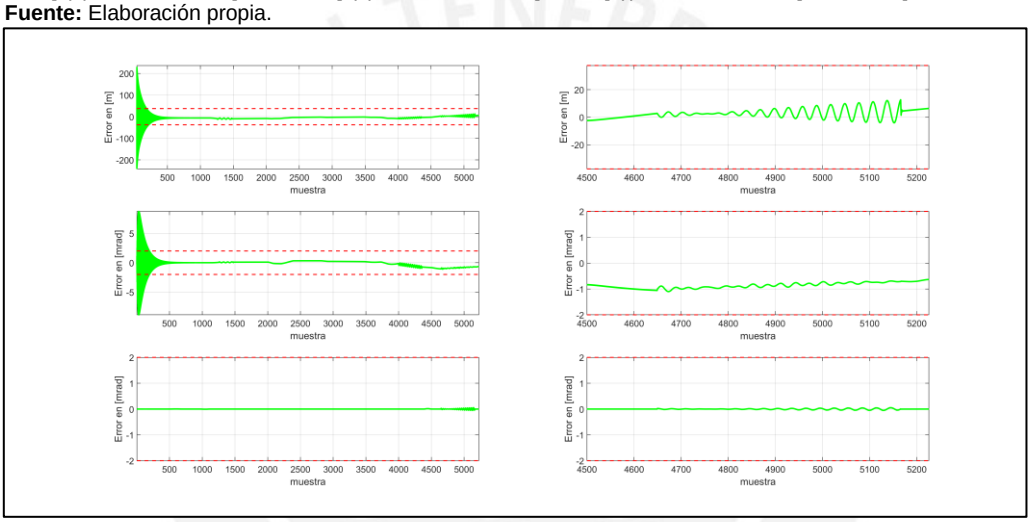

Figura 3.14.: Errores polares (ESSMO). (a) Alcance Ad k = [0 5226] (b) Alcance Ad k = [4800 5226] (c) Azimuth Bdn k = [0 5226] (d) Azimuth Bdn k = [4800 5226] (e) Elevación Ed k = [0 5226] (f) Elevación Ed k = [4800 5226].
Fuente: Elaboración propia.

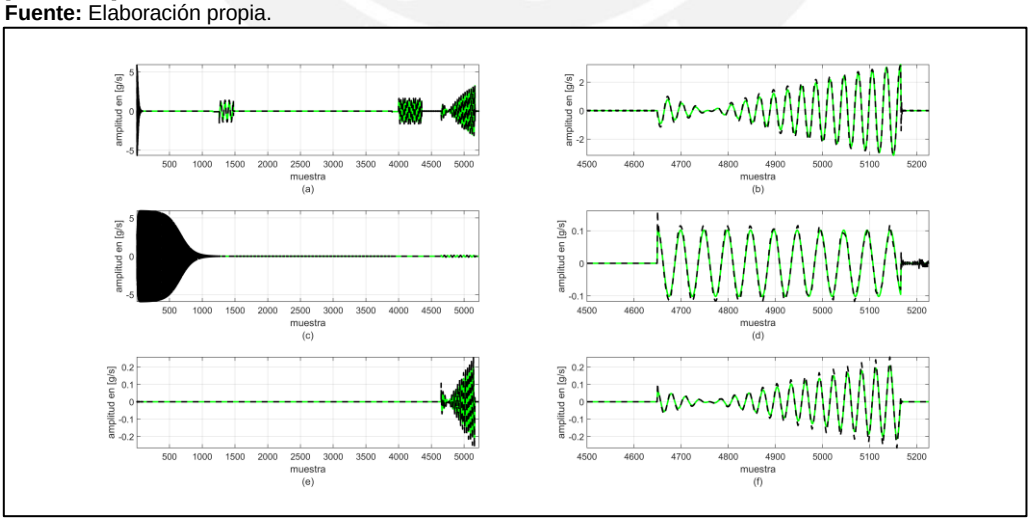

Figura 3.15.: Reconstrucción de perturbaciones (ESSMO). Perturbación real (línea verde). Perturbación estimada (línea negra) (a) Coordenada “x” k = [0 5226] (b) Coordenada “x” k = [4800 5226] (c) Coordenada “y” k = [0 5226] (d) Coordenada “y” k = [4800 5226] (e) Coordenada “z” k = [0 5226] (f) Coordenada “z” k = [4800 5226].
Fuente: Elaboración propia.

3.6 Observador de modos deslizantes de Walcott-Zak (WZSMO)

De acuerdo a lo descrito en el apartado anterior, para diseñar el observador de modos deslizantes de Edwards-Spurgeon es necesario efectuar la síntesis de las ganancias G_l y G_n por medio de transformaciones matriciales, a fin de llevar el espacio de estados del sistema a su forma canónica; el proceso de síntesis fue complejo debido a la cantidad de transformaciones realizadas, sin embargo, se lograron estimaciones robustas y exactas en la ausencia de ruido. Por otro lado, fue evidente en la simulación con ruido que el observador es bastante sensible al ruido angular. Por ello, con la finalidad de ofrecer un método más sencillo de síntesis de las ganancias y mejorar la sensibilidad al ruido, Xiang et al. [64] exponen un método de diseño por medio de desigualdades lineales matriciales (LMI).

3.6.1 Fundamentos teóricos

Sea el sistema lineal incierto detallado en (2.23), el observador de modos deslizantes de Walcott-Zak (WZSMO) posee la siguiente forma:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) - G_l\big(\mathrm{C}\hat{x}(t) - \mathrm{y(t)}\big) + G_n\mathrm{v(t)} \tag{3.102}$$

donde $\hat{x}(t)$ es el vector de estados estimados, G_l es la ganancia de retroalimentación lineal, $u(t)$ es la entrada de control, $\mathrm{y(t)}$ es el vector de estados medidos, v es la entrada discontinua y G_n es la matriz de retroalimentación no lineal. Por lo que se puede observar su estructura es similar al observador de Edwards-Spurgeon, por lo que la versión digital es similar a la de este.

Según lo descrito en [63], la ***condición de existencia*** del observador de Walcott-Zak WZSMO radica en que se cumpla lo siguiente:

1) $P(A - G_lC) + (A - G_lC)^T P = -Q$ (3.103a)
2) $D^T P = FC$ (3.103b)
3) Las matrices positivas semi definidas P y Q, la matriz de retroalimentación lineal G_l y la matriz F, que afecta directamente la implementación de la función discontinua v, satisfacen (1) y (2) si y solo sí $rango(CD) = rango(D)$ y la tripleta matricial $\{A, D, C\}$ es de fase mínima (3.103c).

Adicionalmente, se considera la existencia de dos matrices simétricas W_1 y W_2, de dimensiones $(n-q)\,x\,(n-q)$ y $p\,x\,p$, y una matriz Y de dimensiones $n\,x\,p$, que permiten establecer las siguientes desigualdades:

1) $D^{\perp}W_1D^{\perp^T} + C^TW_2C > 0$ (3.104a)
2) $D^{\perp}W_1D^{\perp^T}A + C^TW_2CA - YC + \left(D^{\perp}W_1D^{\perp^T}A + C^TW_2CA - YC\right)^T < 0$ (3.104b)

donde $D^{\perp}$ es el complemento ortogonal de la matriz de distribución de perturbaciones D.

Dado que las matrices P y Q son positivas semi definidas se iguala P con (3.104a) y $-Q$ con (3.104b) de la siguiente forma:

1) $P = D^{\perp}W_1D^{\perp^T} + C^TW_2C > 0$ (3.105a)
2) $-Q = D^{\perp}W_1D^{\perp^T}A + C^TW_2CA - YC + \left(D^{\perp}W_1D^{\perp^T}A + C^TW_2CA - YC\right)^T < 0$ (3.105b)

Posteriormente, se reemplaza (3.105a) y (3.105b) en (3.103a) de la siguiente forma:

$$P(A - G_lC) + (A - G_lC)^T P = -Q$$
$$(D^{\perp}W_1D^{\perp^T} + C^TW_2C)(A - G_lC) + (A - G_lC)^T(D^{\perp}W_1D^{\perp^T} + C^TW_2C)$$
$$= D^{\perp}W_1D^{\perp^T}A + C^TW_2CA - YC + \left(D^{\perp}W_1D^{\perp^T}A + C^TW_2CA - YC\right)^T$$

De lo anterior, se puede observar que el primer término del lado izquierdo de la igualdad es similar al primero del lado derecho, por lo que estos términos son igualados:

$$(D^{\perp}W_1D^{\perp^T} + C^TW_2C)(A - G_lC) = D^{\perp}W_1D^{\perp^T}A + C^TW_2CA - YC$$
$$D^{\perp}W_1D^{\perp^T}A + C^TW_2CA - (\boldsymbol{D^{\perp}W_1D^{\perp^T}} + \boldsymbol{C^TW_2C})\boldsymbol{G_lC} = D^{\perp}W_1D^{\perp^T}A + C^TW_2CA - \boldsymbol{YC}$$

Igualando los términos en negrita se obtiene la ***ganancia lineal de retroalimentación del observador***:

$$-\left(D^{\perp}W_1D^{\perp^T} + C^TW_2C\right)G_lC = -YC$$
$$G_l = \left(D^{\perp}W_1D^{\perp^T} + C^TW_2C\right)^{-1}Y \quad (3.106)$$

Asimismo, se reemplaza $P = D^{\perp}W_1D^{\perp^T} + C^TW_2C$ en (3.103b) a fin de obtener la matriz F:

$$D^T\left(D^{\perp}W_1D^{\perp^T} + C^TW_2C\right) = FC$$
$$D^TD^{\perp}W_1D^{\perp^T} + D^TC^TW_2C = FC$$
$$F = D^TC^TW_2CC^{-1}$$
$$F = D^TC^TW_2 \quad (3.107)$$

Las igualdades (3.105), (3.106) y (3.107) son las soluciones de (3.103), a partir de las restricciones planteadas (3.104), por lo que todo esto debe cumplirse a fin de satisfacer la condición de existencia del observador de Walcott-Zak. Por otro lado, para el ***diseño de la ganancia no lineal de retroalimentación*** (G_n) se parte del modelo lineal incierto descrito en (2.23) y de la ecuación del observador descrita en (3.102) para formar el sistema de error de lazo cerrado del observador:

$$\dot{x}(t) - \dot{\hat{x}}(t) = Ax(t) + Bu(t) + D\xi(t,y,u) - (A\hat{x}(t) + Bu(t) + G_l(\mathrm{y(t)} - \mathrm{C}\hat{x}(t)) - G_n\mathrm{v})$$
$$\dot{x}(t) - \dot{\hat{x}}(t) = A(x(t) - \hat{x}(t)) + Bu(t) + D\xi(t,y,u) - Bu(t) - G_l(\mathrm{y(t)} - \mathrm{C}\hat{x}(t)) + G_n\mathrm{v})$$
$$\dot{x}(t) - \dot{\hat{x}}(t) = A(x(t) - \hat{x}(t)) + D\xi(t,y,u) - G_lC(x(t) - \hat{x}(t)) + G_n\mathrm{v}$$
$$\dot{e}(t) = Ae(t) + D\xi(t,y,u) - G_lC\mathrm{e(t)} + G_n\mathrm{v}$$
$$\dot{e}(t) = A_oe(t) + D\xi(t,y,u) + G_n\mathrm{v} \quad (3.108)$$

La solución hallada anteriormente correspondiente a la ganancia lineal de retroalimentación G_l permite reducir el sistema de error de lazo cerrado al que figura en (3.108), por lo que la ganancia no lineal de retroalimentación deberá ofrecer la robustez suficiente ante incertidumbres estructuradas no lineales no contempladas en el modelamiento, siendo estas canalizadas mediante el término $D\xi(t,y,u)$. Por tal motivo, para que (3.108) sea insensible al término $D\xi(t,y,u)$, se debe cumplir que exista una matriz Γ tal que se cumpla:

$$D = G_n\Gamma \quad (3.109)$$

De acuerdo a lo descrito en Xiang et al. [64], existe un sistema de estructura variable de modos deslizantes que permite que el sistema de error de lazo cerrado (3.108) sea asintóticamente estable si y solo si existe una matriz positiva semi definida P y un escalar positivo σ tal que:

1) $A_O{}^TP + PA_0 - \sigma PG_nG_n{}^TP < 0$ (3.110a)
2) $G_n{}^TP = C$ (3.110b)

De lo anteriormente descrito, la ganancia no lineal de retroalimentación es igual a $G_n = Q^{-1}C^T$ (3.111) si se satisface lo siguiente:

1) $QA_0 + A_O{}^TQ - \sigma C^TC < 0$ (3.112a)
2) $D^TQ = \Gamma C$ (3.112b)

Si se asume que el término incierto $\xi(t,y,u)$ es acotado por $\|\xi(t,y,u)\| \leq \alpha(t,y,u)$, donde $\alpha(t,y,u)$ es una función escalar conocida, si se escoge $G_n = Q^{-1}C^T$ la función discontinua del

sistema de estructura variable que hace que el sistema de error de lazo cerrado (3.108) sea asintóticamente estable es igual a:

$$v(t) = -\sigma e_y - (\|\Gamma\| \alpha(t,y,u) + n)\frac{e_y}{\|e_y\|} \tag{3.113}$$

Donde n es un escalar positivo que permite calibrar que tanto el error inicial del sistema se encuentra contenido en la superficie deslizante y e_y es el vector de errores entre las variables de estado estimadas y las variables de estado medibles.

Asimismo, aplicando la restricción $\|D^T C^T W_4\| < k$, donde k es un escalar que debe ser elegido por el diseñador, permite que el costo de la inyección no lineal $G_n v(t)$ sea minimizado cuando se asegure los modos deslizantes del sistema. Al respecto, la siguiente desigualdad se cumple:

$$\|v(t)\| < \|G_n\|(\sigma\|e_y(0)\| + k\alpha(t,y,u) + n) \tag{3.114}$$

Donde $e_y(0)$ es el error de estados medibles inicial y se debe cumplir $Q^{-1} < \|G_n\| I$ para fines de acotación de la norma de la ganancia de retroalimentación no lineal $\|G_n\|$[18]. Posteriormente, de las desigualdades (3.112) Xiang et al. [64] propone que se cumplan las siguientes condiciones para la matriz Q:

1) $Q = D^{\perp} W_3 D^{\perp T} + C^T W_4 C > 0$ (3.115a)
2) $QA_0 + A_O{}^T Q - \sigma C^T C < 0$ (3.115b)
3) $\Gamma = D^T C^T W_4$ (3.115c)

Mediante el teorema 3.8 (Método directo de Lyapunov) se define la siguiente función candidata de Lyapunov:

$$V(e(t)) = \frac{1}{2} e(t)^T Q e(t) \tag{3.116}$$

Se verifica que $V(e) > 0 \; \forall \; e(t) \in D - \{0\}, V(0) = 0$ dado que Q es una matriz simétrica positiva definida. Asimismo, se define la derivada de la función candidata de Lyapunov descrita en (3.116) de la siguiente forma:

$$\dot{V}(e(t)) = \dot{e}(t)^T Q e(t) + e(t)^T Q \dot{e}(t) < 0 \tag{3.117}$$

Reemplazando (3.108) en (3.117) se tiene:

$$\dot{V}(e(t)) = \dot{e}(t)^T Q e(t) + e(t)^T Q \dot{e}(t) < 0$$
$$\dot{e}(t) = A_o e(t) + D\xi(t,y,u) + G_n \mathrm{v(t)}$$

$$\dot{V}\big(e(t)\big) = \big(A_o e(t) + D\xi(t,y,u) + G_n \mathrm{v(t)}\big)^T Q e(t) + e(t)^T Q\big(A_o e(t) + D\xi(t,y,u) + G_n \mathrm{v(t)}\big) < 0$$
$$\dot{V}\big(e(t)\big) = \big(e(t)^T A_o{}^T + \xi(t,y,u)^T D^T + v(t)^T G_n{}^T\big) Q e(t) + e(t)^T Q A_o e(t) + e(t)^T Q D\xi(t,y,u) + e(t)^T Q G_n \mathrm{v(t)} < 0$$
$$\dot{V}\big(e(t)\big) = e(t)^T A_o{}^T Q e(t) + \xi(t,y,u)^T D^T Q e(t) + v(t)^T G_n{}^T Q e(t) + e(t)^T Q A_o e(t) + e(t)^T Q D\xi(t,y,u) + e(t)^T Q G_n \mathrm{v(t)} < 0$$
$$\dot{V}\big(e(t)\big) = e(t)^T\big(A_o{}^T Q + Q A_o\big)e(t) + \xi(t,y,u)^T D^T Q e(t) + v(t)^T G_n{}^T Q e(t) + e(t)^T Q D\xi(t,y,u) + e(t)^T Q G_n \mathrm{v(t)}$$
$$\dot{V}\big(e(t)\big) = e(t)^T\big(A_o{}^T Q + Q A_o\big)e(t) + (D\xi(t,y,u))^T Q e(t) + \big(G_n v(t)\big)^T Q e(t) + e(t)^T Q D\xi(t,y,u) + e(t)^T Q G_n \mathrm{v(t)} < 0$$
$$\dot{V}\big(e(t)\big) = e(t)^T\big(A_o{}^T Q + Q A_o\big)e(t) + e(t)^T Q D\xi(t,y,u) + e(t)^T Q G_n v(t) + e(t)^T Q D\xi(t,y,u) + e(t)^T Q G_n \mathrm{v(t)} < 0$$
$$\dot{V}\big(e(t)\big) = e(t)^T\big(A_o{}^T Q + Q A_o\big)e(t) + 2e(t)^T Q D\xi(t,y,u) + 2e(t)^T Q G_n v(t) < 0$$

[18] Debe ser designada como la función objetivo a ser minimizada para el problema de optimización que se planteará posteriormente en este apartado.

Reemplazando (3.109) en (3.118):

$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + 2e(t)^T Q G_n \Gamma\xi(t,y,u) + 2e(t)^T Q G_n v(t) < 0 \qquad (3.118)$$

Reemplazando (3.111) en (3.118):

$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + 2e(t)^T Q Q^{-1} C^T \Gamma\xi(t,y,u) + 2e(t)^T Q Q^{-1} C^T v(t) < 0$$
$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + 2e(t)^T Q Q^{-1} C^T \Gamma\xi(t,y,u) + 2e(t)^T Q Q^{-1} C^T v(t) < 0$$

$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + 2e(t)^T C^T \Gamma\xi(t,y,u) + 2e(t)^T C^T v(t) < 0 \qquad (3.119)$$

Reemplazando la función discontinua del sistema de estructura variable (3.113) en (3.119) se tiene:

$$v(t) = -\sigma e_y - (\|\Gamma\|\,\alpha(t,y,u) + n)\frac{e_y}{\|e_y\|}$$
$$v(t) = -\sigma Ce(t) - (\|\Gamma\|\,\alpha(t,y,u) + n)\frac{Ce(t)}{\|C(e(t))\|}$$

$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + 2e(t)^T C^T \Gamma\xi(t,y,u) + 2e(t)^T C^T v(t) < 0$$
$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + 2e(t)^T C^T(\Gamma\xi(t,y,u) + v(t)) < 0$$
$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + 2e(t)^T C^T(\Gamma\xi(t,y,u) + v(t)) < 0$$
$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + 2e(t)^T C^T(\Gamma\xi(t,y,u) - \sigma Ce(t)\ldots$$
$$\ldots -(\|\Gamma\|\,\alpha(t,y,u) + n)\frac{Ce(t)}{\|C(e(t))\|}) < 0 \qquad (3.120)$$

Dado que el término incierto $\xi(t,y,u)$ es acotado por $\|\xi(t,y,u)\| \leq \alpha(t,y,u)$, el término $\Gamma\xi(t,y,u)$ de la ecuación (3.120) es cancelado por el término de la función discontinua $-(\|\Gamma\|\,\alpha(t,y,u) + n)\frac{Ce(t)}{\|C(e(t))\|}$, quedando solo el término de retroalimentación lineal $-\sigma Ce(t)$[19]. Después de esta operación, la ecuación (3.120) queda de la siguiente forma:

$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + 2e(t)^T C^T(-\sigma Ce(t)) < 0$$
$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o)e(t) + e(t)^T(-2\sigma C^T C)e(t) < 0$$

$$\dot{V}(e(t)) = e(t)^T(A_o{}^T Q + QA_o - 2\sigma C^T C)e(t) < 0 \qquad (3.121)$$

Si la igualdad (1) de (3.31) es negativa semi definida, el término $A_o{}^T Q + QA_o - 2\sigma C^T C$ es también negativo semi definido. Por ello, queda demostrado que el sistema de error de lazo cerrado planteado es asintóticamente estable.

3.6.2 Diseño del observador de Walcott-Zak por síntesis LQG/LMI

De los fundamentos teóricos anteriormente expuestos, mediante el programa del apéndice 2.4 se sintetiza la ganancia lineal del observador de Walcott-Zak de acuerdo al siguiente detalle:

1) **Síntesis de la ganancia lineal de retroalimentación G_l:** A partir de la ecuación (3.106), se utiliza un diseño óptimo lineal cuadrático gaussiano o LQG (Linear Quadratic Gaussian, por sus siglas en inglés), resolviendo la siguiente ecuación algebraica de Riccati (ARE):

$$AQ_{ARE} + Q_{ARE}A^T - Q_{ARE}C^T Y^{-1} C Q_{ARE} + W = 0 \qquad (3.122)$$

[19] Según [57], este término permite forzar al sistema de error hacia el colector deslizante, compensando por los errores provenientes de las variables de estado no medibles. Cabe resaltar que existen varios diseños de sistemas de estructura variable que no incluyen este término.

Donde W es una matriz positiva semi definida que representa la matriz de pesos de performance del observador, elegida a criterio del diseñador, y Y es una matriz positiva semi definida que representa la covarianza del ruido del sensor. Cabe resaltar que este diseño de la ganancia lineal de retroalimentación permite tomar en consideración el ruido de medición, que para el diseño del observador de modos deslizantes de Edwards-Spurgeon no fue tomado en cuenta.

Esto permite deducir la ganancia lineal de retroalimentación de la siguiente forma:

$$G_l = Q_{ARE} C^T Y^{-1} \tag{3.123}$$

Para el caso particular en estudio, se eligió una matriz de pesos y una matriz de covarianza del ruido de medición inicial del radar $W = diag([0.1\ 0.1\ 0.8\ 0.1\ 0.1\ 0.8\ 0.1\ 0.1\ 0.8])$[20] y $Y = diag([7.98e-05\ 4.56e-05\ 1.39e-04\ 4.56e-05\ 5.18e-06\ 4.56e-05])$, respectivamente. Por tanto, se obtuvo la ganancia lineal del observador:

$$G_l = \begin{bmatrix} 0.2238 & 0.0034 & 0 & 0 & 0 & 0 \\ 0.0034 & 0.2249 & 0 & 0 & 0 & 0 \\ 0.0137 & 0.5871 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1385 & 0.0028 & 0 & 0 \\ 0 & 0 & 0.0028 & 0.2249 & 0 & 0 \\ 0 & 0 & 0.0106 & 0.5871 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1842 & 0.0031 \\ 0 & 0 & 0 & 0 & 0.0031 & 0.2249 \\ 0 & 0 & 0 & 0 & 0.0124 & 0.5871 \end{bmatrix}$$

2) Síntesis de la ganancia no lineal de retroalimentación G_n: Al hacer $A_o = A - G_l C$, escoger el escalar $k > 0$ y el escalar $\sigma > 0$ se debe resolver el problema de optimización planteado en Xiang et al. [64] por medio de desigualdades lineales matriciales (LMI), utilizando el toolbox de Control Robusto de MATLAB®:

$\min(\|G_n\|), sujeto\ a$:

1) $\begin{bmatrix} -kI & D^T C^T W_4 \\ W_4 C D & -kI \end{bmatrix} < 0$ (3.124a)

2) $\begin{bmatrix} -\|G_n\| I & I \\ I & -Q \end{bmatrix} < 0$ (3.124b)

3) $Q = D^{\perp} W_3 D^{\perp T} + C^T W_4 C$ (3.124c)

4) $Q A_0 + {A_O}^T Q - \sigma C^T C < 0$ (3.124d)

Como se puede observar (3.124c) es una igualdad y en Xiang et al. [64] no se especifica como trabajar con una igualdad para resolver las LMI en el toolbox de MATLAB® o en YAMILP. Sin embargo, Boukas [65] propone que las igualdades matriciales pueden ser aproximadas como desigualdades matriciales de la siguiente manera:

$$\left(Q - D^{\perp} W_3 D^{\perp T} + C^T W_4 C\right)^T \left(Q - D^{\perp} W_3 D^{\perp T} + C^T W_4 C\right) \leq bI$$

donde b es un escalar. De lo anteriormente expuesto, (3.124c) puede ser reemplazado por la siguiente LMI:

$$\begin{bmatrix} -bI & \left(Q - D^{\perp} W_3 D^{\perp T} + C^T W_4 C\right)^T \\ Q - D^{\perp} W_3 D^{\perp T} + C^T W_4 C & -I \end{bmatrix} < 0$$

Finalmente, el problema de optimización para sintetizar la ganancia no lineal de retroalimentación G_n queda planteado de la siguiente forma:

[20] Estos polos son adecuados para efectuar la simulación con ruido, sin embargo, para la simulación sin ruido es mejor sintetizar las ganancias utilizando la matriz de pesos $W = diag([0.8\ 0.8\ 0.8\ 0.8\ 0.8\ 0.8\ 0.8\ 0.8\ 0.8])$

$\min(\|G_n\|), sujeto\ a$:

1) $\begin{bmatrix} -kI & D^T C^T W_4 \\ W_4 CD & -kI \end{bmatrix} < 0$ (3.125a)

2) $\begin{bmatrix} -\|G_n\| I & I \\ I & -Q \end{bmatrix} < 0$ (3.125b)

3) $\begin{bmatrix} -bI & \left(Q - D^{\perp} W_3 D^{\perp T} + C^T W_4 C\right)^T \\ Q - D^{\perp} W_3 D^{\perp T} + C^T W_4 C & -I \end{bmatrix} < 0$ (3.125c)

4) $QA_0 + {A_O}^T Q - \sigma C^T C < 0$ (3.125d)

Al resolver el problema de optimización anterior, se obtienen las variables óptimas $\|G_n\|_{opt}$, $W_{3_{opt}}$, $W_{4_{opt}}$, b_{opt} y Q_{opt}. Esta última es utilizada para hallar la ganancia no lineal de retroalimentación G_n por medio de (3.111). Asimismo, al efectuar el procedimiento anterior y establecer los parámetros de la función discontinua del observador $\sigma = [1000\ 0.1\ 0.1\ 0.01\ 100\ 0.01]^T$ y $n = 1$, se obtuvieron las siguientes matrices por medio del toolbox LMI de control robusto de MATLAB®:

$$A_o = \begin{bmatrix} 0.7762 & 0.0166 & 2e-04 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0034 & 0.7751 & 0.02 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0137 & -0.5871 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.8615 & 0.0172 & 2e-04 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0028 & 0.7751 & 0.02 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0106 & -0.5871 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.8158 & 0.8158 & 2e-04 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.0031 & 0.7751 & 0.02 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.0124 & -0.5871 & 1 \end{bmatrix}$$

$$G_n = \begin{bmatrix} 0.0018 & 3.64e-05 & 0 & 0 & 0 & 0 \\ 3.64e-05 & 0.0035 & 0 & 0 & 0 & 0 \\ 0.0021 & 0.1775 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0019 & 3.61e-05 & 0 & 0 \\ 0 & 0 & 3.61e-05 & 0.0035 & 0 & 0 \\ 0 & 0 & 0.0020 & 0.1775 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0018 & 3.62e-05 \\ 0 & 0 & 0 & 0 & 3.62e-05 & 0.0035 \\ 0 & 0 & 0 & 0 & 0.0021 & 0.1775 \end{bmatrix}$$

3) Diseño de la función discontinua $\boldsymbol{v(t)}$: Se elige un escalar positivo n, se halla la matriz $\Gamma = D^T C^T W_{4_{opt}}$ y se establece la función escalar $\alpha(t, y, u)$. Mediante la función discontinua (3.113) se establece un sistema de estructura variable de modos deslizantes apropiado, donde del parámetro σ brinda mayor robustez, pero también una mayor sensibilidad al ruido de medición. Asimismo, el incremento de n genera una mayor dependencia del error inicial del sistema. Al establecer $\alpha(t, y, u) = |\alpha(t, y, u)| = 1000$, y utilizando la matriz $W_{4_{opt}}$ obtenida en el paso anterior, se logró sintetizar la matriz:

$$\Gamma = \begin{bmatrix} -1.08e-05 & -9.52e-06 & 0 & 0 & 0 & 0 \\ 0 & 0 & -9.71e-06 & -9.58e-06 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1.02e-05 & -9.55e-06 \end{bmatrix}$$

3.6.3 Algoritmo del observador de Walcott-Zak por síntesis LQG/LMI

A continuación, se muestra el pseudocódigo del programa del apéndice 2.5 correspondiente al observador de modos deslizantes de Walcott-Zak (WZSMO), siendo este bastante similar al presentado para el observador de modos deslizantes de Edwards-Spurgeon. Cabe resaltar que la diferencia fundamental es que la síntesis de este observador es realizada por medio del algoritmo de optimización LQG y desigualdades lineales matriciales (LMI):

Algoritmo 3.3: Observador de Walcott-Zak (WZSMO)
Entradas: Matrices $\mathbf{G_l}$, $\mathbf{G_n}$, $\boldsymbol{Z_3}$, $\boldsymbol{A_o}$ y $\mathbf{G_k}$,vectores $\mathbf{y_k}$, $\mathbf{u_k}$ y constantes **g, s0** y muestra actual **k.**
Salidas: Variables de estado estimadas $\boldsymbol{xh_k}$.
Inicio del Programa: Invocado por el programa sim_WZSMO.m
1: Declara e inicializa variable persistente **xhatk, yhatk y ey_log_old**

Algoritmo 3.3: Observador de Walcott-Zak (WZSMO) (continuación)

2: Halla error de estados medibles **ey = yhatk - yk**
3: Concatenar error de estados medibles **ey_log = [ey_log_old ey]**
4: Declara peso inicial del error ***n=1*** y el límite superior de la función ***alfa=1000.***
5: Declara parámetro de robustez ***sig=[1;0.0001;0.0001;0.00001;0.1;0.00001].*s0***
6: Calcular ***v=-sig.*ey -norm(Z3)*alfa.*(ey./(norm(ey_log)))-n*(ey./(norm(ey_log)))***
7: Hallar VE estimadas futuras **xhatk_1 = Fk*xhatk + Gk*uk – Gl*ey + Gn*v**
8: Guardar variables para siguiente iteración
9: Fin de programa

3.6.4 Simulación de la trayectoria del misil

Se desarrolló el programa de MATLAB® del apéndice 2.6, el cual invoca al programa de síntesis de ganancias del apéndice 2.4 y al observador propiamente dicho del apéndice 2.5, siendo su algoritmo detallado a continuación:

Algoritmo 3.4: Simulación de trayectoria del blanco con observador WZSMO

Entradas: Señal de entrada de control $\boldsymbol{\mu_k}$, perturbaciones $\boldsymbol{\xi_k}$ y modelo mixto gaussiano (GMM)
Salidas: Vector de variables de estado (VE) reales del modelo $\boldsymbol{x_k}$, VE medidas del modelo $\boldsymbol{y_k}$, VE estimadas $\boldsymbol{xh_k}$, perturbaciones reconstruidas $\boldsymbol{ph_k}$, errores cartesianos $\boldsymbol{error\_c_k}$, errores polares $\boldsymbol{error\_c_k}$, error **RMSE** y gráficos.
Inicio del programa:
1: Declara tiempo de muestreo $\boldsymbol{T}$, gravedad $\boldsymbol{g}$, constante de mach **M**, sobre aceleración máxima ***load_factor*** tiempo inicial de simulación $\boldsymbol{t_i}$, tiempo final de simulación t_{ff} y número total de muestras $\boldsymbol{nt}$.
2: Declara condiciones iniciales de traqueo al blanco: distancia $\boldsymbol{Ad}$, acimut verdadero $\boldsymbol{Bdn}$, elevación $\boldsymbol{Ed}$, velocidad $\boldsymbol{Vm}$ y rumbo verdadero $\boldsymbol{Rv}$.
3: Convierte posición de coordenadas polares a cartesianas usando $\boldsymbol{Ad, Bdn\ y\ Ed}$
4: Declara el vector de estados inicial $\boldsymbol{x_k(0)}$
5: Declara matrices del espacio de estados $\boldsymbol{F_k}$, $\boldsymbol{G_k}$, $\boldsymbol{D_k}$ y $\boldsymbol{C_k}$.
6: Obtén $\boldsymbol{u_k}$ (invoca a función ***gen_jerk_sea_skimming3.m*** **(programa del apéndice 2.5))**
7: Obtén $\boldsymbol{\xi_k}$ **(invoca** a función ***perturb_1.m*** **(programa del apéndice 2.6))**
8: Cargar ruido angular ***glint_puntos.mat*** **(datos guardados del programa del apéndice 2.7)**
9: Invoca **WZSMO_sintesis.m (programa del apéndice 2.4)** y carga matrices sintetizadas del observador
10: Ingrese "1" para simulación con ruido y "0" para simulación sin ruido.
11: Para tt = ti:T:tff **Hacer**
12: Ejecuta el modelo de transición de estados para obtener $\boldsymbol{x_k}$ futuro.
13: Invoca ***c2p.m*** para convertir $\boldsymbol{x_k}$ a coordenadas polares.
14: Invoca ***yk_noise.m*** para obtener el vector de mediciones y_k
15: Invoca **WZSMO.m** para obtener VE estimadas $\boldsymbol{xh_k}$ y las perturbaciones estimadas **fi**
16: Invoca ***c2p.m*** para convertir $\boldsymbol{xh_k}$ a coordenadas polares.
17: Extrae errores polares, errores cartesianos y RMSE
18: Almacena vectores para gráficas
19: Fin Para
20: Conversión de unidades del vector de estados a Mn, Mach y g's.
21: Graficar
22: Fin de programa

En las figuras 3.16 y 3.18 se observa que el WZSMO, en la ausencia de ruido, posee robustez similar al ESSMO y en las figuras 3.17 y 3.19 el observador WZSMO presenta mejor insensibilidad al ruido.

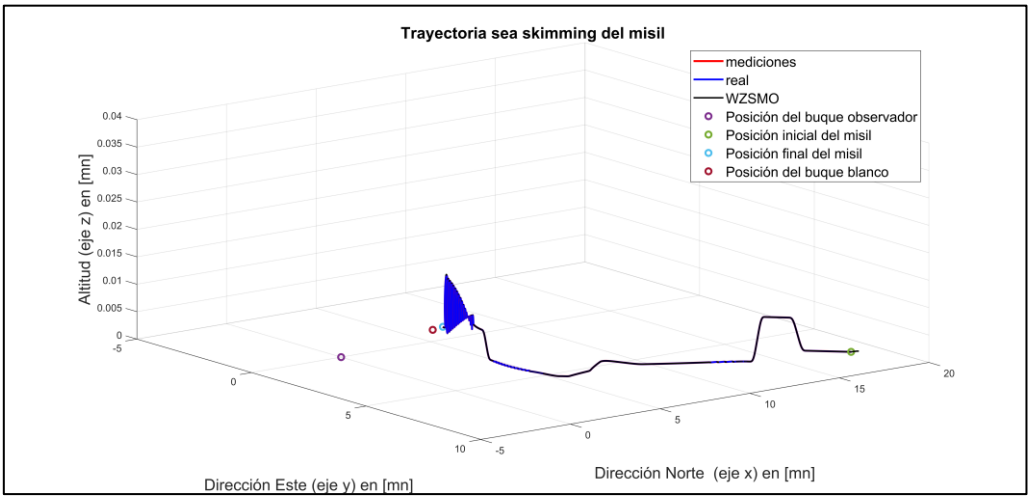

Figura 3.16.: Trayectoria en tres dimensiones sin ruido (WZSMO).
Fuente: Elaboración propia.

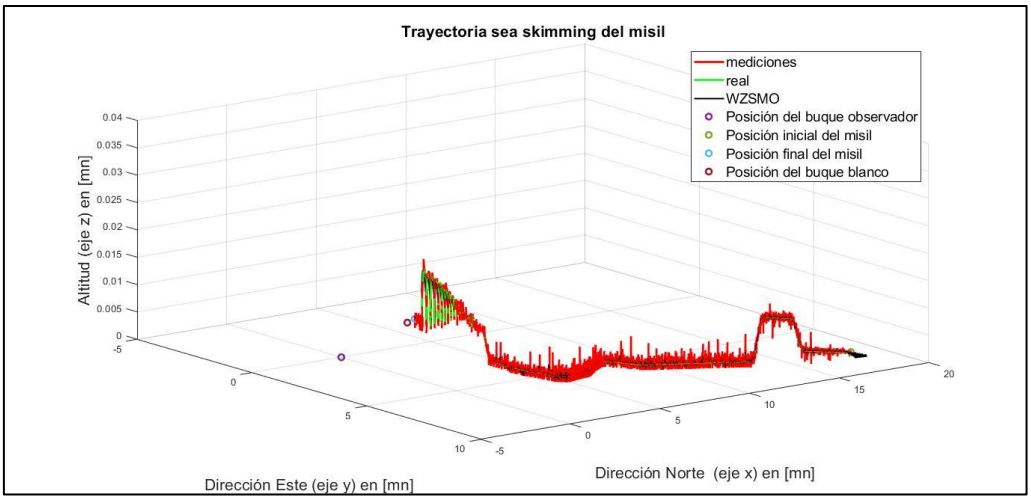

Figura 3.17.: Trayectoria en tres dimensiones con ruido (WZSMO).
Fuente: Elaboración propia.

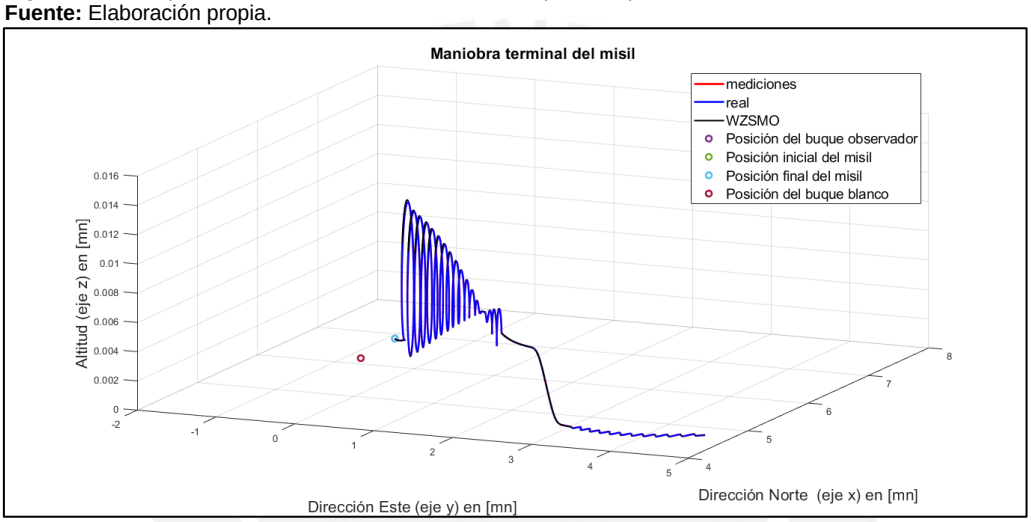

Figura 3.18.: Maniobra terminal del misil sin ruido (WZSMO).
Fuente: Elaboración propia.

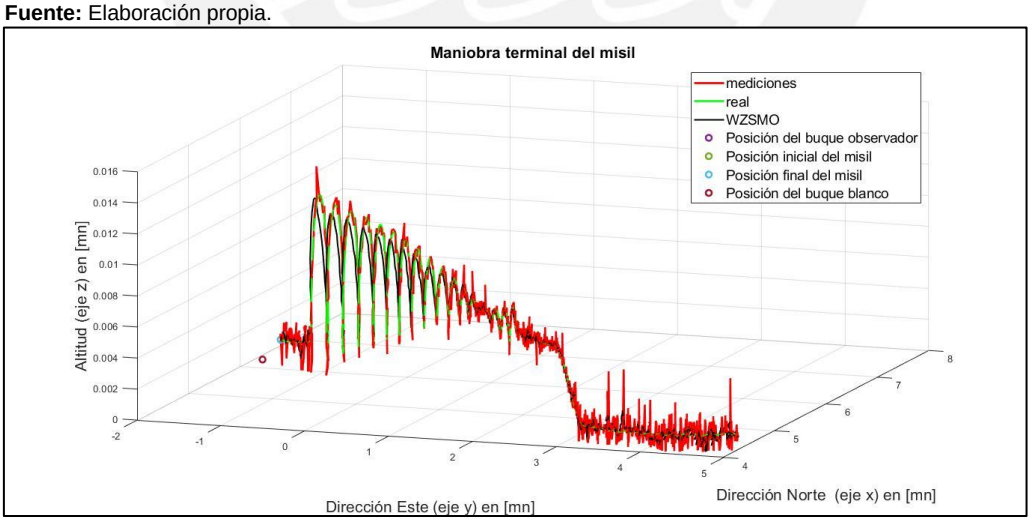

Figura 3.19.: Maniobra terminal del misil con ruido (WZSMO).
Fuente: Elaboración propia.

Por otro lado, en las figuras 3.18, 3.20 y 3.22, que presentan las estimaciones en las coordenadas “x”, “y” y “z”, se constatan los resultados obtenidos en la ausencia de ruido de las figuras 3.14 y 3.16. En adición, en las figuras 3.19, 3.21 y 3.23 se observa que al mejorar la insensibilidad al ruido la robustez del observador decrece en relación al observador ESSMO; esto debido a que en esta simulación se disminuyen los pesos del algoritmo LQG a fin de poder hacer un filtrado mejor del ruido angular. Por tanto, la síntesis de ganancias por LQG ofrece una mejor

insensibilidad al ruido, pero su aplicación en tiempo real es limitada debido a que es necesario conocer de primera mano la covarianza del ruido a filtrar, lo cual en muchos casos prácticos es difícil de saber. Sin embargo, la síntesis propuesta para este observador permite obtener robustez a las perturbaciones mientras se posee cierta capacidad de filtrado del ruido angular.

En la figura 3.24 se observa que los errores de traqueo cartesianos aumentan durante la maniobra terminal del misil, pero se mantienen en todo momento dentro de los límites máximos establecidos en coordenadas polares establecidos (figura 3.25).

Finalmente, el diseño del observador WZSMO no permite efectuar la reconstrucción de las perturbaciones al modelo, a diferencia del observador ESSMO, dado que este no efectúa transformaciones que canalicen las perturbaciones a los canales medibles del modelo.

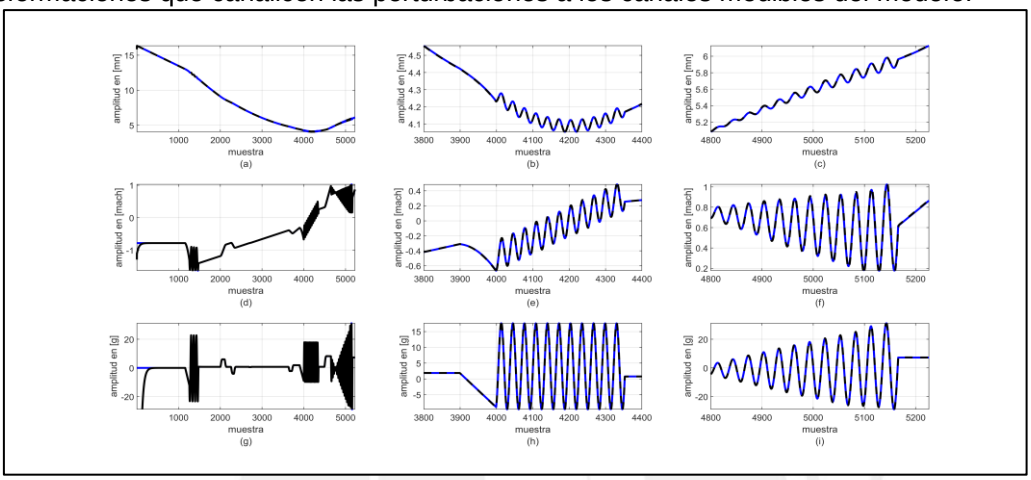

Figura 3.20.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "x" en ausencia de ruido (WZSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [3800 4400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [3800 4400]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

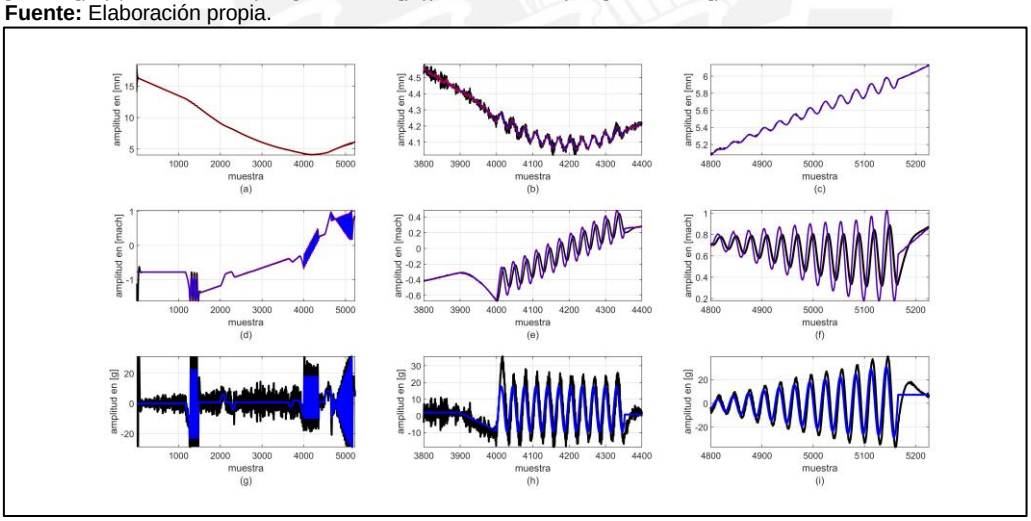

Figura 3.21.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "x" con ruido (WZSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [3800 4400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [3800 4400]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

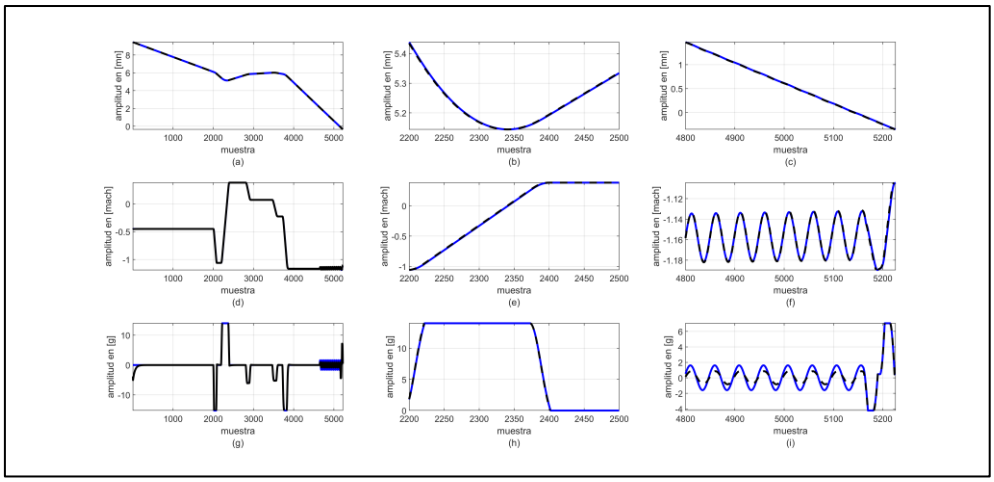

Figura 3.22.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada “y” en la ausencia de ruido (WZSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [2200 2500]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [2200 2500]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [2200 2500]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

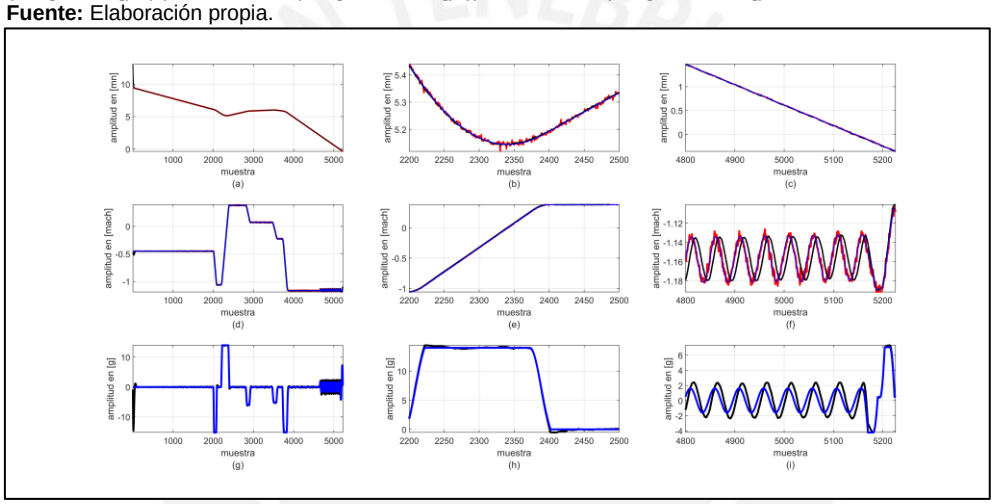

Figura 3.23.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada “y” con ruido (WZSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [2200 2500]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [2200 2500]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [2200 2500]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

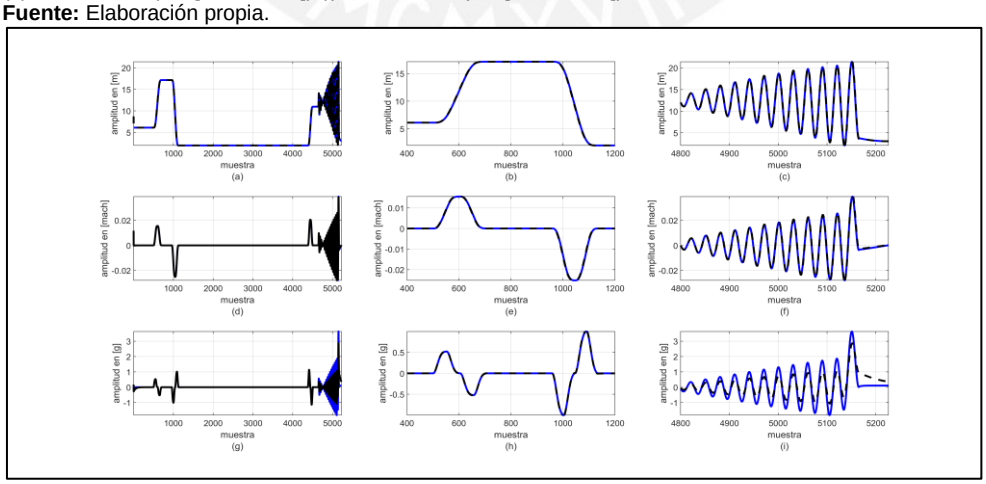

Figura 3.24.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada “z” en la ausencia de ruido (WZSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [400 1200]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [400 1200]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

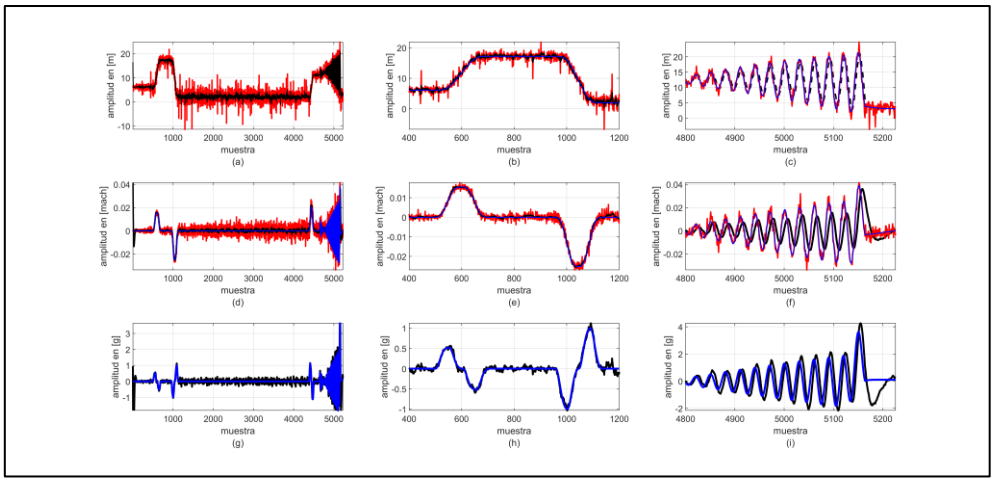

Figura 3.25.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada “z” con ruido (WZSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [400 1200]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [400 1200]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

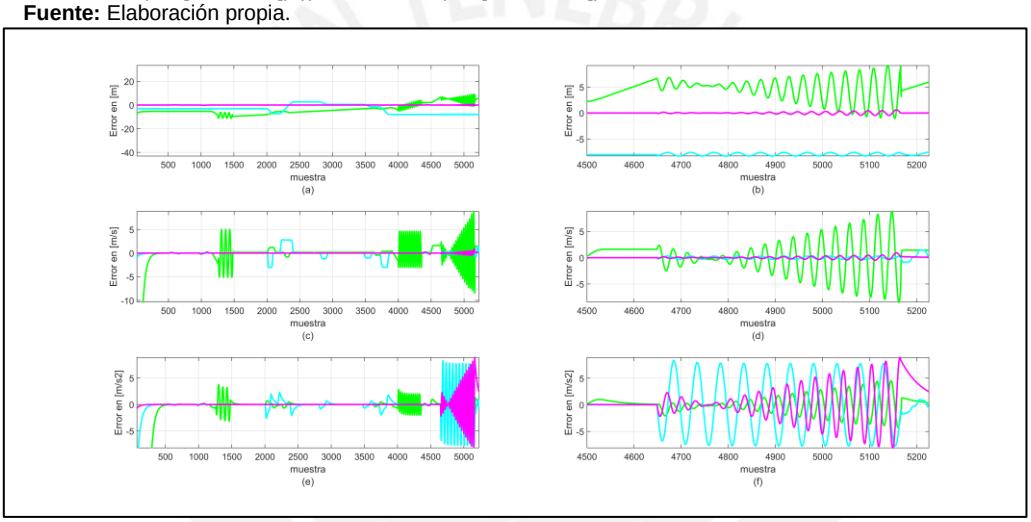

Figura 3.26.: Errores cartesianos (WZSMO). Coordenada “x” (línea verde). Coordenada “y” (línea cyan). Coordenada “z” (línea magenta). (a) Posición k = [0 5226] (b) Posición k = [4800 5226] (c) Velocidad k = [0 5226] (d) Velocidad k = [4800 5226] (e) Aceleración k = [0 5226] (f) Aceleración k = [4800 5226].
Fuente: Elaboración propia.

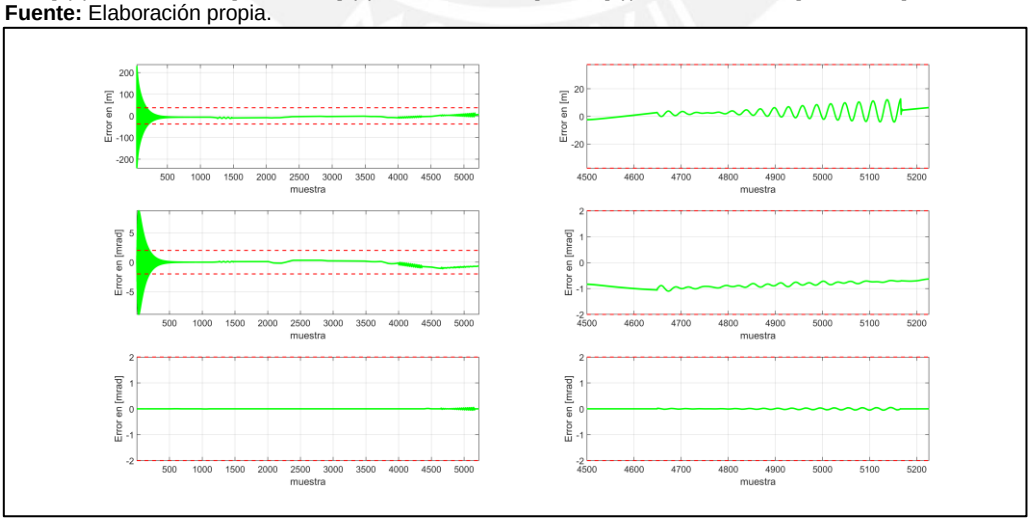

Figura 3.27.: Errores polares (WZSMO). (a) Alcance Ad k = [0 5226] (b) Alcance Ad k = [4800 5226] (c) Azimuth Bdn k = [0 5226] (d) Azimuth Bdn k = [4800 5226] (e) Elevación Ed k = [0 5226] (f) Elevación Ed k = [4800 5226].
Fuente: Elaboración propia.

3.7 Observador de modos deslizantes con algoritmo Super-Twisting de Orden Superior (HOSMO)

El estudio de la existencia y condiciones de estabilidad de los sistemas de modos deslizantes de orden superior al primer orden, según lo expuesto por Utkin [50], es realizado por primera vez por Barbashin y Gerashchenko [66] en la década de los años sesenta. Este concepto se desarrolla complementariamente a los estudios referentes a los *modos deslizantes convencionales*[21] o también conocidos como de primer orden, expuestos en los apartados anteriores, los cuales proveen excelente robustez ante perturbaciones debido a que reaccionan inmediatamente a errores de las variables de estados estimadas respecto a las reales con un esfuerzo enérgico considerable [67]. Sin embargo, según Shtessel et al. [68] las principales desventajas de este tipo de controladores u observadores son la generación del fenómeno de castañeo o "chattering", la desproporcionalidad entre la desviación máxima de la superficie deslizante y el intervalo de tiempo entre muestras de una medición y la restricción de grado relativo uno de la superficie deslizante respecto al grado de las salidas medidas, siendo esto evidenciado durante el diseño y con los resultados obtenidos en los apartados anteriores.

La limitación más grande para el diseño de observadores de modos deslizantes de grado relativo uno es que si, por ejemplo, se requiere estimar la variable de estado de aceleración sería necesario contar con la medición de las variables de estado de posición y velocidad, siendo común que en muchos casos reales de implementación de estos sistemas en plantas o procesos industriales no se cuente con la medición de la velocidad; en el caso particular de radares de control de tiro, como se ha mencionado anteriormente, se puede contar con capacidad de medición de velocidad por efecto Doppler pero esto no se cumple para todos los casos.

Posteriormente, en la década de los años ochenta Emelyanov [69] introduce el concepto de modos deslizantes de orden superior o *Higher Order Sliding Mode* (HOSM por sus siglas en inglés) para el control automático, exponiendo temas relacionados a los conceptos de los modos deslizantes, condiciones de existencia, grado relativo y supresión de castañeo o "chattering". Se conoce que de lo propuesto por Stanislav Emelyanov, autores como Arie Levant, Leonid Fridman, Jaime Moreno, Igor Boiko, Andrey Polyakov, Alexander Pozniak, Hebertt Sira-Ramirez, entre otros, profundizan en esta área de control automático, dando lugar al desarrollo del conocido algoritmo de torsión o Twisting, de super torsión o *Super Twisting* y el *Super Twisting* de orden arbitrario. Asimismo, se conoce que otros autores tales como Shtessel et al. [70] y posteriormente Barth et al. [71] profundizan en algoritmos Super-Twisting para control con propiedades adaptativas aplicado a casos donde el límite superior de la perturbación o incertidumbre estructurada es desconocido. Cabe resaltar que de esta teoría de control se derivan las técnicas de diseño de observadores de estado de modos deslizantes.

En los algoritmos de torsión o *twisting* los estados del sistema convergen en el plano de fase de la superficie deslizante dando giros hasta llegar en un tiempo finito al origen del plano o a un conjunto compacto, tal y como se muestra en la figura 3.24. Cabe resaltar que los giros de la trayectoria permiten demostrar que la señal discontinua de control es suavizada, a diferencia de los modos deslizantes descritos en los apartados anteriores. En el siguiente apartado se expone los fundamentos teóricos de control de los algoritmos basados en el algoritmo de torsión o *twisting* y su aplicación en métodos de observación.

[21] El término "convencional" es introducido por primera vez por Shtessel et al. [68], haciendo referencia a los a los sistemas de modos deslizantes de primer orden. Sin embargo, en el año 2020 Utkin et al. [47] critica esta nomenclatura debido a que no debería existir una diferencia entre los "convencionales" y los algoritmos propuestos en [68].

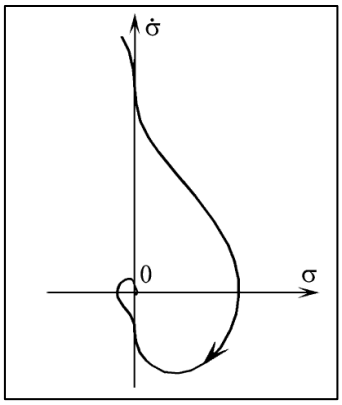

Figura 3.28.: Plano de estados del algoritmo Twisting.
Fuente: Shtessel et al. [70].

3.7.1 Fundamentos teóricos

Sea el sistema dinámico planteado en [68],

$$\dot{x}(t) = a(t,x) + b(t,x)u \quad (3.126a)$$
$$\sigma = \sigma(t,x) \quad (3.126b)$$

Donde x es un vector de dimensión $n\ x\ 1$, u es un escalar, σ es una superficie deslizante con una sola variable de estado medida y las funciones a, b, σ son funciones continuas y suaves. La derivada total respecto al tiempo de σ es igual a $\dot{\sigma} = \sigma'_t + \sigma'_x a + \sigma'_x bu$. Si $\sigma'_x b = 0$, entonces la segunda derivada total respecto al tiempo es:

$$\ddot{\sigma} = \sigma''_{tt} + 2\sigma''_{tx}a + \sigma'_x a'_t + [\sigma''_{xx}(a+bu)]a + \sigma'_x[a'_x(a+bu)]$$
$$\ddot{\sigma} = \sigma''_{tt} + 2\sigma''_{tx}a + \sigma'_x a'_t + \sigma''_{xx}a^2 + (\sigma''_{xx}bu)a + \sigma'_x a'_x a + \sigma'_x a'_x bu \quad (3.127)$$

De la ecuación 3.127 se agrupan los términos para formar las funciones $h(t,x)$ y $g(t,x)$ de la siguiente manera:

$$h(t,x) = \sigma''_{tt} + 2\sigma''_{tx}a + \sigma'_x a'_t + \sigma''_{xx}a^2 + \sigma'_x a'_x a$$
$$g(t,x) = (\sigma''_{xx}b)a + \sigma'_x(a'_x b)$$

Por tanto, la ecuación (3.127) en función a $h(t,x)$ y $g(t,x)$ es igual a:

$$\ddot{\sigma} = h(t,x) + g(t,x)u \quad (3.128)$$

siendo el grado relativo del sistema igual a uno si $\sigma'_x b \neq 0$ e igual a dos si $\sigma'_x b = 0$ y $(\sigma''_{xx}b)a + \sigma'_x(a'_x b) \neq 0$.

Definición 3.14 (Grado relativo [68]): *El sistema 3.126 tiene grado relativo local r en el punto x_0 si:*

(i) La derivada de Lie es $Lg L^k_f h(x) = 0$ para todo x en el vecindario de x_0 y todo $k < r - 1$
(ii) La derivada de Lie es $Lg L^{r-1}_f h(x) \neq 0$

Donde la derivada de Lie $L_f\lambda(x)$ es la derivada de λ respecto a f. Si se diferencia k veces respecto a f, la notación $L^k_f\lambda(x)$ es usada satisfaciendo la recursión $L^k_f\lambda(x) = \frac{\partial L^{k-1}_f\lambda(x)}{\partial x} f(x)$, cumpliéndose $L^0_f\lambda(x) = \lambda(x)$.

Definición 3.15 (Orden deslizante [72]): Si se considera a la variable deslizante como una cantidad dependiente de las variables de estado del sistema que se desvanece cuando arriba al colector deslizante. Por tanto, el grado relativo *q entre la variable deslizante y la variable de entrada de control al sistema puede ser hallado mediante la definición 3.14. Sin embargo, el orden deslizante es definido como el grado relativo entre la variable deslizante y la señal discontinua de control, cumpliéndose que el orden deslizante r no puede ser menor que q.*

Por consiguiente, de acuerdo a lo establecido en [73], cuando el grado relativo es igual a uno la función $\dot{\sigma}$ es discontinua y σ es continua. Por otro lado, cuando el grado relativo es igual a dos la función $\ddot{\sigma}$ es discontinua, pero σ y $\dot{\sigma}$ son continuas. Finalmente, se concluye que un sistema de modos deslizantes convencional (1-sliding) solo puede ser obtenido cuando el grado relativo es igual a uno mientras que un sistema de modos deslizantes de segundo orden (2-sliding) requiere grado relativo igual a dos.

Si se tiene como objetivo de control que la superficie deslizante σ sea igual a cero en un tiempo finito utilizando una función discontinua de control[22] y la función σ tiene derivadas $\dot{\sigma}$ y $\ddot{\sigma}$, se puede suponer que las siguientes desigualdades cumplen por lo menos localmente para un sistema de grado relativo dos:

$$0 < K_m \leq g(t,x) \leq K_M \quad \text{(3.129a)}$$

$$|h(t,x)| \leq C \quad \text{(3.129b)}$$

Para todo $K_m, K_M, C > 0$

De lo anteriormente expuesto, se conoce que no existe ningún controlador de la forma $u = f(\sigma, \dot{\sigma})$ que pueda cumplir con el objetivo de control debido a que ese tipo de control, que asegura $\sigma = 0$, debe satisfacer la igualdad $\ddot{\sigma} = 0$ y por consiguiente cumplirse $f(0,0) = -h(t,x)/g(t,x)$ cuando $\sigma = \dot{\sigma} = 0$. Si $\ddot{\sigma} = c + ku$, donde $K_m \leq k \leq K_M$ y $|c| \leq C$, y existe una incertidumbre en el sistema, la función $f(0,0) \neq -c/k$ y por consiguiente será necesario un modo deslizante de segundo orden o *Second-order Sliding Mode* (SOSM por sus siglas en inglés). Por otro lado, si las ecuaciones (3.129) son válidas globalmente, de (3.128) y (3.129) se deduce que:

$$\ddot{\sigma} = [-C, C] + [K_m, K_M]u \quad \text{(3.130)}$$

Por tanto, de (3.130) se deduce que el objetivo de control es *diseñar una ley de control* $u = f(\sigma, \dot{\sigma})$*, tal que las trayectorias de* u *y (3.130) converjan al origen del plano* $\sigma, \dot{\sigma}$ *en un tiempo finito, por medio de trayectorias suaves.* Esto es, que el control discontinuo actúe en una derivada de orden mayor que la variable deslizante a fin de atenuar el castañeo o *chattering.*

3.7.1.1 Algoritmo de torsión o *Twisting* (TA)

El Dr. Arie Levant en Levantovksy [74] propone la primera solución al problema antes mencionado por medio de un algoritmo de modos deslizantes de segundo orden llamado algoritmo de torsión o *Twisting*, el cual puede ser aplicado para sistemas con grado relativo uno y dos. El algoritmo *Twisting* implementa la siguiente ley de control:

$$z(t) = -(r_1 sign(\sigma) + r_2 sign(\dot{\sigma})) \quad \text{(3.131)}$$

donde se deben cumplir las siguientes condiciones:

$$r_1 > r_2 > 0 \quad \text{(3.132a)}$$

$$(r_1 + r_2) K_m - C > (r_1 - r_2) K_M + C \quad \text{(3.132b)}$$

$$(r_1 - r_2) K_m > C \quad \text{(3.133c)}$$

Donde K_m , K_M y C son elegidas a criterio del diseñador. Este tipo de algoritmo de control, según lo señalado en [68] requiere contar con la medición de $\dot{\sigma}$ para la implementación, por lo que normalmente se utilizan observadores para poder estimar la variable $\dot{\sigma}$. Asimismo, cabe resaltar que la ley de control se mantiene discontinua para sistemas de grado relativo dos.

3.7.1.2 Algoritmo de Super Torsión o *Super twisting* (STA)

Posteriormente, en el año de 1998 se propone el algoritmo de control *Super twisting*, el cual asegura la convergencia de la superficie deslizante en un tiempo finito sin necesidad de contar

[22] Cabe resaltar que esta ley de control, por ser discontinua, no cumple las características de continuidad local de Lipschitz, por lo que las soluciones o trayectorias del sistema deben ser entendidas desde la perspectiva de Fillipov.

con la derivada $\dot{\sigma}$, según lo expuesto por Levant [75], siendo aplicado a sistemas de segundo orden. El algoritmo *Super twisting* implementa la siguiente ley de control:

$$z(t) = z_1(t) + z_2(t) \quad (3.133a)$$
$$z_1(t) = -r_1|\sigma|^{\frac{1}{2}} sign(\sigma) \quad (3.133b)$$
$$z_2(t) = -r_2 sign(\sigma) \quad (3.133c)$$

Donde σ es una superficie deslizante en base a la medición disponible. Cabe resaltar que $que\ r_1, r_2 > 0$ y adicionalmente se debe cumplir,

$$r_1 > 1.5\sqrt{\delta} \quad (3.134a)$$
$$r_2 > 1.1\delta \quad (3.134b)$$

Siendo δ es el límite máximo de la incertidumbre o perturbación acotada. Como se puede observar que el término de compensación $z_2(t)$ es un término discontinuo de la variable deslizante σ, mientras el término $z_1(t)$ es una función continua de la variable deslizante. Esto permite que se cuente con un término continuo de compensación durante la fase de llegada al colector deslizante [75].

Este algoritmo de control puede ser aplicado para la implementación de un observador de estados, el cual consta de una copia del modelo del sistema y los dos términos de compensación *Super twisting* $z_1(t)$ y $z_2(t)$. A continuación, se muestra en la columna de la izquierda el modelo ejemplo de un sistema doble integrador y en la columna derecha el observador *Super twisting* correspondiente:

Modelo del sistema (ejemplo)	Observador Super-Twisting (STASMO)
$\dot{x}_1(t) = x_2(t)$	$\dot{\hat{x}}_1(t) = \hat{x}_2(t) + z_1(t)$
$\dot{x}_2(t) = u(t) + \rho$	$\dot{\hat{x}}_2(t) = u(t) + z_2(t)$

Dado que el algoritmo *Super twisting* solo es aplicable a sistemas de segundo orden, resultó necesario expandir este algoritmo para que sea compatible con sistemas de orden superior al segundo.

3.7.1.3 Algoritmo Super-Twisting de orden arbitrario o generalizado (n-STA)

Desde la perspectiva de Fillipov, cualquier ecuación diferencial discontinua de la forma $\dot{x} = v(x)$, donde $x \in \mathbb{R}^n$ y v es una función vectorial medible y localmente acotada, es reemplazable por una inclusión diferencial del tipo $\dot{x} \in V(x)$. Una inclusión diferencial $V(x)$ es una generalización del concepto de ecuación diferencial ordinaria y desde la perspectiva de Fillipov representa el cierre convexo del conjunto de todos los posibles límites de $v(y)$ para $y \to x$, siendo el conjunto de valores de y puntos de continuidad de la función $v(x)$. Por tanto, cualquier solución es definida como una función continua $x(t)$, satisfaciendo la inclusión diferencial en casi cualquier punto [76]. Si el sistema de la forma $\dot{x} = v(x)$, en el sentido de Fillipov y según lo explicado por Levant [76], cumple con:

1) Poseer una superficie deslizante suave del tipo σ.
2) Poseer una ley de control discontinua de retro alimentación.
3) Las sucesivas derivadas de la superficie deslizante σ son funciones continuas en términos de las variables de estado del sistema, que sus derivadas guardan la relación $\sigma = \dot{\sigma} = \ddot{\sigma} = \sigma^{r-1} = 0$, siendo no vacías y consistentes con trayectorias de Fillipov.
4) El conjunto de velocidades permitidas de Fillipov contiene más de un vector.

Entonces, se considera que el sistema muestra un movimiento de modos deslizantes de grado r o también conocido como *r-sliding mode*. Por tanto, el algoritmo de control Super-Twisting de orden arbitrario (n-STA) propuesto por Kamal et al. [77] aplicado a sistemas de control cuenta con la siguiente estructura generalizada:

Modelo de la planta o proceso con controlador Super-Twisting n-STA

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = x_3(t)$$
$$\vdots$$
$$\dot{x}_{n-1}(t) = -k_1|\phi_{n-2}|^{\frac{1}{2}}sign(\phi_{n-2}) + x_n(t)$$
$$\dot{x}_n(t) = -k_n sign(\phi_{n-2}) + \rho$$

Donde $x_1(t),..,x_n(t)$ son las variables de estado del sistema y ρ es una perturbación acotada que debe cumplir $|\rho| \leq \Delta$, donde Δ es el valor escalar máximo o límite superior de la perturbación. Por otro lado, el término ϕ_{n-2} es hallado de la siguiente manera:

1) $R_{1,r-1} = |x_1(t)|^{\frac{r}{r+1}}$, donde r es el grado relativo del sistema respecto a $x_1(t)$
2) $R_{i,r-1} = ||x_1(t)|^{r_1} + |x_2(t)|^{r_2} + \cdots + |x_{i-2}(t)|^{r_{i-2}}|^{q_i}$, donde $i = 2,3,..,(r-1)$ $r_1, r_2, ..., r_{i-2}$ y q_i son parámetros de diseño en base a la homogeneidad ponderada[23] del estado x_{i+1}
3) $S_{0,r-1} = x_1$
4) $S_{1,r-1} = x_2 + k_2 R_{1,r-1} sign(x_1)$
5) $S_{i,r-1} = x_{i+1} + k_{i+1} R_{i,r-1} sign(S_{i-1,r-1})$, donde $i = 2,3,..,(r-1)$
6) $\phi_{n-2} = S_{r-1,r-1}$

3.7.1.4 Algoritmo Super-Twisting de orden superior (HOSMO)

Utilizando el concepto de modos deslizantes de orden superior o *Higher Order Sliding Modes* expuesto en [76,78], se añade en el algoritmo de control *Super twisting de orden arbitrario* el término $z_n = x_n + \rho$, el cual representa la integral del término discontinuo $\dot{z}_n = -k_n sign(\phi_{n-2}) + \dot{\rho}$; la adición de la integral de un término discontinuo eleva el orden del algoritmo, por ello la denominación "orden superior" y permite reconstruir la perturbación del sistema al incluir el control una señal que compensa la perturbación o incertidumbre estructurada adecuadamente. Asimismo, permite obtener derivadas de la variable deslizante atenuando el castañeo o *chattering*. A continuación, se muestra la estructura de un ejemplo de modelo de la planta o proceso con los términos de compensación del controlador *Super twisting de Orden Superior*:

Modelo de la planta o proceso con controlador Super-Twisting de Orden Superior

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = x_3(t)$$
$$\vdots$$
$$\dot{x}_{n-1}(t) = -k_1|\phi_{n-2}|^{\frac{1}{2}}sign(\phi_{n-2}) + z_n(t)$$
$$\dot{z}_n(t) = -k_n sign(\phi_{n-2}) + \dot{\rho}$$

A partir del controlador *Super twisting de orden superior* planteado, en [79] se propone el siguiente observador para la planta modelo de doble integrador, el cual consta de una copia del modelo del sistema y tres términos de compensación de super torsión que ofrecen robustez ante perturbaciones o incertidumbres estructuradas:

[23] Sea $\alpha \geq$ un real positivo y el vector de pesos $p = (p_1, p_2, p_3, \dots, p_n)$ con $p_i > 0$, se define el operador "dilatación" definido como $\Lambda_\alpha = diag(\alpha^{p_i})$. Por ello, según lo descrito en [47], se dice que una función escalar $V: \mathbb{R}^n \rightarrow \mathbb{R}$ es homogénea con pesos $p = (p_1, p_2, p_3, \dots, p_n)$ y grado de homogeneidad $\delta \in \mathbb{R}$ si se satisface lo siguiente: $V(\Lambda_\alpha x) = \alpha^\delta V(x)$ para todo $x \in \mathbb{R}^n$ y $\alpha \geq 0$. Asimismo, un campo vectorial $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ es homogéneo con pesos $p = (p_1, p_2, p_3, \dots, p_n)$ y grado de homogeneidad $\delta \in \mathbb{R}$ si se satisface la siguiente relación $f(\Lambda_\alpha x) = \alpha^\delta \Lambda_\alpha f(x)$ para todo $x \in \mathbb{R}^n$ y $\alpha \geq 0$.

Modelo del sistema (ejemplo)	Observador de orden superior (HOSMO)
$\dot{x}_1(t) = x_2(t)$ $\dot{x}_2(t) = u(t) + \rho_1$	$\dot{\hat{x}}_1(t) = \hat{x}_2(t) + z_1(t)$ $\dot{\hat{x}}_2(t) = u(t) + z_2(t) + x_3(t)$ $\dot{\hat{x}}_3(t) = z_3(t)$

Siendo los términos de corrección son $z_1(t) = k_1|e_1|^{\frac{2}{3}}sign(e_1)$, $z_2(t) = k_2|e_1|^{\frac{1}{3}}sign(e_1)$ y $z_3(t) = k_3 sign(e_1)$, $e_1 = x_1(t) - \hat{x}_1(t)$, $e_2 = x_2(t) - \hat{x}_2(t)$, $e_3 = -\hat{x}_3(t) + \rho_1$ y k_1,k_2 y k_3 son constantes positivas. Asimismo, la dinámica del error del observador es la siguiente:

$$\dot{e}_1(t) = -k_1|e_1|^{\frac{2}{3}}sign(e_1) + e_2$$
$$\dot{e}_2(t) = -k_2|e_1|^{\frac{1}{3}}sign(e_1) + e_3$$
$$\dot{e}_3(t) = -k_3 sign(e_1) + \dot{\rho}_1$$

Del sistema de error planteado se concluye que en un tiempo finito $e_1, e_2\ y\ e_3$ convergerán a cero seleccionando las ganancias apropiadas k_1,k_2 y k_3, y por consiguiente $x_1(t) = \hat{x}_1(t)$, $x_2(t) = \hat{x}_2(t)$ y $\hat{x}_3(t) = \rho_1$. Cabe resaltar que el algoritmo *Super twisting de orden superior* permite estimar la perturbación al sistema, pero por un método diferente que el de control equivalente utilizado por el observador de Edwards-Spurgeon. A continuación, se plantea el diseño del observador Super twisting de orden superior.

3.7.2 Diseño del observador Super twisting de orden superior (HOSMO).

Sea el sistema dinámico lineal incierto en espacio de estados del sistema discreto planteado en (2.23),

$$x_{k+1} = F_k x_k + G_k u_k + D_k \xi_k + w_k$$
$$y_k = H_k{}^{24} x_k + v_k$$

El espacio de estados se coloca en forma de ecuaciones con la finalidad de efectuar una copia del modelo dinámico del sistema. A continuación, se muestran las ecuaciones de la coordenada "x", siendo idénticas las ecuaciones de las coordenadas "y" y "z".

$$x_{1_{k+1}} = x_{1_k} + Tx_{2_k} + T^2 x_{3_k} + \frac{T^3}{6} u_{1_k} + 75e4\frac{T^3}{6} \xi_{1_k}$$
$$x_{2_{k+1}} = x_{2_k} + Tx_{3_k} + \frac{T^2}{2} u_{1_k} + 5e3\frac{T^2}{2} \xi_{1_k}$$
$$x_{3_{k+1}} = x_{3_k} + Tu_{1_k} + 1e2T\xi_{1_k}$$
$$y_{1_k} = x_{1_k} + v_k$$

Donde x_{1_k}, x_{2_k} y x_{3_k} son las variables de estado actuales de posición, velocidad y aceleración en la coordenada "x", respectivamente, $x_{1_{k+1}}$, $x_{2_{k+1}}$ y $x_{3_{k+1}}$ son las variables de estado futuras de posición, velocidad y aceleración en la coordenada "x", respectivamente, u_{1_k} es la señal de control en la coordenada "x", ξ_{1_k} es la perturbación acotada en la coordenada "x", y_{1_k} es la posición medida del blanco en la coordenada "x" y T es el tiempo de muestreo. A partir de las ecuaciones anteriores y tomando como referencia la estructura del observador de modos deslizantes de orden superior planteado para el modelo de doble integrador, se define el observador de modos deslizantes de orden superior (HOSMO) de la coordenada "x", siendo los observadores de las coordenadas "y" e "z" idénticos.

[24] Considerar la matriz de mediciones (2.28), donde solo es posible medir las variables de estado de posición. Para el diseño del presente observador se considerará que la medición de velocidad no se encuentra disponible

HOSMO coordenada "x"

$$\hat{x}_{1_{k+1}} = \hat{x}_{1_k} + T\hat{x}_{2_k} + \frac{T^2}{2}\hat{x}_{3_k} + \frac{T^3}{6}u_{1_k} + 75e4\frac{T^3}{6}\hat{\xi}_{1_k} + z_{1x_k}$$
$$\hat{x}_{2_{k+1}} = \hat{x}_{2_k} + T\hat{x}_{3_k} + \frac{T^2}{2}u_{1_k} + 5e3\frac{T^2}{2}\hat{\xi}_{1_k} + z_{2x_k}$$
$$\hat{x}_{3_{k+1}} = \hat{x}_{3_k} + Tu_{1_k} + 1e2T\hat{\xi}_{1_k} + z_{3x_k}$$
$$\hat{\xi}_{1_{k+1}} = \hat{\xi}_{1_k} + Tz_{4x_k}$$

donde, los términos de corrección son $z_{1x_k} = k_1|\sigma_1|^{\frac{4}{5}}sign(\sigma_1)$, $z_{2x_k} = k_2|\sigma_1|^{\frac{3}{4}}sign(\sigma_1)$, $z_{3x_k} = k_3|\sigma_1|^{\frac{1}{2}}sign(\sigma_1)$ y $z_{4x_k} = k_4 sign(\sigma_1)$, $\sigma_1 = y_{1_k} - \hat{x}_{1_k}$ es la superficie deslizante definida como el error de posición del blanco en la coordenada "x" y $\hat{\xi}_{1_k}$ es la estimación de la perturbación ξ_{1_k} en la coordenada "x". A continuación, se presentan las ganancias utilizadas para el observador HOSMO las cuales fueron sintonizadas mediante simulaciones:

Observador HOSMO	K_1	K_2	K_3	K_4
Coordenada x	1.7155	2.6136	7.1166	7.5758
Coordenada y	1.0835	1.2151	1.6175	3.9872
Coordenada z	1.1737	1.3885	1.8732	13.9553

Tabla 3.1.: Ganancias sintonizadas de los observadores HOSMO.
Fuente: Elaboración propia.

3.7.3 Algoritmo del observador Super twisting de orden superior (HOSMO).

A continuación, se presenta el algoritmo del observador de modos deslizantes Super-Twisting de orden Superior, el cual es implementado en el programa del apéndice 2.7.

Algoritmo 3.5: Observador Super Twisting de Orden Superior (HOSMO)
Entradas: Matrices $\boldsymbol{F_k}$, $\mathbf{G_k}$ *y* $\mathbf{D_k}$, vectores $\mathbf{y_k}$, $\mathbf{u_k}$ y tiempo de muestreo ***T***
Salidas: VE estimadas $\boldsymbol{xh_k}$, perturbaciones estimadas $\boldsymbol{ph_k}$ *y* colectores deslizantes $\boldsymbol{sig_k}$
Inicio del Programa: Invocado por el programa sim_HOSMO.m
1: Declara e inicializa variables persistentes de estados estimados en coordenada x **(x1hatk, x2hatk, x3hatk),** coordenada y **(x4hatk, x5hatk, x6hatk)** y coordenada z **(x7hatk, x8hatk, x9hatk).**
2: Declara e inicializa variables persistentes de perturbación estimada en coordenada x **(p1hatk),** coordenada y **(p2hatk)** y coordenada z **(p3hatk)**
3: Define los errores de posición **sigx=x1hatk-yk(1,1)**, **sigy=x4hatk-yk(3,1)** y **sigz=x7hatk- yk(5,1)** y velocidad **sigdx=x2hatk-yk(2,1)**, **sigdy=x5hatk-yk(4,1)** y **sigdz=x8hatk-yk(6,1)** (errores de velocidad solo para gráficos de plano de estados, no intervienen en el algoritmo de estimación).
4: Declara ganancias homogéneas del observador de la coordenada "x" **(k1x, k2x, k3x y k4x)** coordenada "y" **(k1y, k2y, k3y y k4y)** y coordenada "z"**(k1z, k2z, k3z y k4z).**
5: Calcula términos de compensación en coordenada "x" **(z1xk, z2xk, z3xk y z4xk),** coordenada "y" **(z1yk, z2yk, z3yk y z4yk)** y coordenada "z" **(z1zk, z2zk, z3zk y z4zk)**.
6: Calcular VE estimadas futuras **x1hatk_1,x2hatk_2,… (ver código).**
7: Integra términos **z4xk, z4yk y z4zk y** estima perturbaciones **p1hatk, p2hatk** y **p3hatk.**
8: Guardar variables para siguiente iteración.
9: Fin de programa

3.7.4 Simulación de la trayectoria del misil

El algoritmo 3.6 define la lógica de funcionamiento del programa del apéndice 2.8:

Algoritmo 3.6: Simulación de trayectoria del blanco con observador HOSMO
Entradas: Señal de entrada de control $\boldsymbol{\mu_k}$, perturbaciones $\boldsymbol{\xi_k}$ y modelo mixto gaussiano (GMM)
Salidas: Vector de variables de estado (VE) reales del modelo $\boldsymbol{x_k}$, VE medidas del modelo $\boldsymbol{y_k}$, VE estimadas $\boldsymbol{xh_k}$, perturbaciones estimadas $\boldsymbol{ph_k}$, errores cartesianos $\boldsymbol{error\_c_k}$, errores polares $\boldsymbol{error\_c_k}$, error **RMSE** y gráficos.
Inicio del programa:
1: Declara tiempo de muestreo ***T***, gravedad $\boldsymbol{g}$, constante de mach **M**, sobre aceleración máxima ***load_factor*** tiempo inicial de simulación $\boldsymbol{t_i}$, tiempo final de simulación t_{ff} y número total de muestras $\boldsymbol{nt}$.
2: Declara condiciones iniciales de traqueo al blanco: distancia $\boldsymbol{Ad}$, acimut verdadero $\boldsymbol{Bdn}$, elevación $\boldsymbol{Ed}$, velocidad $\boldsymbol{Vm}$ y rumbo verdadero $\boldsymbol{Rv}$.
3: Convierte posición de coordenadas polares a cartesianas usando $\boldsymbol{Ad, Bdn\ y\ Ed}$

Algoritmo 3.6: Simulación de trayectoria del blanco con observador HOSMO (continuación)

4: Declara el vector de estados inicial $\boldsymbol{x_k(0)}$
5: Declara matrices del espacio de estados $\boldsymbol{F_k}$, $\boldsymbol{G_k}$, $\boldsymbol{D_k}$ y $\boldsymbol{C_k}$.
6: Obtén $\boldsymbol{u_k}$ (invoca a función ***gen_jerk_sea_skimming3.m*** **(programa del apéndice 2.5))**
7: Obtén $\boldsymbol{\xi_k}$ **(invoca** a función ***perturb_1.m*** **(programa del apéndice 2.6))**
8: Cargar ruido angular ***glint_puntos.mat*** **(datos guardados del programa del apéndice 2.7)**
9: Ingrese "1" para simulación con ruido y "0" para simulación sin ruido.
10: Para tt = ti:T:tff **Hacer**
11: Ejecuta el modelo de transición de estados para obtener $\boldsymbol{x_k}$ futuro.
12: Invoca ***c2p.m*** para convertir $\boldsymbol{x_k}$ a coordenadas polares.
13: Invoca ***yk_noise.m*** para obtener el vector de mediciones y_k
14: Invoca **HOSMO.m** para obtener VE estimadas $\boldsymbol{xh_k}$ y las perturbaciones estimadas $\boldsymbol{uh_k}$
15: Invoca ***c2p.m*** para convertir $\boldsymbol{xh_k}$ a coordenadas polares.
16: Extrae errores polares, errores cartesianos y RMSE
17: Almacena vectores para gráficas
18: Fin Para
19: Conversión de unidades del vector de estados a Mn, Mach y g's.
20: Graficar
21: Fin de programa

Al simular la trayectoria del misil se pudo apreciar que el observador HOSMO en la ausencia de ruido posee buena robustez y exactitud en la estimación de las variables de estado, sin embargo, se observa que su convergencia es más lenta que la de los observadores de Edward-Spurgeon y Walcott-Zak; esto debido a que efectúa una super torsión hacia el origen o conjunto compacto en la fase final de convergencia a las variables de estado reales. Asimismo, en la simulación con ruido se observó que es sensible al ruido angular, siendo esta sensibilidad menor que la del observador de Edward-Spurgeon (ESSMO) pero mayor que la del observador de Walcott-Zak (WZSMO). Cabe resaltar que es una ventaja preponderante respecto a los observadores anteriormente estudiados que el observador super twisting de orden superior (HOSMO) posea la capacidad de obtener las variables de estado no medidas de velocidad y aceleración solo a partir de la medición de posición.

A continuación, se presentan los resultados de la trayectoria completa del misil sin ruido (figura 3.29), con ruido (figura 3.30), maniobra terminal sin ruido (3.31) y con ruido (3.32). Asimismo, se presentan los resultados de la trayectoria estimada con y sin ruido de la coordenada "x" en las figuras 3.33 y 3.34, respectivamente. Posteriormente, se presentan los resultados de la trayectoria estimada del misil sin ruido de las coordenadas "y" y "z" en las figuras 3.35 y 3.36, dado que los resultados con ruido fueron similares a los de la coordenada "x", siendo estos no satisfactorios en términos de sensibilidad al ruido. Asimismo, en estas figuras se observa que la velocidad de convergencia a la variable de estado real durante los cambios de rumbo y maniobra terminal es más lenta que los observadores anteriormente expuestos.

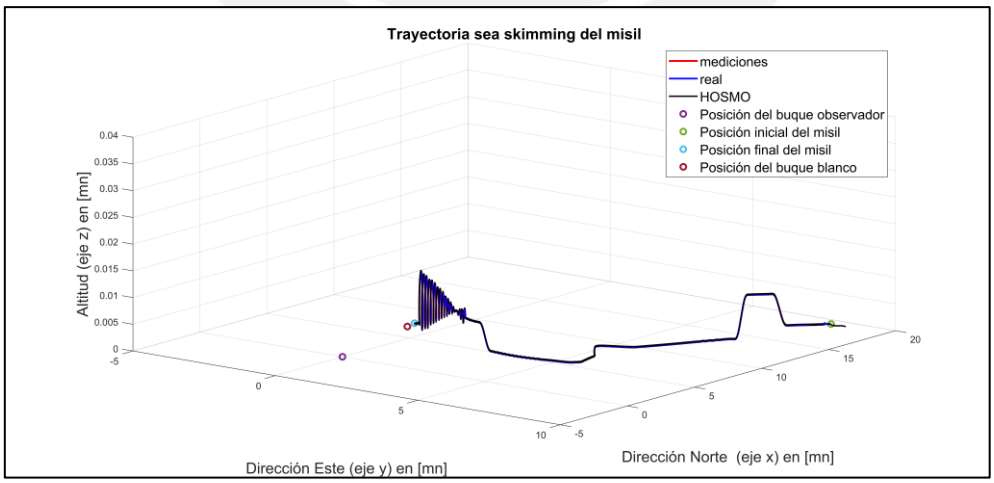

Figura 3.29.: Trayectoria en tres dimensiones sin ruido (HOSMO).
Fuente: Elaboración propia.

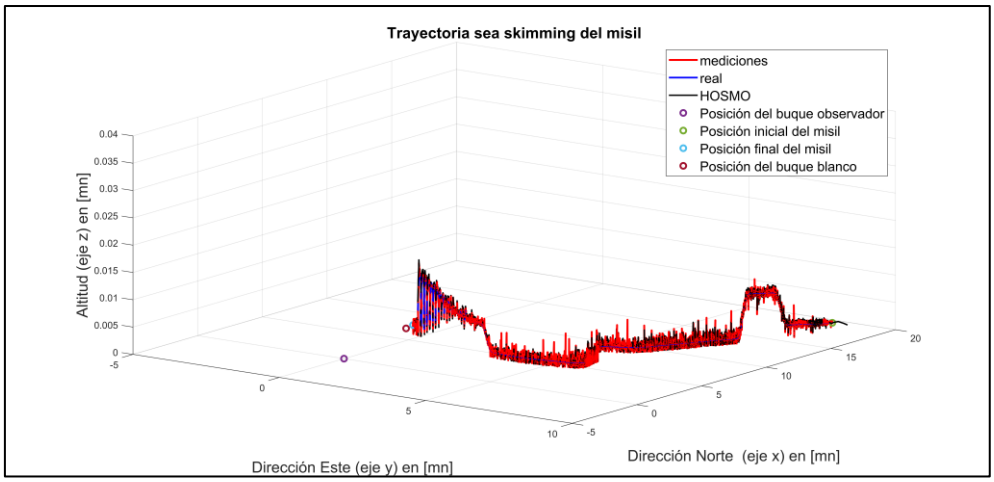

Figura 3.30.: Trayectoria en tres dimensiones con ruido (HOSMO).
Fuente: Elaboración propia.

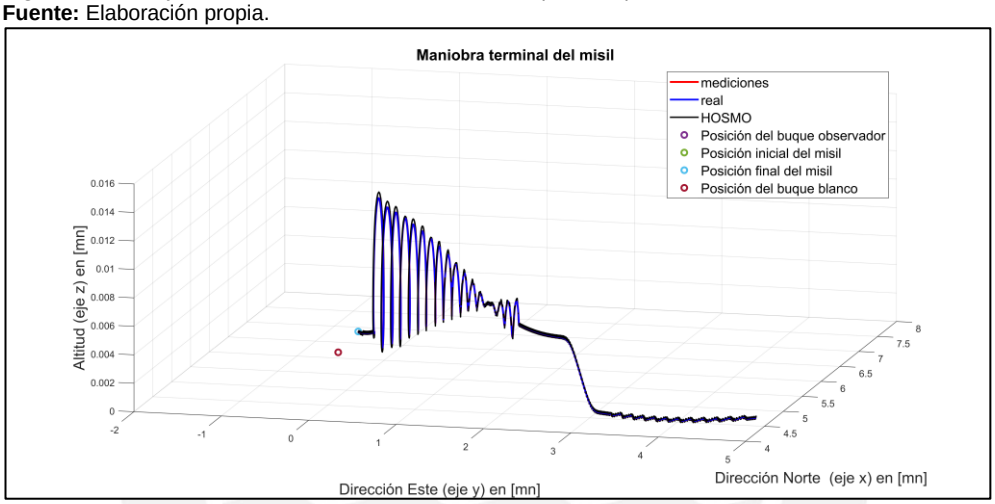

Figura 3.31.: Maniobra terminal del misil sin ruido (HOSMO).
Fuente: Elaboración propia.

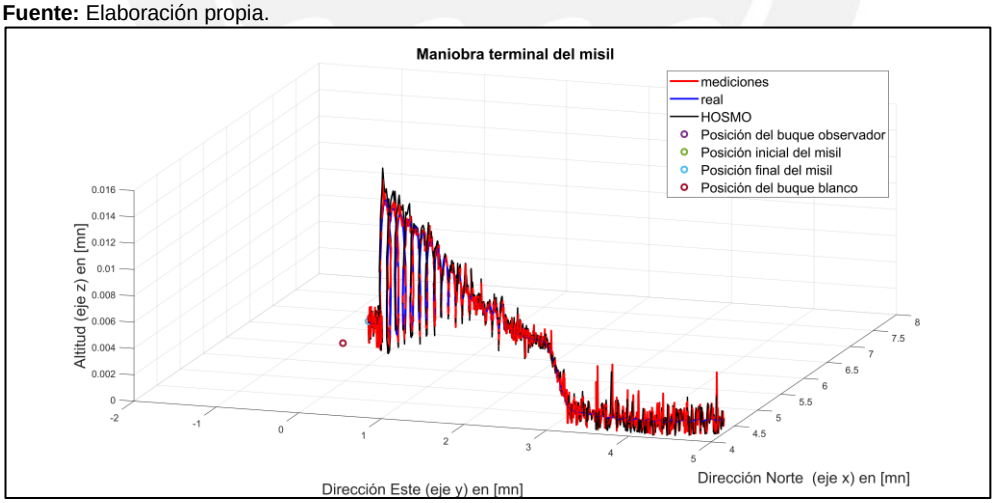

Figura 3.32.: Maniobra terminal del misil con ruido (HOSMO).
Fuente: Elaboración propia.

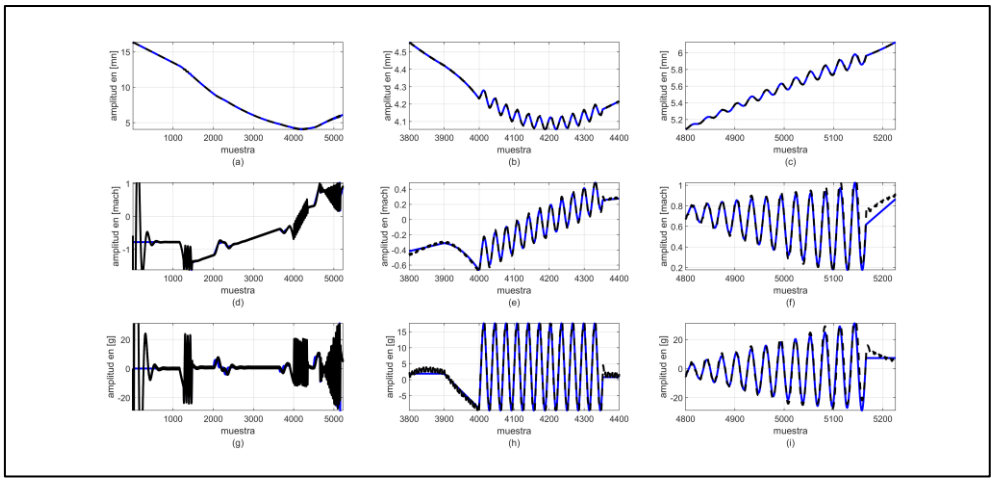

Figura 3.33.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "x" en ausencia de ruido (HOSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [3800 4400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [3800 4400]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

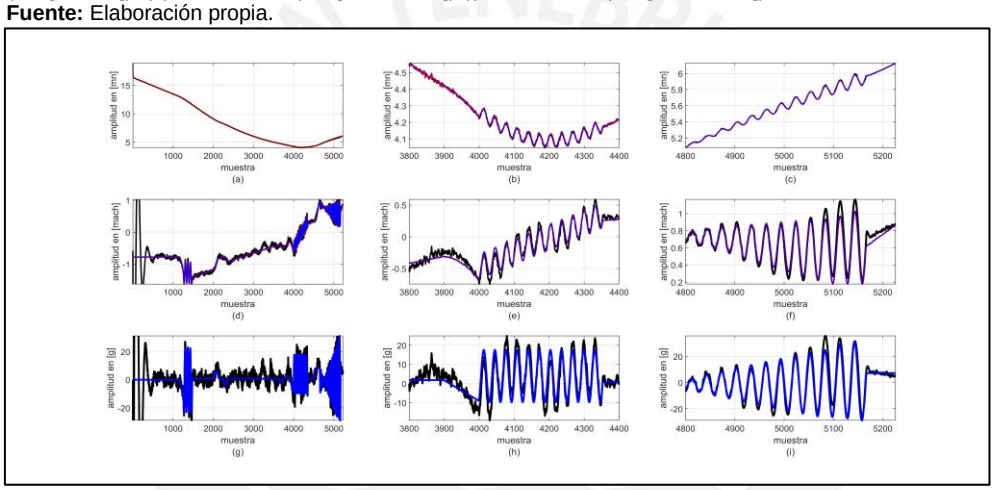

Figura 3.34.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "x" con ruido (HOSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [3800 4400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [3800 4400]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

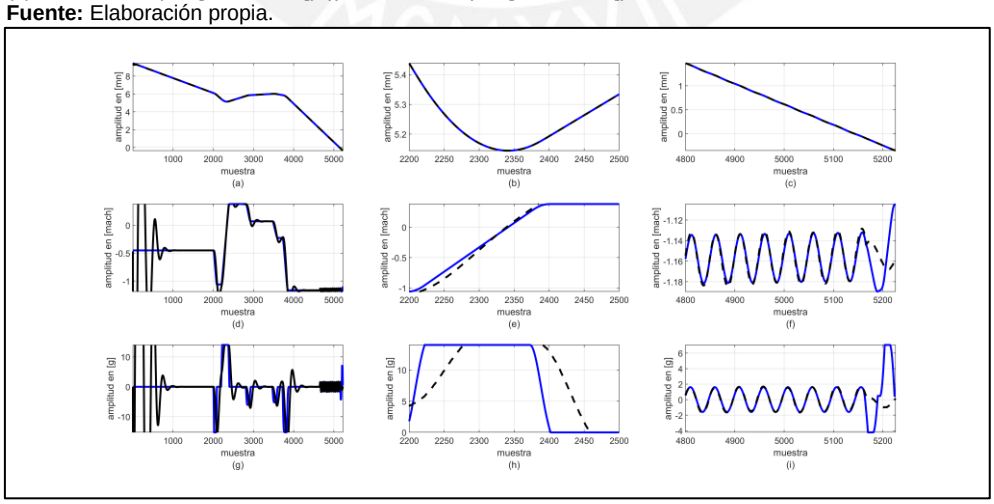

Figura 3.35.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "y" en la ausencia de ruido (WZSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [2200 2500]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [2200 2500]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [2200 2500]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

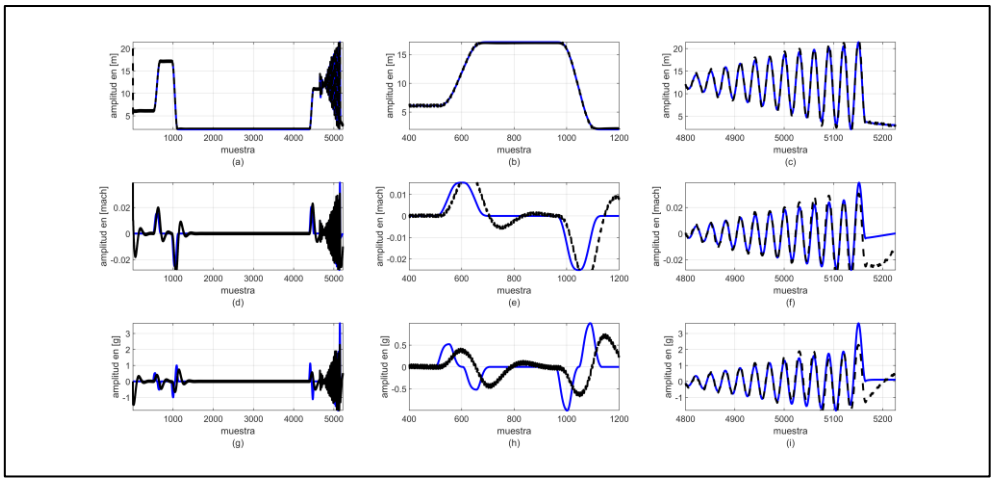

Figura 3.36.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "z" en la ausencia de ruido (HOSMO). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [400 1200]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [400 1200]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

En la figura 3.37 se presentan los errores cartesianos sin ruido, observándose una convergencia más lenta al origen de los errores y un aumento de estos durante la maniobra terminal, siendo los errores mayores que los observadores anteriormente expuestos. Sin embargo, en las figuras 3.38 y 3.39 se constata que los errores polares se mantienen dentro de los parámetros máximos establecidos con y sin ruido angular, respectivamente.

Asimismo, en la figura 3.40 se aprecia una buena estimación de las perturbaciones, siendo evidente una forma de onda triangular de las estimaciones, característica del término integral implementado en el algoritmo. Sin embargo, las estimaciones efectuadas por HOSMO son menos exactas que las reconstrucciones efectuadas por ESSMO. Cabe resaltar que en la presencia de ruido angular se pierde por completo la reconstrucción de las perturbaciones, demostrando mayor sensibilidad al ruido que el observador de Walcott-Zak. En la figura 3.41, se muestra la convergencia de super torsión que muestran las variables deslizantes en el plano de estados, siendo más prominente la super torsión en la coordenada "y".

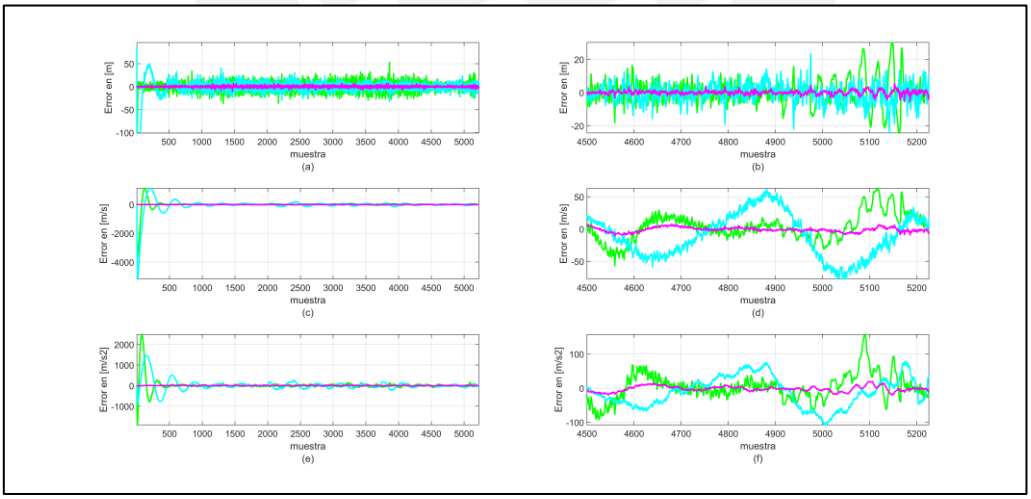

Figura 3.37.: Errores cartesianos (HOSMO). Coordenada "x" (línea verde). Coordenada "y" (línea cyan). Coordenada "z" (línea magenta). (a) Posición k = [0 5226] (b) Posición k = [4800 5226] (c) Velocidad k = [0 5226] (d) Velocidad k = [4800 5226] (e) Aceleración k = [0 5226] (f) Aceleración k = [4800 5226].
Fuente: Elaboración propia.

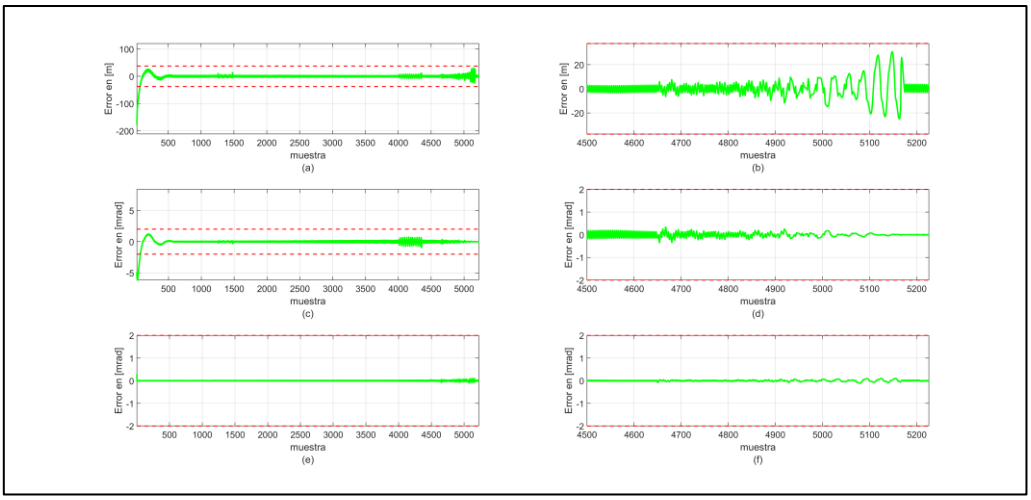

Figura 3.38.: Errores polares sin ruido (HOSMO). (a) Alcance Ad k = [0 5226] (b) Alcance Ad k = [4800 5226] (c) Azimuth Bdn k = [0 5226] (d) Azimuth Bdn k = [4800 5226] (e) Elevación Ed k = [0 5226] (f) Elevación Ed k = [4800 5226].
Fuente: Elaboración propia.

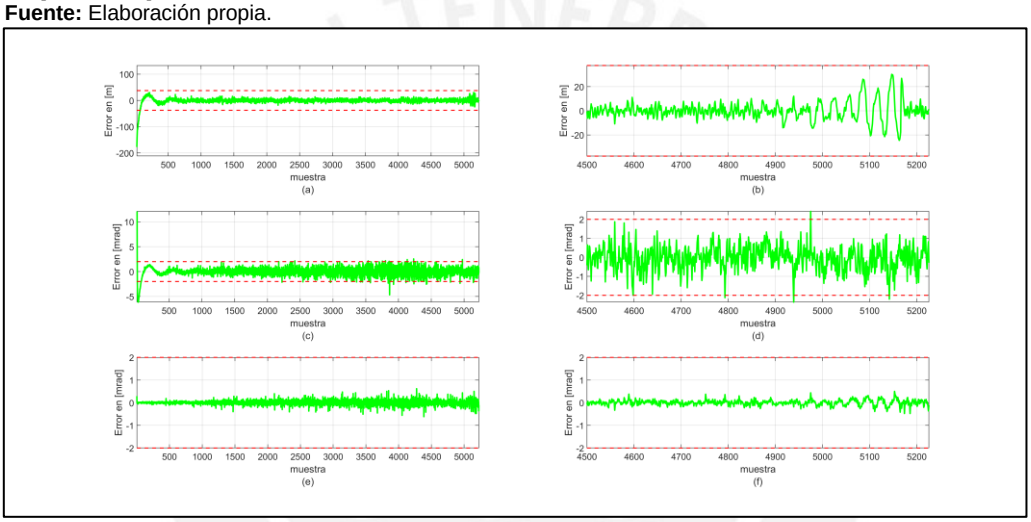

Figura 3.39.: Errores polares con ruido (HOSMO). (a) Alcance Ad k = [0 5226] (b) Alcance Ad k = [4800 5226] (c) Azimuth Bdn k = [0 5226] (d) Azimuth Bdn k = [4800 5226] (e) Elevación Ed k = [0 5226] (f) Elevación Ed k = [4800 5226].
Fuente: Elaboración propia.

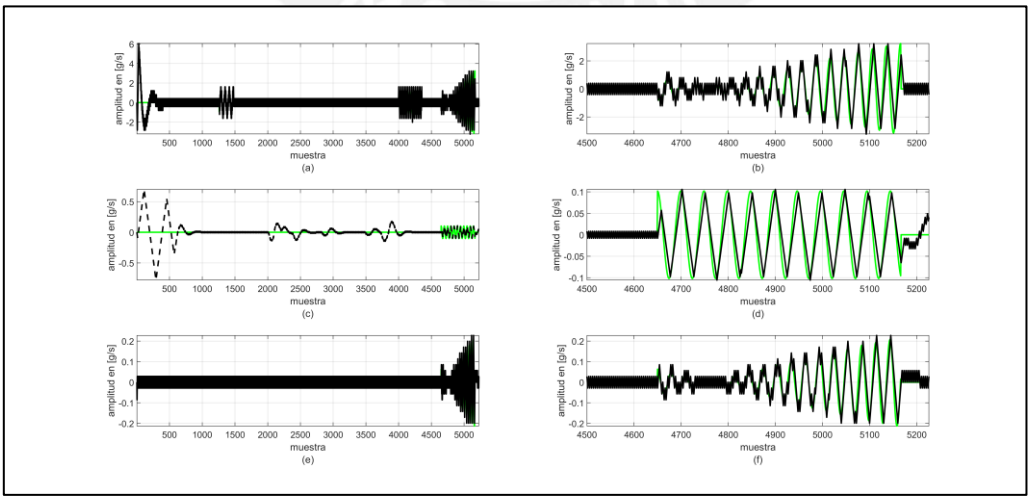

Figura 3.40.: Reconstrucción de perturbaciones (HOSMO). Perturbación real (línea verde). Perturbación estimada (línea negra) (a) Coordenada “x” k = [0 5226] (b) Coordenada “x” k = [4800 5226] (c) Coordenada “y” k = [0 5226] (d) Coordenada “y” k = [4800 5226] (e) Coordenada “z” k = [0 5226] (f) Coordenada “z” k = [4800 5226].
Fuente: Elaboración propia.

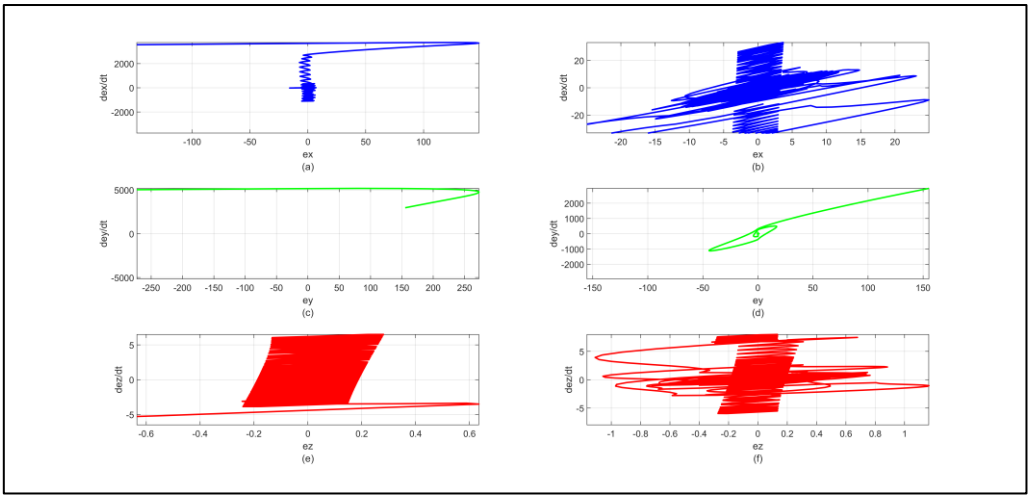

Figura 3.41.: Plano de estados (HOSMO). Trayectoria coordenada "x" (Línea azul). Trayectoria coordenada "y" (Línea verde). Trayectoria coordenada "z" (Línea roja). (a) Coordenada "x" k = [0 5226] (b) Coordenada "x" k = [4800 5226] (c) Coordenada "y" k = [0 5226] (d) Coordenada "y" k = [4800 5226] (e) Coordenada "z" k = [0 5226] (f) Coordenada "z" k = [4800 5226].
Fuente: Elaboración propia.

3.8 Diferenciador Robusto exacto estándar (RED)

Levant y Livne [80] exponen que la estabilización exacta en un tiempo finito de la salida de una planta o proceso denominada $f_0(t)$ puede ser efectuada por un sistema de control de modos deslizantes de orden r o *r-SMC*, siendo un requerimiento imprescindible el conocimiento de las derivadas de la salida $f_0(t)$ hasta el orden $r-1$. Al respecto, se conoce que en la mayoría de aplicaciones reales no se cuenta con ese número de derivadas, siendo esto una limitación que puede ser resuelta implementando observadores conexos al sistema de control que permitan brindar estimaciones de las derivadas no disponibles; esto es factible, pero aumenta la complejidad en el diseño, costo adicional de sensores y posterior implementación.

El diferenciador robusto exacto se presenta como una solución al problema expuesto permitiendo, por medio de la operación de diferenciación, obtener las derivadas no disponibles en una planta o proceso. Por tanto, el problema de diferenciación robusta se resume en evaluar las derivadas $f_0(t), \dot{f}_0(t), \ldots {f_0}^{(n)}(t)$ de la señal $f(t) = f_0(t) + n(t)$ de forma robusta y en tiempo real, bajo la condición de que $\left| {f^{(n+1)}}_0(t) \right| \leq L$, donde $n(t)$ es el ruido de medición, $L > 0$ es la constante de Lipschitz, $n \in N$ es el orden de diferenciación y las derivadas deben ser exactas para $n(t) = 0$ [80]. Cabe resaltar que el término exactitud en tiempo finito, definido por Angulo et al. [81], connota la capacidad del diferenciador de proveer derivadas estimadas con valores exactos a los de las derivadas reales en un tiempo finito, a pesar de la existencia de incertidumbres del modelo.

3.8.1 Fundamentos teóricos

Según Levant [80] el proceso de síntesis de los diferenciadores de orden $r-1$ puede ser simplificado explotando la propiedad de homogeneidad. Esto es, si se cuenta con un diferenciador en función a las variables (t, f, z_i, v_i, w_i) y este cuenta con la propiedad de homogeneidad, entonces el diferenciador es invariante respecto a la transformación $(t, f, z_i, v_i, w_i) \rightarrow (nt, n^{n+1}f, n^{n-i+1}z_i, n^{n-i}v_i, n^{n-i}w_i)$, lo cual permite que las pruebas de convergencia de diferenciadores de orden $r-1$ sean estándar y que se logre la mayor exactitud posible en presencia de ruido de medición. Según lo descrito por Levant [82], la propiedad de homogeneidad permite comprobar que el diferenciador posee convergencia exacta en tiempo finito y de forma asintótica, que estas comprobaciones pueden ser realizadas de forma sencilla y general en la presencia de ruido de medición, mediciones en tiempo discreto y no linealidades tales como retardos de conmutación.

Definición 3.14 (Ponderado homogéneo [80]): *Sean los pesos* $m_1, \ldots, m_{n_x} > 0$ *de las coordenadas* $x_1, \ldots, x_{n_x} \in \mathbb{R}^{n_x}$, *siendo el grado de* x_i *igual al de* m_i *y la dilación* d_k: $(x_1, \ldots, x_{n_x}) \rightarrow$

$(k^{m_1}x_1, \ldots, k^{m_{n_x}}x_{n_x})$, *donde* $k \geq 0$. *Una función* $g: \mathbb{R}^{n_x} \to \mathbb{R}^m$ *tiene grado de homogeneidad* $q \in \mathbb{R}$, *con grado de* g *igual al de* q, *si la identidad* $g(x) = k^{-q}g(d_k x)$ *es válida para cualquier* $x \in \mathbb{R}^{n_x} - \{0\}$ *y* $k > 0$.

Definición 3.16 (Homogeneidad de grado q de una inclusión diferencial [82]): Sea *una inclusión diferencial según la definición (3.4), se dice que esta es de grado de homogeneidad* $q \in \mathbb{R}$, *si la identidad* $F(x) = k^{-q}{d_k}^{-1}F(d_k, x)$ *es válida para cualquier* $x \neq 0$ *y* $k > 0$.

Sea $Lip_n(L)$ el conjunto de funciones $\mathbb{R}_+ \to \mathbb{R}$ que su derivada número n posee la constante de Lipschitz $L > 0$. Por tanto, los diferenciadores deben ser exactos en $Lip_n(L)$ después de un tiempo finito transitorio, teniendo en cuenta que al efectuar la medición de la señal de entrada $f(t)$, para $t \geq 0$, esta posee la forma $f(t) = f_0(t) + n(t)$, donde $f_0(t) \in Lip_n(L)$ es una señal desconocida y $n(t)$ ruido medible de tipo Lebesgue acotado, cumpliéndose $|n| \leq \varepsilon_0$ y $\varepsilon_0 \geq 0$ [80].

Teorema 3.15 (Exactitud de la diferenciación [80]): *Para cualquier* $t_0 > 0$ *existe un* $\varepsilon_* > 0$ *tal que para cualquier* ε_0, $0 < \varepsilon_0 < \varepsilon_k$ *y* $f_0, f_1 \in Lip_n(L)$ *la desigualdad* $sup_{t \geq 0}|f_1(t) - f_0(t)| \leq \varepsilon_0$ *implica la siguiente desigualdad:*

$$sup_{t \geq t_0}\left|{f_1}^i(t) - {f_0}^i(t)\right| \leq K_{i,n}(2L)^{\frac{i}{n-1}}{\varepsilon_0}^{\frac{n+1-i}{n+1}}, i = 0,1,\ldots,n \qquad (3.135)$$

Donde $K_{i,n}$ y $K_{i,n} \in [1, \frac{\pi}{2}]$ son las denominadas constantes de Kolmorog. Para cada par ε_0, i estas desigualdades se convierten en igualdades para las funciones f_0 y f_1.

Definición 3.17 (Exactitud de la diferenciación en el peor de los casos [80]): *Sea* $z_0(t), z_1(t), \ldots, z_n(t)$ *las salidas de un diferenciador que estima* $f_0(t), \dot{f}_0(t), \ldots {f_0}^{(n)}(t)$ *de la señal* $f_0 \in Lip_n(L)$ *y sea* $z_i(t) \equiv {f_0}^{(i)}(t)$ *para* $t \geq t_0$. *Entonces, al definir* $f = f_1 = f_0 + n$, *despejando* $n = f_1 - f_0$, *se puede decir que la mejor exactitud de diferenciación para un instante* $t_k > t_0$ *es:*

$$sup_{f_{o,n}} sup_{t \geq t_k}\left|z_i - {f_0}^{(i)}\right| \geq K_{i,n}(2L)^{\frac{i}{n-1}}{\varepsilon_0}^{\frac{n+1-i}{n+1}} \qquad (3.136)$$

Definición 3.18 (Diferenciador asintóticamente óptimo [80]: Un diferenciador es llamado asintóticamente óptimo si para las constantes $\mu_i > 0$ se rige de acuerdo a lo establecido en el Teorema (3.15) y provee el grado de exactitud $\left|z_i - {f_0}^{(i)}\right| \leq \mu_i(L)^{\frac{i}{n-1}}{\varepsilon_0}^{\frac{n+1-i}{n+1}}, i = 1,2,\ldots n$ para todas las entradas, ruidos y $\varepsilon_0 \geq 0$, donde $\mu_i \geq K_{i,n}2^{\frac{i}{n-1}} \geq 2^{\frac{i}{n-1}}$.

Definición 3.19 (Diferenciador exactamente convergente [83]: *Un sistema exactamente convergente si todas sus trayectorias convergen al origen en tiempo finito en presencia de perturbaciones que no se desvanecen en el origen.*

Definición 3.20 (Diferenciador exacto y uniforme [84]: Aquel diferenciador en el que cualquier trayectoria del sistema converge al origen para cada perturbación acotada $\left|{f_0}^{(r)}(t)\right| \leq L$ y su tiempo de convergencia es acotado superiormente por una constante, independiente de la condición inicial $\sigma(0) = [\sigma_0(0), \sigma_1(0), ..]^T$.

Definición 3.21 (Diferenciador uniformemente convergente en tiempo finito [84]: *Un sistema es uniformemente convergente en tiempo finito si converge al punto de equilibrio en tiempo finito y el tiempo de convergencia de cualquier trayectoria del sistema al punto de equilibrio esta acotado por una constante que es independiente de las condiciones iniciales.*

Las propiedades de exactitud de convergencia son estudiadas por Levant y Livne [86] y la convergencia uniforme por Angulo et al. [81] y Cruz-Zavala et al. [83,84]. Sin embargo, los diferenciadores pueden mostrar prestaciones adicionales de filtrado, como por ejemplo en el diferenciador robusto exacto de filtrado estándar de Levant y Livne [86], y adaptación de constante de Lipschitz cuando el límite superior de la perturbación es desconocido (Reichhartinger y Spurgeon [88]). Asimismo, Mojallizadeh et al. [85] realizan un estudio completo de los tipos de diferenciadores que existen en la literatura, clasificando a los diferenciadores en su forma continua de acuerdo al siguiente detalle:

- Diferenciador de valor establecido de Slotine-Hedrick-Misawa
- Diferenciador Super twisting
- Diferenciador Super twisting de orden arbitrario (Levant)
- Diferenciador Robusto Exacto y Uniforme
- Diferenciador de modos deslizantes cuadrático
- Diferenciador homogéneo
- Diferenciador adaptativo
- Diferenciador ALIEN
- Diferenciador de ganancias altas (High gain differentiator)

Sin lugar a dudas, el diferenciador más conocido es el diferenciador robusto exacto de orden arbitrario de Levant, el cual es presentado en [80], quien propone un algoritmo no recursivo,

$$\begin{aligned}
\dot{z}_0 &= -\tilde{\lambda}_n L^{\frac{1}{n+1}} \lfloor z_0 - f(t) \rceil^{\frac{n}{n+1}} + z_1 \\
\dot{z}_1 &= -\tilde{\lambda}_{n-1} L^{\frac{2}{n+1}} \lfloor z_0 - f(t) \rceil^{\frac{n-1}{n+1}} + z_2 \\
&\cdot \\
&\cdot \\
&\cdot \\
\dot{z}_{n-1} &= -\tilde{\lambda}_1 L^{\frac{n}{n+1}} \lfloor z_0 - f(t) \rceil^{\frac{1}{n+1}} + z_n \\
\dot{z}_n &= -\tilde{\lambda}_0 L sign(z_0 - f(t))
\end{aligned} \tag{3.137}$$

Donde, la función $\lfloor x \rceil^y = |x|^y sign(x)$ para $x, y \in \mathbb{R}$, y los parámetros $\tilde{\lambda}_0, \tilde{\lambda}_1, .. y\ \tilde{\lambda}_n$ pueden ser calculados mediante las siguientes relaciones:

$$\tilde{\lambda}_n = \lambda_n \tag{3.138a}$$

$$\tilde{\lambda}_i = \lambda_i\, \tilde{\lambda}_{i+1}^{\frac{i}{i+1}}, \qquad i = n-1, n-2, \ldots, 0 \tag{3.138b}$$

En [80] el autor presenta la siguiente tabla de parámetros $\tilde{\lambda}_0, \tilde{\lambda}_1, .. y\ \tilde{\lambda}_n$ hasta el orden $n = 7$:

n	$\tilde{\lambda}_0$	$\tilde{\lambda}_1$	$\tilde{\lambda}_2$	$\tilde{\lambda}_3$	$\tilde{\lambda}_4$	$\tilde{\lambda}_5$	$\tilde{\lambda}_6$	$\tilde{\lambda}_7$
0	1.1	-	-	-	-	-	-	-
1	1.1	1.5	-	-	-	-	-	-
2	1.1	2.12	2	-	-	-	-	-
3	1.1	3.06	4.16	3	-	-	-	-
4	1.1	4.57	9.30	10.03	5	-	-	-
5	1.1	6.75	20.26	32.24	23.72	7	-	-
6	1.1	9.91	43.65	101.96	110.08	47.69	10	-
7	1.1	14.13	88.78	295.74	455.40	281.37	84.14	12

Tabla 3.2.: Ganancias homogéneas del diferenciador robusto exacto (RED)
Fuente: Livne y Levant [80].

Y también propone un algoritmo recursivo:

$$\begin{aligned}
\dot{z}_0 &= v_0, v_0 = -\lambda_n L^{\frac{1}{n+1}} \lfloor z_0 - f(t) \rceil^{\frac{n}{n+1}} + z_1 \\
\dot{z}_1 &= v_1, v_1 = -\lambda_{n-1} L^{\frac{1}{n}} \lfloor z_1 - v_0 \rceil^{\frac{n-1}{n}} + z_2 \\
&\cdot \\
&\cdot \\
&\cdot \\
\dot{z}_{k-1} &= v_{k-1}, v_{k-1} = -\lambda_1 L^{\frac{1}{2}} \lfloor z_{k-1} - v_{k-2} \rceil^{\frac{1}{2}} + z_n \\
\dot{z}_n &= -\lambda_0 L sign(z_n - v_{k-1})
\end{aligned} \tag{3.139}$$

Donde los parámetros $\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_n$ pueden ser elegidos mediante la secuencia infinita $\vec{\lambda} = \{1.1, 1.5, 2, 3, 5, 7, 10, 12, 14, 17, 20, 26, 32, \ldots\}$.

A partir de esta introducción a los diferenciadores, se efectuará el diseño del diferenciador robusto exacto estándar de Levant, el cual sirve de base para el diseño de diferenciadores más complejos.

3.8.2 Diseño del diferenciador robusto exacto estándar (RED)

Sea el sistema dinámico lineal incierto en espacio de estados del sistema discreto planteado en (2.23),

$$x_{k+1} = F_k x_k + G_k u_k + D_k \xi_k + w_k$$
$$y_k = H_k{}^{25} x_k + v_k$$

Y $f_x(t)$, $f_y(t)$ y $f_z(t)$ la posición medida del blanco en la coordenada "x", coordenada "y" y coordenada "z", respectivamente, obtenidas del modelo de medición $y_k = H_k x_k + v_k$, el diferenciador robusto exacto estándar de cuarto orden[26] de forma continua en la coordenada "x", siendo los diferenciadores de las otras dos coordenadas idénticos a este, se detalla a continuación:

$$\begin{aligned}
\dot{z}_{x_0} &= -\tilde{\lambda}_4 L^{\frac{1}{5}} \left\lfloor z_{x_0} - f_x(t) \right\rceil^{\frac{4}{5}} + z_{x_1} \\
\dot{z}_{x_1} &= -\tilde{\lambda}_3 L^{\frac{2}{5}} \left\lfloor z_{x_0} - f_x(t) \right\rceil^{\frac{3}{5}} + z_{x_2} \\
\dot{z}_{x_2} &= -\tilde{\lambda}_2 L^{\frac{3}{5}} \left\lfloor z_{x_0} - f_x(t) \right\rceil^{\frac{2}{5}} + z_{x_3} \\
\dot{z}_{x_3} &= -\tilde{\lambda}_1 L^{\frac{4}{5}} \left\lfloor z_{x_0} - f_x(t) \right\rceil^{\frac{1}{5}} + z_{x_4} \\
\dot{z}_{x_4} &= -\tilde{\lambda}_0 L sign(z_{x_0} - f_x(t))
\end{aligned} \tag{3.140}$$

Donde, $f_x(t)$ es la versión continua de posición medida de la coordenada x, z_{x_0} es la estimación de la posición en la coordenada "x", L es la constante de Lipschitz y los parámetros $\tilde{\lambda}_0$, $\tilde{\lambda}_1$, $\tilde{\lambda}_{n-1}$y $\tilde{\lambda}_n$ son las constantes homogéneas del diferenciador, respectivamente. Por otro lado, el diferenciador robusto exacto estándar en su versión continua debe ser discretizado a fin de que sea compatible con el modelo lineal incierto del blanco aéreo de alta maniobrabilidad planteado en el capítulo dos, dado que este modelo es digital. Al respecto, Livne y Levant [86] proponen el modelo discreto de un diferenciador robusto exacto homogéneo por medio de una aproximación de Taylor, dado que según sus estudios el método de discretización de un solo paso de Euler destruye la homogeneidad del diferenciador. Por tanto, la versión discreta del diferenciador robusto exacto estándar queda estructurada de la siguiente manera:

$$\begin{aligned}
z_{x_{0_{k+1}}} &= z_{x_{0_k}} + T\tilde{\lambda}_4 L^{\frac{1}{5}} \left| z_{x_{0_k}} - f_{x_k} \right|^{\frac{4}{5}} + \sum_{j=1}^{4} \frac{T^j}{j!} z_{x_{j_k}} \\
z_{x_{1_{k+1}}} &= z_{x_{1_k}} + T\tilde{\lambda}_3 L^{\frac{2}{5}} \left| z_{x_{0_k}} - f_{x_k} \right|^{\frac{3}{5}} + \sum_{j=1}^{3} \frac{T^j}{j!} z_{x_{j+1_k}} \\
z_{x_{2_{k+1}}} &= z_{x_{2_k}} + T\tilde{\lambda}_2 L^{\frac{3}{5}} \left| z_{x_{0_k}} - f_{x_k} \right|^{\frac{2}{5}} + \sum_{j=1}^{2} \frac{T^j}{j!} z_{x_{j+2_k}} \\
z_{x_{3_{k+1}}} &= z_{x_{3_k}} + T\tilde{\lambda}_1 L^{\frac{4}{5}} \left| z_{x_{0_k}} - f_{x_k} \right|^{\frac{1}{5}} + \sum_{j=1}^{1} \frac{T^j}{j!} z_{x_{j+3_k}} \\
z_{x_{4_{k+1}}} &= z_{x_{4_k}} + T\tilde{\lambda}_0 L sign(z_{x_{4_k}} - f_{x_k})
\end{aligned} \tag{3.141}$$

[25] Al igual que en el observador de modos deslizantes Super-Twisting de orden superior, considerar la matriz de mediciones (2.28) donde solo es posible medir las variables de estado de posición. Para el diseño del presente observador se considerará que la medición de velocidad no se encuentra disponible

[26] Se tuvo a bien llegar hasta la diferenciación de la derivada de la sobre aceleración (snap) con la finalidad de obtener una forma de onda suave para z_{x_3}, dado que si se diseña hasta el tercer orden ($\dot{z}_{x_3} = -\tilde{\lambda}_0 L sign(z_{x_0} - f_x(t))$) la forma de onda de z_{x_3} presenta discontinuidades muy pronunciadas.

donde $\sum_{j=1}^{4}\frac{T^j}{j!}z_{x_{j_k}} = Tz_{x_{1_k}} + \frac{T^2}{2}z_{x_{2_k}} + \frac{T^3}{6}z_{x_{3_k}} + \frac{T^4}{24}z_{x_{4_k}}$, $\sum_{j=1}^{3}\frac{T^j}{j!}z_{x_{j+1_k}} = Tz_{x_{2_k}} + \frac{T^2}{2}z_{x_{3_k}} + \frac{T^3}{6}z_{x_{4_k}}$, $\sum_{j=1}^{2}\frac{T^j}{j!}z_{x_{j+2_k}} = Tz_{x_{3_k}} + \frac{T^2}{2}z_{x_{4_k}}$ y $\sum_{j=1}^{1}\frac{T^j}{j!}z_{x_{j+3_k}} = Tz_{x_{4_k}}$ son las aproximaciones de Taylor del diferenciador y f_{x_k} es la posición medida del blanco discretizada. Posteriormente, se establecieron las ganancias homogéneas del diferenciador y constantes de Lipschitz de acuerdo al siguiente detalle:

Diferenciador RED	L	$\tilde{\lambda}_4$	$\tilde{\lambda}_3$	$\tilde{\lambda}_2$	$\tilde{\lambda}_1$	$\tilde{\lambda}_0$
Coordenada x	455	5.5	10.03	9.30	4.57	1.1
Coordenada y	1162	5.5	10.03	9.30	4.57	1.1
Coordenada z	306	5.5	10.03	9.30	4.57	1.1

Tabla 3.3.: Parámetros sintonizados del diferenciador robusto exacto (RED).
Fuente: Elaboración propia.

Cabe resaltar que la constante de Lipschitz fue elegida en base al límite superior de las perturbaciones en las coordenadas "x", "y" y "z". Por ejemplo, para el caso de la coordenada "x" se conoce que el límite superior de la perturbación (en unidades de sobre aceleración) es de $\left| f^{(4)}{}_0(t) \right| = 10\, m/s4$. Por consiguiente, para ofrecer robustez ante las perturbaciones, $T\tilde{\lambda}_0 L \geq 10\, m/s4$ y si $\tilde{\lambda}_1 = 4.57$ y $T = 0.02s$ entonces la constante de Lipschitz en la coordenada "x" deberá ser igual a $L = 454.5455$.

3.8.3 Algoritmo del diferenciador robusto exacto estándar (RED)

A continuación, se presenta el algoritmo del programa del apéndice 2.9 correspondiente al diferenciador robusto exacto estándar:

Algoritmo 3.7: Diferenciador Robusto Exacto (RED)

Entradas: Vectores $\mathbf{y_k}$ y tiempo de muestreo ***T***
Salidas: VE estimados de posición, velocidad y aceleración $\boldsymbol{xh_k}$, entrada de control estimada del modelo $\boldsymbol{uh_k}$ y su derivada $\boldsymbol{duh_k}$ ***y*** colectores deslizantes $\boldsymbol{sig_k}$
Inicio del Programa: Invocado por el programa sim_RED.m
1: Declarar e inicializa variables de estado estimadas de posición, velocidad, aceleración, sobre aceleración y derivada de la sobre aceleración de la coordenada "x" ($\boldsymbol{z0x, z1x, z2x, z3x}$ ***y*** $\boldsymbol{z4x}$), de la coordenada "y" ($\boldsymbol{z0y, z1y, z2y, z3y}$ ***y*** $\boldsymbol{z4y}$) y coordenada "z" ($\boldsymbol{z0z, z1z, z2z, z3z}$ ***y*** $\boldsymbol{z4z}$)**.**
2: Definir constante de Lipschitz de las coordenadas "x", "y" y "z".
3: Definir el orden ***n=4*** de diferenciación de la coordenadas "x","y" y "z"
4: Definir las ganancias homogéneas del diferenciador de la coordenada "x", "y" y "z"
k0x, k1x, k2x, k3x y **k4y, k0y, k1y, k2y, k3y** y **k4y y k0z, k1z, k2z, k3z** y **k4z.**
5: Establecer series de Taylor para discretización del diferenciador
6: Establecer las variables deslizantes **sigx=z0x-yk(1,1), sigy=z0y-yk(3,1) y sigz=z0z-yk(5,1) y sigdx=z1x-yk(2,1), sigdy=z1y-yk(4,1) y sigdz=z1z-yk(6,1).**
7: Calcular VE estimadas futuras **z0x_1, z1x_1, z2x_1, z3x_1** y **z4x_1, z0y_1, z1y_1, z2y_1, z3y_1** y **z4y_1,** y **z0z_1, z1z_1, z2z_1, z3z_1** y **z4z_1.**
8: Guardar variables para siguiente iteración.
9: Entrega $\boldsymbol{xh_k} = [\boldsymbol{z0x}; \boldsymbol{z1x}; \boldsymbol{z2x}; \boldsymbol{z0y}; \boldsymbol{z1y}; \boldsymbol{z2y}; \boldsymbol{z0z}; \boldsymbol{z1z}; \boldsymbol{z2z}]$
10: Entrega $\boldsymbol{uh_k} = [\boldsymbol{z3x}; \boldsymbol{z3y}; \boldsymbol{z3z}]$
11: Entrega $\boldsymbol{duh_k} = [\boldsymbol{z4x}; \boldsymbol{z4y}; \boldsymbol{z4z}]$
12: Entrega $\boldsymbol{sig_k} = [\mathbf{sigx}; \mathbf{sigy}; \mathbf{sigz}; \mathbf{sigdx}; \mathbf{sigdy}; \mathbf{sigdz}]$
13: Fin de programa

3.8.4 Simulación de la trayectoria del misil

El programa del apéndice 2.10 permite la simulación de la trayectoria del misil, el cual se basa en el algoritmo que se detalla a continuación:

Algoritmo 3.8: Simulación de trayectoria del blanco con diferenciador RED

Entradas: Señal de entrada de control $\boldsymbol{\mu_k}$**,** perturbaciones $\boldsymbol{\xi_k}$ y modelo mixto gaussiano de ruido
Salidas: VE reales del modelo $\boldsymbol{x_k}$, $\boldsymbol{VE}$ medidas del modelo $\boldsymbol{y_k}$ **, VE estimadas** $\boldsymbol{xh_k}$, entrada de control al modelo (sobre aceleración) estimado $\boldsymbol{uh_k}$, derivada de la entrada de control al modelo estimado $\boldsymbol{duh_k}$, variables deslizantes $\boldsymbol{sig_k}$, errores cartesianos $\boldsymbol{error\_c_k}$ errores polares $\boldsymbol{error\_c_k}$, error RMSE y gráficos.
Inicio del programa:
1: Declara tiempo de muestreo $\boldsymbol{T}$, gravedad $\boldsymbol{g}$**,** constante de mach **M**, sobre aceleración máxima ***load_factor*** tiempo inicial de simulación $\boldsymbol{t_i}$**,** tiempo final de simulación t_{ff} y número total de

Algoritmo 3.8: Simulación de trayectoria del blanco con diferenciador RED (continuación)

muestras nt.
2: Declara condiciones iniciales de traqueo al blanco: distancia Ad, acimut verdadero Bdn, elevación Ed, velocidad Vm y rumbo verdadero Rv.
3: Convierte posición de coordenadas polares a cartesianas usando Ad, Bdn y Ed
4: Declara el vector de estados inicial $x_k(0)$
5: Declara matrices del espacio de estados F_k, G_k, D_k y C_k.
6: Obtén u_k (invoca a función ***gen_jerk_sea_skimming3.m*** **(programa del apéndice 2.5))**
7: Obtén ξ_k **(invoca** a función ***perturb_1.m*** **(programa del apéndice 2.6))**
8: Cargar ruido angular ***glint_puntos.mat*** **(datos guardados del programa del apéndice 2.7)**
9: Ingrese "1" para simulación con ruido y "0" para simulación sin ruido.
10: Para tt = ti:T:tff **Hacer**
11: Ejecuta el modelo de transición de estados para obtener x_k futuro.
12: Invoca ***c2p.m*** para convertir x_k a coordenadas polares.
13: Invoca ***yk_noise.m*** para obtener el vector de mediciones y_k
14: Invoca **RED.m (programa del apéndice 2.9)** para obtener xh_k, uh_k, duh_k y sig_k
15: Invoca **c2p.m** para convertir xh_k a coordenadas polares.
16: Extrae errores polares, errores cartesianos y RMSE
17: Almacena vectores para gráficas
18: Fin Para
19: Conversión de unidades del vector de estados a Mn, Mach y g's.
20: Graficar
21: Fin de programa

Al efectuar la simulación sin ruido, se observa en la figura 3.42 que aparentemente la trayectoria del diferenciador es exacta, sin embargo, la figura 3.43 muestra un error de exactitud mayor que el de los otros observadores durante la maniobra terminal. Asimismo, en las figuras 3.44, 3.45 y 3.46 se aprecia que el diferenciador, durante las oscilaciones, no presenta la propiedad de uniformidad y exactitud dado que en algunos casos el valor de la constante de Lipschitz asumido es muy bajo o muy alto respecto a su valor real, dando lugar a subestimaciones o sobreestimaciones de la trayectoria, respectivamente.

De los resultados obtenidos se verifica que el desempeño del diferenciador se ve afectado porque el valor verdadero de la constante de Lipschitz *L* es cambiante durante la trayectoria, y al colocar un valor fijo de esta en el algoritmo se produce una sobre estimación o sub estimación de las variables de estado del sistema, lo que genera una convergencia lenta e inexacta. Cabe resaltar que en Barth et al. [71] se explica que en la mayoría de casos prácticos se desconoce el límite máximo de la perturbación y que sus parámetros de amplitud, frecuencia y fase son variantes en el tiempo, por lo que es usual en los observadores robustos a base de algoritmos *Super twisting* que las ganancias y la constante de Lipschitz sean escogidas en base al peor escenario o *worst-case scenario* que se podría presentar, generando una posible sobre estimación de estos parámetros y por consiguiente un innecesario y desproporcionado esfuerzo de compensación de las incertidumbres estructuradas o perturbaciones. Por ello, en el siguiente apartado se tratará de resolver las limitaciones del diferenciador a fin conseguir derivadas exactas y uniformes en la ausencia de ruido de medición.

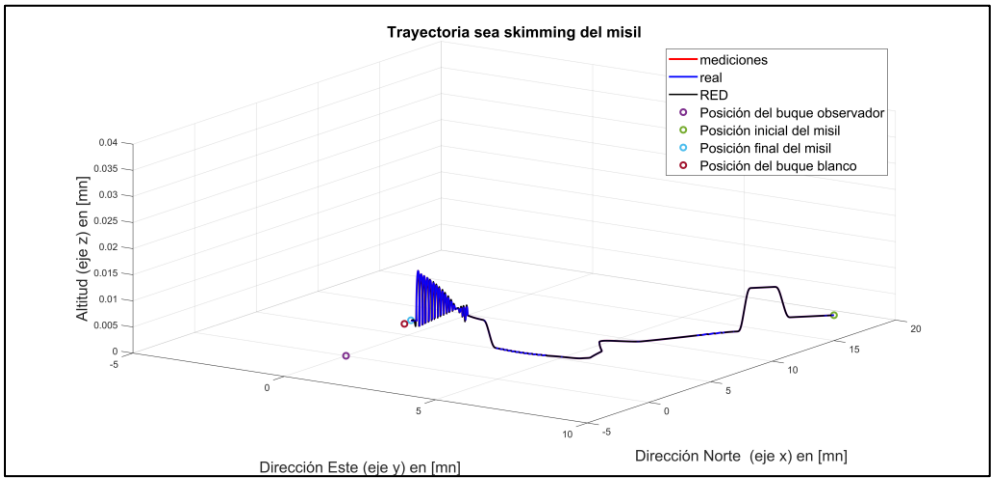

Figura 3.42.: Trayectoria completa del misil sin ruido (RED).
Fuente: Elaboración propia.

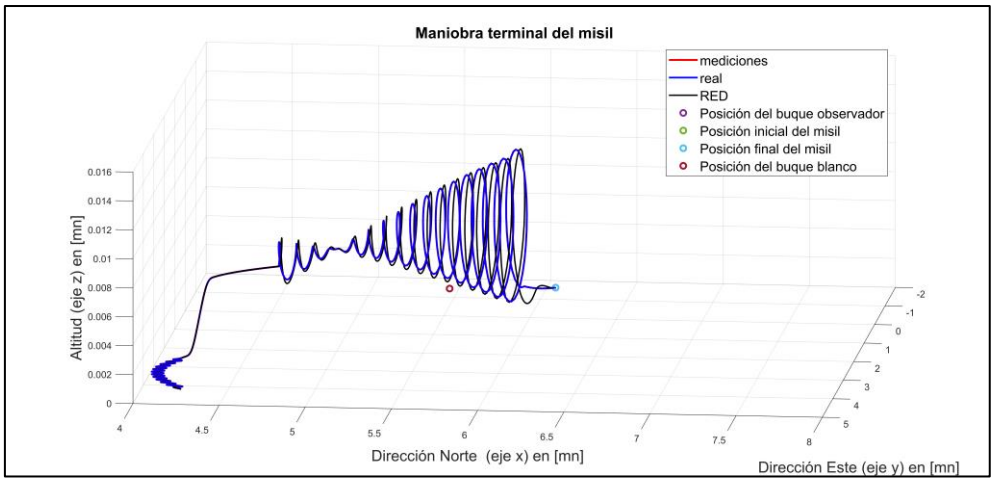

Figura 3.43.: Maniobra terminal del misil sin ruido (RED).
Fuente: Elaboración propia.

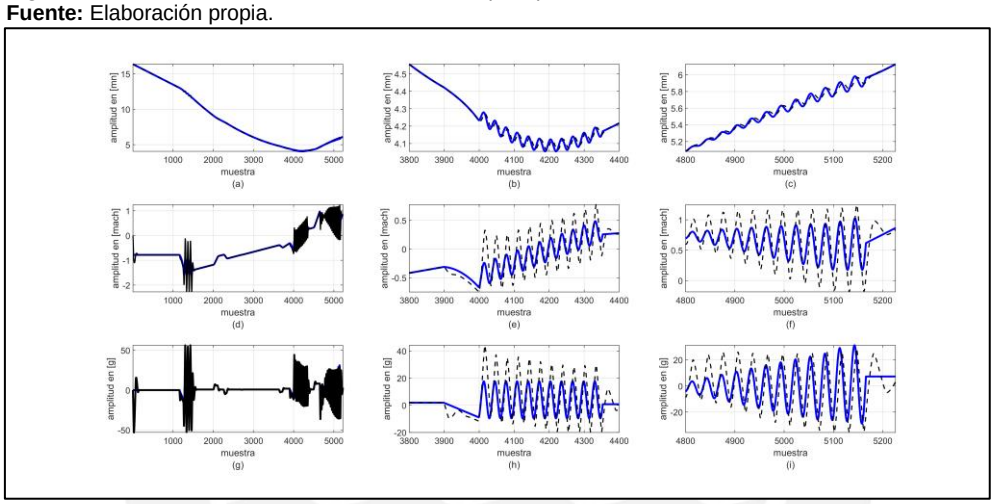

Figura 3.44.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "x" en ausencia de ruido (RED). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [3800 4400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [3800 4400]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

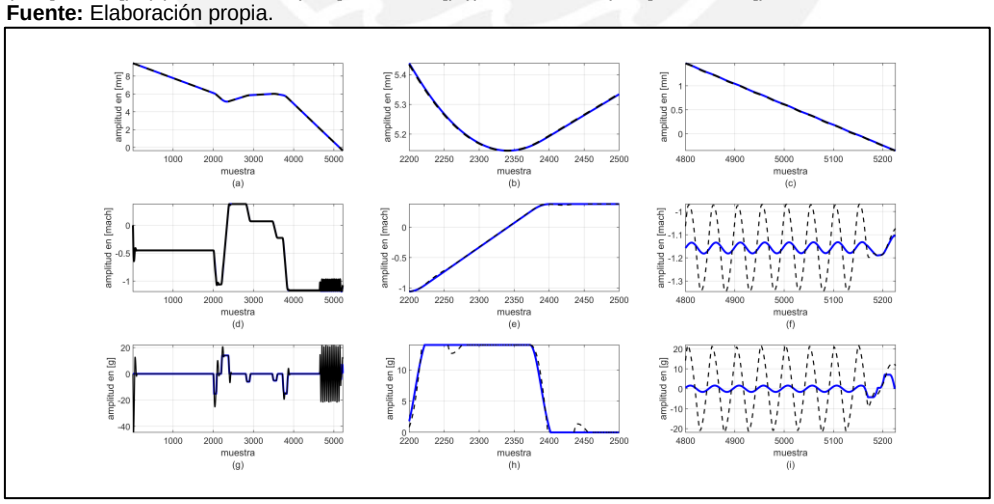

Figura 3.45.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "y" en la ausencia de ruido (RED). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [2200 2500]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [2200 2500]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [2200 2500]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

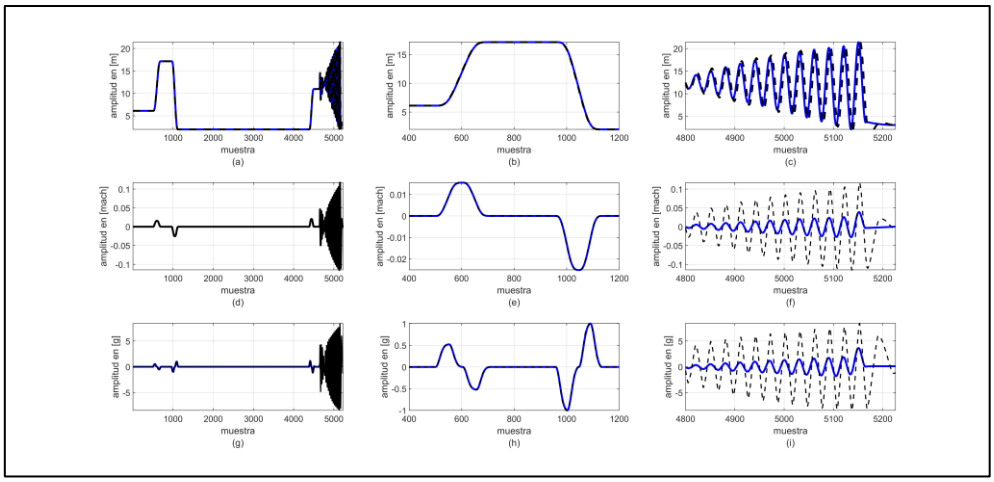

Figura 3.46.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "z" en la ausencia de ruido (RED). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [400 1200]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [400 1200]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

3.9 Diferenciador Robusto exacto adaptativo (ARED)

La variación en el tiempo de los parámetros de frecuencia, amplitud y fase de las variables de estado y perturbaciones del sistema lineal incierto del blanco aéreo de alta maniobrabilidad obliga a introducir la propiedad de adaptación de la constante de Lipschitz en línea. Según lo expuesto anteriormente, los trabajos de Shtessel et al. [70] y Barth et al. [71] son algunos de los más resaltantes en este campo, proponiendo soluciones para la adaptación de parámetros en sistemas de control de modos deslizantes basados en algoritmos super twisting donde el límite superior de las incertidumbres o perturbaciones es desconocido. Asimismo, Plestan et al. [86] propone nuevas metodologías para el diseño de leyes de adaptación a base de funciones de Lyapunov, las cuales cuentan con la posibilidad de brindar estabilidad global. Por otro lado, Reichhartinger y Spurgeon [88] proponen métodos matemáticos de sintonización del diferenciador robusto exacto con ganancias constantes, concluyendo que los métodos de sintonización a base de simulaciones son superiores, dado que la sintonización de ganancias constantes es difícil cuando se requiere efectuar la diferenciación de una señal con incertidumbres y con un rango de operación amplio [88], siendo este el caso en muchas aplicaciones reales. Por ello, el método de adaptación de la constante de Lipschitz desconocida que figura en [87] permite obtener estabilidad local partiendo de una representación pseudo lineal del diferenciador y efectuando un análisis de estabilidad de Lyapunov.

3.9.1 Fundamentos teóricos

Sea el diferenciador robusto exacto planteado en 3.137, y en base a la representación pseudo lineal propuesta por Reichhartinger y Spurgeon en [88], se plantea la siguiente representación pseudo lineal[27] del sistema dinámico del error del diferenciador de cuarto orden, adicionando términos de orden alto para la convergencia uniforme del diferenciador:

$$\dot{e} = \begin{bmatrix} -\theta k_0 \gamma |e_0|^{-\frac{1}{5}} - (1-\theta) k_0 \mu^5 |e_0|^{\frac{a}{5}} & 1 & 0 & 0 & 0 \\ -\theta k_1 \gamma^2 |e_0|^{-\frac{2}{5}} - (1-\theta) k_1 \mu^4 |e_0|^{\frac{2a}{5}} & 0 & 1 & 0 & 0 \\ -\theta k_2 \gamma^3 |e_0|^{-\frac{3}{5}} - (1-\theta) k_2 \mu^3 |e_0|^{\frac{3a}{5}} & 0 & 0 & 1 & 0 \\ -\theta k_3 \gamma^4 |e_0|^{-\frac{4}{5}} - (1-\theta) k_3 \mu^2 |e_0|^{\frac{4a}{5}} & 0 & 0 & 0 & 1 \\ -\theta k_4 \gamma^5 |e_0|^{-1} - (1-\theta) k_4 \mu^1 |e_0|^{a} & 0 & 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \frac{d^4}{dt^4} f(t) \qquad (3.142)$$

[27] Considerar en la representación pseudo lineal que se cumple Considerar que $|e_0|^b sign(e_0) = |e_0|^b \frac{e_0}{|e_0|} = |e_0|^{b-1} e_0$

donde $\mathrm{k}_0, k_1, k_2, k_3$ y k_4 son las ganancias del diferenciador robusto exacto según los valores descritos en la tabla 3.1, $\mathrm{e}^{\mathrm{T}} = [\mathrm{e}_0 \quad \mathrm{e}_1 \quad \mathrm{e}_2 \quad \mathrm{e}_3 \quad \mathrm{e}_4]$ es el vector de errores del diferenciador entre las variables de estado reales del sistema y las variables de estado estimadas, donde las funciones $\mathrm{e}_\mathrm{i} = z_i - f^{(i)}(t)$ para i=0,1,2,3,4 deben ser tales que $|e_i| \leq e_{i_M}$ cumple para un límite superior e_{i_M} existente pero desconocido, γ es la constante desconocida de Lipschitz y μ es la constante óptima de uniformidad de los términos de orden alto, $\frac{d^4}{dt^4}f$ es la cuarta derivada de la función $f(t)$, donde debe cumplir $k_4\gamma^5 \geq sup(\frac{d^4}{dt^4}f)$ o $k_4\mu \geq sup(\frac{d^4}{dt^4}f)$, $y\ \theta$ es la función de conmutación entre las funciones de convergencia exacta y las funciones de convergencia uniforme del diferenciador.

Las funciones de orden alto $|e_0|^{\frac{(5+a)}{5}}sign(e_0)$, $|e_0|^{\frac{(5+2a)}{5}}sign(e_0)$, ..., donde a es un parámetro pequeño a ser elegido por el diseñador brindan al diferenciador la propiedad de uniformidad, permitiendo que el error converja al origen en un tiempo finito independientemente de las condiciones iniciales, en la ausencia de ruido de la señal a diferenciar [82,83]. Cabe resaltar que estos términos permitirán mejorar las estimaciones durante los instantes de tiempo en los cuales el modelo del blanco aéreo se ve afectado por perturbaciones senoidales y cosenoidales o no linealidades.

La derivada de la sobre aceleración o también denominada *snap* no es necesaria de estimar, sin embargo, con la finalidad de obtener una derivada suave de la sobre aceleración se aumenta el orden del diferenciador al cuarto orden dado que de esta forma el término de compensación de super torsión para la estimación de la sobre aceleración no es discontinuo. Por otro lado, el sistema 3.142 se puede expresar de la siguiente forma:

$$\dot{\mathrm{e}} =: \mathbf{M}(\mathrm{e}_0, \gamma)\mathrm{e} + \mathbf{b}\frac{\mathrm{d}^4}{\mathrm{dt}^4}\mathrm{f} \tag{3.143}$$

Donde el polinomio característico de la matriz $\mathbf{M}(\mathrm{e}_0, \gamma)$ del sistema de error 3.143 para $\theta = 1$ es:

$$\Delta(s) = s^5 + k_0\gamma|e_0|^{-\frac{1}{5}}s^4 + k_1\gamma^2|e_0|^{-\frac{2}{5}}s^3 + k_2\gamma^3|e_0|^{-\frac{3}{5}}s^2 + k_3\gamma^4|e_0|^{-\frac{4}{5}}s + k_4\gamma^5|e_0|^{-1}$$

y para $\theta = 0$:

$$\Delta(s) = s^5 + k_0\mu_0|e_0|^{\frac{a}{5}}s^4 + k_1\mu_1|e_0|^{\frac{2a}{5}}s^3 + k_2\mu_2|e_0|^{\frac{3a}{5}}s^2 + k_3\mu_3|e_0|^{\frac{4a}{5}}s + k_4\mu_4|e_0|^{a}$$

La estabilidad del sistema se puede evaluar mediante el criterio de estabilidad de Routh-Hurwitz al evidenciar, mediante un método algebraico, si existen polos del sistema en el plano derecho del lugar de las raíces [89]. Al observar los coeficientes del polinomio característico, la condición necesaria pero no suficiente de estabilidad es que para que no existan raíces con partes positivas todos los coeficientes deberán tener el mismo signo y no se deberá contar con coeficientes ausentes. Para este caso en particular se conoce que $\mathrm{k}_0, \mathrm{k}_1, \mathrm{k}_2, \mathrm{k}_3, \mathrm{k}_4 > 0$ y $\gamma, \mu > 1$, por lo que aparentemente para cualquier valor de los parámetros antes mencionados las raíces se encontrarán en el lado izquierdo del lugar geométrico de las raíces o locus. A continuación, se muestra la tabla de coeficientes confeccionada a fin evaluar la estabilidad para $\theta = 1$:

s^5	1	$\mathrm{k}_1\gamma^2\mathrm{e}_0\|^{-\frac{2}{5}}$	$\mathrm{k}_3\gamma^4\mathrm{e}_0\|^{-\frac{4}{5}}$	0
s^4	$\mathrm{k}_0\gamma\mathrm{e}_0\|^{-\frac{1}{5}}$	$\mathrm{k}_2\gamma^3\mathrm{e}_0\|^{-\frac{3}{5}}$	$\mathrm{k}_4\gamma^5\mathrm{e}_0\|^{-1}$	0
s^3	a_1	a_2	0	0
s^2	a_3	a_4	0	
s^1	a_5	0		
s^0	a_6	0		

Donde los determinantes de la tabla de coeficientes son los siguientes:

$$a_1 = \frac{-\begin{vmatrix} 1 & k_1\gamma^2|e_0|^{-\frac{2}{5}} \\ k_0\gamma|e_0|^{-\frac{1}{5}} & k_2\gamma^3|e_0|^{-\frac{3}{5}} \end{vmatrix}}{k_0\gamma|e_0|^{-\frac{1}{5}}} = \left(-\frac{k_2}{k_0} + k_1\right)\gamma^2|e_0|^{-\frac{2}{5}}$$

$$a_2 = \frac{-\begin{vmatrix} 1 & k_3\gamma^4|e_0|^{-\frac{4}{5}} \\ k_0\gamma|e_0|^{-\frac{1}{5}} & k_4\gamma^5|e_0|^{-1} \end{vmatrix}}{k_0\gamma|e_0|^{-\frac{1}{5}}} = \left(-\frac{k_4}{k_0} + k_3\right)\gamma^4|e_0|^{-\frac{4}{5}}$$

$$a_3 = \frac{-\begin{vmatrix} k_0\gamma|e_0|^{-\frac{1}{5}} & k_2\gamma^3|e_0|^{-\frac{3}{5}} \\ a_1 & a_2 \end{vmatrix}}{a_1} = \left(\frac{k_0k_4 - k_3{k_0}^2}{k_1k_0 - k_2} + k_2\right)\gamma^3|e_0|^{-\frac{3}{5}}$$

$$a_4 = \frac{-\begin{vmatrix} k_0\gamma|e_0|^{-\frac{1}{5}} & k_4\gamma^5|e_0|^{-1} \\ a_1 & 0 \end{vmatrix}}{a_1} = k_4\gamma^5|e_0|^{-1}$$

$$a_5 = \frac{-\begin{vmatrix} a_1 & a_2 \\ a_3 & a_4 \end{vmatrix}}{a_3} = \left(\frac{k_1k_2k_4 - k_0{k_1}^2k_4 - {k_4}^2 + 2k_0k_3k_4 - {k_0}^2{k_3}^2 + k_0k_1k_2k_3 - {k_2}^2k_3}{k_0k_4 - {k_0}^2k_3 + k_0k_1k_2 - {k_2}^2}\right)\gamma^4|e_0|^{-\frac{4}{5}}$$

$$a_6 = \frac{-\begin{vmatrix} a_3 & a_4 \\ a_5 & 0 \end{vmatrix}}{a_5} = k_4\gamma^5|e_0|^{-1}$$

Hallados los coeficientes, la condición suficiente establece que para que el sistema sea estable los coeficientes de la primera columna de la tabla deberán ser mayores a cero. Asimismo, el número de cambios de signo de estos coeficientes representa el número de polos ubicados en el lado derecho del locus. Para que esto cumpla, se deben establecer las siguientes desigualdades:

$$a_1 = \left(-\frac{k_2}{k_0} + k_1\right)\gamma^2|e_0|^{-\frac{2}{5}} > 0 \rightarrow a_1 = \left(-\frac{k_2}{k_0} + k_1\right) > 0$$

$$a_2 = \left(-\frac{k_4}{k_0} + k_3\right)\gamma^4|e_0|^{-\frac{4}{5}} > 0 \rightarrow a_2 = \left(-\frac{k_4}{k_0} + k_3\right) > 0$$

$$a_3 = \left(\frac{k_0k_4 - k_3{k_0}^2}{k_1k_0 - k_2} + k_2\right)\gamma^3|e_0|^{-\frac{3}{5}} \rightarrow a_3 = \left(\frac{k_0k_4 - k_3{k_0}^2}{k_1k_0 - k_2} + k_2\right) > 0$$

$$a_4 = k_4\gamma^5|e_0|^{-1} \rightarrow a_4 = k_4 > 0$$

$$a_5 = \left(\frac{k_1k_2k_4 - k_0{k_1}^2k_4 - {k_4}^2 + 2k_0k_3k_4 - {k_0}^2{k_3}^2 + k_0k_1k_2k_3 - {k_2}^2k_3}{k_0k_4 - {k_0}^2k_3 + k_0k_1k_2 - {k_2}^2}\right) > 0$$

$$a_6 = k_4\gamma^5|e_0|^{-1} > 0$$

Por último, para que el polinomio característico del sistema pseudo lineal planteado del diferenciador robusto exacto, es necesario que se cumplan las siguientes condiciones

1) $\left(-\frac{k_2}{k_0} + k_1\right) > 0$
2) $\left(-\frac{k_4}{k_0} + k_3\right) > 0$
3) $k_0k_4 - k_3{k_0}^2 + k_2k_1k_0 - {k_2}^2 > 0$
4) $k_1k_2k_4 - k_0{k_1}^2k_4 - {k_4}^2 + 2k_0k_3k_4 - {k_0}^2{k_3}^2 + k_0k_1k_2k_3 - {k_2}^2k_3 > 0$
5) $k_4 > 0$

Se verifica que las ganancias $k_0 = 5.5, k_1 = 10.03, k_2 = 9.30$, $k_3 = 4.57$ y $k_4 = 1.1$ correspondientes a la tabla 3.1 cumplen con las desigualdades planteadas anteriormente, y dadas sus características de homogeneidad, serán utilizadas como punto de partida para la implementación del diferenciador. Asimismo, para el caso $\theta = 0$ el polinomio característico es estable si se cumplen las condiciones 1) al 5).

3.9.2 Diseño del diferenciador robusto exacto adaptativo (ARED)

En este apartado se diseña un mecanismo de adaptación para los parámetros γ y μ, tomando como referencia la metodología empleada por Plestan et al. [87], la formulación del diferenciador robusto exacto adaptativo de Reichhartinger y Spurgeon [88] y los diseños de funciones de Lyapunov propuestos en [90,91,92]. Sea la siguiente función de Lyapunov propuesta para el sistema pseudo lineal del error planteado en 3.142 para el caso $\theta = 1$:

$$V(z) = V_0(z) + \frac{1}{2w}\tilde{\gamma}^2 \tag{3.144}$$

$V_0(z) = z^T P z$ es una forma cuadrática, donde $z = \left[\lfloor e_0 \rceil^{\frac{2}{5}} \quad e_1 \quad e_2 \quad e_3 \quad \lfloor e_4 \rceil^2\right]^T$, que no es localmente Lipschitz debido a que es diferenciable casi en cualquier punto a excepción del origen, debido al término $\lfloor e_0 \rceil^{\frac{2}{5}}$. Asimismo, P>0 debe cumplir de la siguiente manera para que la forma cuadrática $V_0(z)$ sea positiva definida:

$$P = \begin{bmatrix} 5p_1 & -\frac{1}{2}p_{12} & 0 & 0 & 0 \\ -\frac{1}{2}p_{12} & \frac{5}{3}p_2 & -\frac{1}{2}p_{23} & 0 & 0 \\ 0 & -\frac{1}{2}p_{23} & \frac{1}{2}p_3 & -\frac{1}{2}p_{34} & 0 \\ 0 & 0 & -\frac{1}{2}p_{34} & \frac{1}{2}p_4 & -\frac{1}{2}p_{45} \\ 0 & 0 & 0 & -\frac{1}{2}p_{45} & \frac{1}{3}p_5 \end{bmatrix} \tag{3.145}$$

donde el valor de los coeficientes $p_1, p_2, p_3, p_4, p_5, p_{12}, p_{23}, p_{34}, p_{45}$ deben ser hallados a fin de que $V(z) > 0$ y $\dot{V}(z) < 0$ para todo $z \in \mathbb{R}^5, z \neq 0$. Por otro lado, $w > 0$ es una constante y $\tilde{\gamma} = \gamma - \hat{\gamma}$ es el error entre el valor verdadero y la estimación de la constante de Lipschitz. Dado que se cumple la condición número uno del teorema 3.8, la función $V(z)$ es una función candidata de Lyapunov, siendo positiva definida pero radialmente desacotada. Asimismo, $\dot{V}(z)$ queda representada de la siguiente manera:

$$\dot{V}(z) = \dot{V}_0(z) - \frac{1}{w}\tilde{\gamma}\dot{\hat{\gamma}}$$

De acuerdo a la estructura del sistema pseudo lineal 3.142, las derivadas del vector $\dot{z}^T$ son las siguientes:

$$\dot{z}^T = \left[\dot{e}_0\left(-\frac{3}{5}|e_0|^{-\frac{13}{5}}{e_0}^2 + |e_0|^{-\frac{3}{5}}\right) \quad \dot{e}_1 \quad \dot{e}_2 \quad \dot{e}_3 \quad \dot{e}_4\left(\frac{{e_4}^2}{|e_4|} + |e_4|\right)\right]$$

donde, $\dot{e_0} = -k_0\gamma|e_0|^{-\frac{1}{5}}e_0 + e_1, \dot{e_1} = -k_1\gamma^2|e_0|^{-\frac{2}{5}}e_0 + e_2, \dot{e_2} = -k_2\gamma^3|e_0|^{-\frac{3}{5}}e_0 + e_3, \dot{e_3} = -k_3\gamma^4|e_0|^{-\frac{4}{5}}e_0 + e_4$ y $\dot{e_4} = -k_4\gamma^5|e_0|^{-1}e_0$. Asimismo, se asume que en tiempo finito $e_{\mathrm{i}} = \dot{e}_{\mathrm{i}} = 0$, por tanto, de las ecuaciones dinámicas del error se obtiene la relación $|e_{\mathrm{i+1}}| = k_{\mathrm{i}}\gamma^{i+1}|e_0|^{\frac{4-\mathrm{i}}{5}}$.

Al expandir $V_0(z) = z^T P z$ y $\dot{V}_0(\mathrm{z}) = 2\dot{z}^T \mathrm{P z}$ se obtiene:

$$V_0(z) = 5p_2{e_1}^2 + \frac{1}{2}p_3{e_2}^2 + \frac{1}{2}p_4{e_3}^2 - p_{23}e_1e_2 - p_{34}e_2e_3 + 5p_1{e_0}^2|e_0|^{-\frac{6}{5}} + \frac{1}{3}p_5{e_4}^2|e_4|^2 - p_{45}e_3e_4|e_4| - p_{12}e_0e_1|e_0|^{-\frac{3}{5}}$$

$$\dot{V}_0(\mathrm{z}) = \left(-p_{12}|e_0|^{-\frac{3}{5}} + \frac{3}{5}p_{12}{e_0}^2|e_0|^{-\frac{13}{5}}\right){e_1}^2 - p_{23}{e_2}^2 - p_{34}{e_3}^2 + \frac{10}{3}p_2e_1e_2 + p_3e_2e_3 + \left(-10\gamma k_0p_1|e_0|^{-\frac{1}{5}} + \gamma^2k_1p_{12}|e_0|^{\frac{1}{5}} + 6\gamma k_0p_1{e_0}^2|e_0|^{-\frac{11}{5}}\right){e_0}^2|e_0|^{-\frac{6}{5}} - p_{45}|e_4|^{-1}{e_4}^2|e_4|^2 + p_4|e_4|^{-1}e_3e_4|e_4| + \left(10p_1|e_0|^{-\frac{3}{5}}\right)e_0e_1|e_0|^{-\frac{3}{5}} - p_{23}e_1e_3 - p_{34}e_2e_4 - 6p_1{e_0}^3e_1|e_0|^{-\frac{16}{5}} -$$

$\frac{3}{5}\gamma k_0 p_{12} {e_0}^3 e_1 |e_0|^{-\frac{14}{5}} - \frac{10}{3}\gamma^2 k_1 p_2 e_0 e_1 |e_0|^{-\frac{2}{5}} + \gamma^3 k_2 p_{23} e_0 e_1 |e_0|^{-\frac{3}{5}} + \gamma k_0 p_{12} e_0 e_1 |e_0|^{-\frac{4}{5}} -$
$\gamma^3 k_2 p_3 e_0 e_2 |e_0|^{-\frac{3}{5}} + \gamma^2 k_1 p_{23} e_0 e_2 |e_0|^{-\frac{2}{5}} + \gamma^4 k_3 p_{34} e_0 e_2 |e_0|^{-\frac{4}{5}} - p_{12} e_0 e_2 |e_0|^{-\frac{3}{5}} -$
$\gamma^4 k_3 p_4 e_0 e_3 |e_0|^{-\frac{4}{5}} + \gamma^3 k_2 p_{34} e_0 e_3 |e_0|^{-\frac{3}{5}} + \gamma^5 k_4 p_{45} e_0 e_3 |e_4||e_0|^{-1} + \gamma^5 k_4 p_{45} e_0 e_3 {e_4}^2 |e_0|^{-1}|e_4|^{-1} -$
$\frac{2}{3}\gamma^5 k_4 p_5 e_0 {e_4}^3 |e_0|^{-1} - \frac{2}{3}\gamma^5 k_4 p_5 e_0 e_4 |e_4|^2 |e_0|^{-1} + \gamma^4 k_3 p_{45} e_0 e_4 |e_4||e_0|^{-\frac{4}{5}}$

Como se puede observar la función $\dot{V}_0(\mathrm{z})$ no es forma cuadrática debido a que posee términos mixtos que impiden expresarla de similar forma que $V_0(z)$. Sin embargo, la citada función puede ser expresada de la siguiente manera:

$$\dot{V}_0(\mathrm{z}) = -z^T \tilde{Q} z - p_{23} e_1 e_3 - p_{34} e_2 e_4 - 6p_1 {e_0}^3 e_1 |e_0|^{-\frac{16}{5}} - \frac{3}{5}\gamma k_0 p_{12} {e_0}^3 e_1 |e_0|^{-\frac{14}{5}}$$
$$-\frac{10}{3}\gamma^2 k_1 p_2 e_0 e_1 |e_0|^{-\frac{2}{5}} + \gamma^3 k_2 p_{23} e_0 e_1 |e_0|^{-\frac{3}{5}} + \gamma k_0 p_{12} e_0 e_1 |e_0|^{-\frac{4}{5}} - \gamma^3 k_2 p_3 e_0 e_2 |e_0|^{-\frac{3}{5}}$$
$$+\gamma^2 k_1 p_{23} e_0 e_2 |e_0|^{-\frac{2}{5}} + \gamma^4 k_3 p_{34} e_0 e_2 |e_0|^{-\frac{4}{5}} - p_{12} e_0 e_2 |e_0|^{-\frac{3}{5}} - \gamma^4 k_3 p_4 e_0 e_3 |e_0|^{-\frac{4}{5}}$$
$$+\gamma^3 k_2 p_{34} e_0 e_3 |e_0|^{-\frac{3}{5}} + \gamma^5 k_4 p_{45} e_0 e_3 |e_4||e_0|^{-1} + \gamma^5 k_4 p_{45} e_0 e_3 {e_4}^2 |e_0|^{-1}|e_4|^{-1}$$
$$-\frac{2}{3}\gamma^5 k_4 p_5 e_0 {e_4}^3 |e_0|^{-1} - \frac{2}{3}\gamma^5 k_4 p_5 e_0 e_4 |e_4|^2 |e_0|^{-1} + \gamma^4 k_3 p_{45} e_0 e_4 |e_4||e_0|^{-\frac{4}{5}}$$

donde, $z^T \tilde{Q} z$ es en efecto una forma cuadrática positiva definida para $\tilde{Q} > 0$. De lo anterior, $\tilde{Q}$ queda definida de la siguiente forma:

$$\tilde{Q} = \begin{bmatrix} q_{11} & -5p_1|e_0|^{-\frac{3}{5}} & 0 & 0 & 0 \\ -5p_1|e_0|^{-\frac{3}{5}} & q_{22} & -\frac{10}{6}p_2 & 0 & 0 \\ 0 & -\frac{10}{6}p_2 & p_{23} & -\frac{1}{2}p_3 & 0 \\ 0 & 0 & -\frac{1}{2}p_3 & p_{34} & -\frac{1}{2}p_4|e_4|^{-1} \\ 0 & 0 & 0 & -\frac{1}{2}p_4|e_4|^{-1} & p_{45}|e_4|^{-1} \end{bmatrix} \quad (3.146)$$

donde,

$$q_{11} = 10\gamma k_0 p_1 |e_0|^{-\frac{1}{5}} - \gamma^2 k_1 p_{12} |e_0|^{\frac{1}{5}} - 6\gamma k_0 p_1 {e_0}^2 |e_0|^{-\frac{11}{5}} > 0$$
$$q_{22} = p_{12}|e_0|^{-\frac{3}{5}} - \frac{3}{5}p_{12}{e_0}^2|e_0|^{-\frac{13}{5}} > 0$$

debe cumplir. Ciertamente, q_{22} es positivo definido para cualquier valor $p_{12} > 0$ y $e_0 \neq 0$. Si ${e_0}^2 \leq |e_0|^2, q_{22}(\gamma, e_0) = \frac{2}{5}p_{12}|e_0|^{-\frac{3}{5}}$. Por otro lado, para $q_{11} > 0$ se debe despejar p_1 de la siguiente manera para que $q_{11} > 0$ cumpla:

$$10\gamma k_0 p_1 |e_0|^{-\frac{1}{5}} - \gamma^2 k_1 p_{12} |e_0|^{\frac{1}{5}} - 6\gamma k_0 p_1 |e_0|^2 |e_0|^{-\frac{11}{5}} = 4\gamma k_0 p_1 |e_0|^{-\frac{1}{5}} - \gamma^2 k_1 p_{12} |e_0|^{\frac{1}{5}} > 0$$

$$p_1 = \frac{\epsilon\gamma k_1 p_{12}}{4k_0} |e_0|^{\frac{2}{5}}$$

donde $\epsilon > 1$ debe cumplir para asegurar que $q_{11} = (\epsilon - 1)\gamma^2 k_1 p_{12} |e_0|^{\frac{1}{5}}$ sea positivo definido. Por otro lado, a fin de eliminar el efecto de los términos cuadráticos mixtos restantes de la función (3.146), es necesario plantear las siguientes igualdades:

$$(-6p_1 {e_0}^2 |e_0|^{-\frac{16}{5}} - \frac{3}{5}\gamma k_0 p_{12} {e_0}^2 |e_0|^{-\frac{14}{5}} - \frac{10}{3}\gamma^2 k_1 p_2 |e_0|^{-\frac{2}{5}} + \gamma^3 k_2 p_{23} |e_0|^{-\frac{3}{5}} + \gamma k_0 p_{12} |e_0|^{-\frac{4}{5}}) e_0 e_1 = 0$$
$$(-\gamma^3 k_2 p_3 |e_0|^{-\frac{3}{5}} + \gamma^2 k_1 p_{23} |e_0|^{-\frac{2}{5}} + \gamma^4 k_3 p_{34} |e_0|^{-\frac{4}{5}} - p_{12} |e_0|^{-\frac{3}{5}}) e_0 e_2 = 0$$

$$(-\gamma^4 k_3 p_4 |e_0|^{-\frac{4}{5}} + \gamma^3 k_2 p_{34} |e_0|^{-\frac{3}{5}} + \gamma^5 k_4 p_{45} e_3 |e_4||e_0|^{-1} + \gamma^5 k_4 p_{45} {e_4}^2 |e_0|^{-1} |e_4|^{-1}) e_0 e_3 = 0$$
$$(-\frac{2}{3}\gamma^5 k_4 p_5 {e_4}^2 |e_0|^{-1} - \frac{2}{3}\gamma^5 k_4 p_5 |e_4|^2 |e_0|^{-1} + \gamma^4 k_3 p_{45} |e_4||e_0|^{-\frac{4}{5}}) e_0 e_4 = 0$$

O también, aplicando la desigualdad $e_i \leq |e_i|$, se obtiene:

$$-6p_1 |e_0|^2 |e_0|^{-\frac{16}{5}} - \frac{3}{5}\gamma k_0 p_{12} |e_0|^2 |e_0|^{-\frac{14}{5}} - \frac{10}{3}\gamma^2 k_1 p_2 |e_0|^{-\frac{2}{5}} + \gamma^3 k_2 p_{23} |e_0|^{-\frac{3}{5}} + \gamma k_0 p_{12} |e_0|^{-\frac{4}{5}} \leq 0$$
$$-\gamma^3 k_2 p_3 |e_0|^{-\frac{3}{5}} + \gamma^2 k_1 p_{23} |e_0|^{-\frac{2}{5}} + \gamma^4 k_3 p_{34} |e_0|^{-\frac{4}{5}} - p_{12} |e_0|^{-\frac{3}{5}} \leq 0$$
$$-\gamma^4 k_3 p_4 |e_0|^{-\frac{4}{5}} + \gamma^3 k_2 p_{34} |e_0|^{-\frac{3}{5}} + \gamma^5 k_4 p_{45} |e_3||e_4||e_0|^{-1} + \gamma^5 k_4 p_{45} |e_4|^2 |e_0|^{-1} |e_4|^{-1} \leq 0$$
$$-\frac{2}{3}\gamma^5 k_4 p_5 |e_4|^2 |e_0|^{-1} - \frac{2}{3}\gamma^5 k_4 p_5 |e_4|^2 |e_0|^{-1} + \gamma^4 k_3 p_{45} |e_4||e_0|^{-\frac{4}{5}} \leq 0$$

Al resolver las desigualdades se obtuvieron las siguientes funciones:

$$p_1 = \frac{\epsilon\gamma k_1 p_{12}}{4k_0} |e_0|^{\frac{2}{5}} = \frac{2\epsilon\gamma^{16} k_1 {k_4}^2 {k_3}^2}{k_0 k_2}$$
$$p_2 = \frac{9k_0 p_{12}}{50\gamma k_1} |e_0|^{-\frac{2}{5}} = \frac{9k_0}{50\gamma k_1} (\frac{8\gamma^{15} {k_4}^2 {k_3}^2}{k_2} |e_0|^{-\frac{2}{5}}) |e_0|^{-\frac{2}{5}} = \frac{36\gamma^{14} k_0 {k_4}^2 {k_3}^2}{25 k_2 k_1} |e_0|^{-\frac{4}{5}}$$
$$p_3 = \frac{k_1 p_{23}}{\gamma k_2} |e_0|^{\frac{1}{5}} = (\frac{12\gamma^{12} {k_1}^2 {k_4}^2 {k_3}^2}{k_0 {k_2}^3} + \frac{48\gamma^{12} k_1 {k_0}^2 {k_4}^2 {k_3}^2}{5k_0 {k_2}^3}) |e_0|^{-\frac{1}{5}}$$
$$p_4 = \frac{3{p_{45}}^2}{p_5} = \frac{3(4\gamma^5 k_4)^2}{3} = 16\gamma^{10} {k_4}^2$$
$$p_5 = 3$$
$$p_{12} = \gamma^4 k_3 p_{34} |e_0|^{-\frac{1}{5}} = \gamma^4 k_3 (\frac{8\gamma^{11} {k_4}^2 k_3}{k_2} |e_0|^{-\frac{1}{5}}) |e_0|^{-\frac{1}{5}} = \frac{8\gamma^{15} {k_4}^2 {k_3}^2}{k_2} |e_0|^{-\frac{2}{5}}$$
$$p_{23} = (\frac{15\gamma k_1 + 12\gamma {k_0}^2}{10\gamma^3 k_0 k_2}) p_{12} = \frac{\gamma^{13} {k_4}^2 {k_3}^2 (60k_1 + 48{k_0}^2)}{5k_0 {k_2}^2} |e_0|^{-\frac{2}{5}}$$
$$p_{34} = \frac{8\gamma^3 {k_4}^2 p_5}{3k_3 k_2} |e_4|^2 |e_0|^{-\frac{3}{5}} = \frac{8\gamma^3 {k_4}^2 3}{3k_3 k_2} (k_3 \gamma^4 |e_0|^{-\frac{4}{5}} |e_0|)^2 |e_0|^{-\frac{3}{5}} = \frac{8\gamma^{11} {k_4}^2 k_3}{k_2} |e_0|^{-\frac{1}{5}}$$
$$p_{45} = \frac{4\gamma k_4 p_5}{3k_3} |e_4||e_0|^{-\frac{1}{5}} = \frac{4\gamma k_4 3}{3k_3} (k_3 \gamma^4 |e_0|^{-\frac{4}{5}} |e_0|) |e_0|^{-\frac{1}{5}} = 4\gamma^5 k_4$$

Al expandir $z^T \tilde{Q} z$ se obtiene,

$$\dot{V}_0(z) = -z^T \tilde{Q} z - p_{23} e_1 e_3 - p_{34} e_2 e_4 \leq -z^T \tilde{Q} z - p_{23} |e_1||e_3| - p_{34} |e_2||e_4| \leq 0$$
$$\dot{V}_0(z) = -z^T \tilde{Q} z - p_{23} \left(k_0 \gamma |e_0|^{\frac{4}{5}}\right) \left(k_2 \gamma^3 |e_0|^{\frac{2}{5}}\right) - p_{34} \left(k_1 \gamma^2 |e_0|^{\frac{3}{5}}\right) \left(k_3 \gamma^4 |e_0|^{\frac{1}{5}}\right) \leq 0$$
$$\dot{V}_0(z) = -z^T \tilde{Q} z - p_{23} k_0 k_2 \gamma^4 |e_0|^{\frac{6}{5}} - p_{34} k_1 k_3 \gamma^6 |e_0|^{\frac{4}{5}} \leq 0$$
$$\dot{V}_0(z) = -\mathrm{q22} {\mathrm{e}_1}^2 - \mathrm{p}_{23} {\mathrm{e}_2}^2 - \mathrm{p}_{34} {\mathrm{e}_3}^2 + \frac{10}{3} \mathrm{p}_2 \mathrm{e}_1 \mathrm{e}_2 + \mathrm{p}_3 \mathrm{e}_2 \mathrm{e}_3 - q11 {\mathrm{e}_0}^2 |\mathrm{e}_0|^{-\frac{6}{5}} - \mathrm{p}_{45} |\mathrm{e}_4|^{-1} {\mathrm{e}_4}^2 |\mathrm{e}_4|^2$$
$$+\mathrm{p}_4 |\mathrm{e}_4|^{-1} \mathrm{e}_3 \mathrm{e}_4 |\mathrm{e}_4| + \left(10 \mathrm{p}_1 |\mathrm{e}_0|^{-\frac{3}{5}}\right) \mathrm{e}_0 \mathrm{e}_1 |\mathrm{e}_0|^{-\frac{3}{5}} - \mathrm{p}_{23} k_0 k_2 \gamma^4 |\mathrm{e}_0|^{\frac{6}{5}} - \mathrm{p}_{34} k_1 k_3 \gamma^6 |\mathrm{e}_0|^{\frac{4}{5}}$$
$$\leq -\left(\frac{2}{5} p_{12} |e_0|^{-\frac{3}{5}}\right) {\mathrm{e}_1}^2 - \left(\frac{\gamma^{13} {k_4}^2 {k_3}^2 (60k_1 + 48{k_0}^2)}{5k_0 {k_2}^2} |e_0|^{-\frac{2}{5}}\right) {\mathrm{e}_2}^2 - \left(\frac{8\gamma^{11} {k_4}^2 k_3}{k_2} |e_0|^{-\frac{1}{5}}\right) {\mathrm{e}_3}^2$$
$$+\frac{10}{3} \left(\frac{36\gamma^{14} k_0 {k_4}^2 {k_3}^2}{25 k_2 k_1} |e_0|^{-\frac{4}{5}}\right) \mathrm{e}_1 \mathrm{e}_2 + \left(\left(\frac{12\gamma^{12} {k_1}^2 {k_4}^2 {k_3}^2}{k_0 {k_2}^3} + \frac{48\gamma^{12} k_1 {k_0}^2 {k_4}^2 {k_3}^2}{5k_0 {k_2}^3}\right) |e_0|^{-\frac{1}{5}}\right) \mathrm{e}_2 \mathrm{e}_3$$
$$-\left((\epsilon - 1)\gamma^2 k_1 p_{12} |e_0|^{\frac{1}{5}}\right) {\mathrm{e}_0}^2 |\mathrm{e}_0|^{-\frac{6}{5}} - (4\gamma^5 k_4) |\mathrm{e}_4|^{-1} {\mathrm{e}_4}^2 |\mathrm{e}_4|^2 + \left(16\gamma^{10} {k_4}^2\right) |\mathrm{e}_4|^{-1} \mathrm{e}_3 \mathrm{e}_4 |\mathrm{e}_4|$$
$$+\left(10 \left(\frac{2\epsilon\gamma^{16} k_1 {k_4}^2 {k_3}^2}{k_0 k_2}\right) |\mathrm{e}_0|^{-\frac{3}{5}}\right) \mathrm{e}_0 \mathrm{e}_1 |\mathrm{e}_0|^{-\frac{3}{5}} - \left(\frac{\gamma^{13} {k_4}^2 {k_3}^2 (60k_1 + 48{k_0}^2)}{5k_0 {k_2}^2} |e_0|^{-\frac{2}{5}}\right) k_0 k_2 \gamma^4 |e_0|^{\frac{6}{5}}$$
$$-\left(\frac{8\gamma^{11} {k_4}^2 k_3}{k_2} |e_0|^{-\frac{1}{5}}\right) k_1 k_3 \gamma^6 |e_0|^{\frac{4}{5}} < 0$$

$$\dot{V}_0(z) = -\left(\frac{2}{5}\left(\frac{8\gamma^{15}{k_4}^2{k_3}^2}{k_2}|e_0|^{-\frac{2}{5}}\right)|e_0|^{-\frac{3}{5}}\right){e_1}^2 - \left(\frac{\gamma^{13}{k_4}^2{k_3}^2(60k_1+48{k_0}^2)}{5k_0{k_2}^2}|e_0|^{-\frac{2}{5}}\right){e_2}^2$$
$$-\left(\frac{8\gamma^{11}{k_4}^2k_3}{k_2}|e_0|^{-\frac{1}{5}}\right){e_3}^2 + \frac{10}{3}\left(\frac{36\gamma^{14}k_0{k_4}^2{k_3}^2}{25k_2k_1}|e_0|^{-\frac{4}{5}}\right)e_1e_2$$
$$+\left(\left(\frac{12\gamma^{12}{k_1}^2{k_4}^2{k_3}^2}{k_0{k_2}^3}+\frac{48\gamma^{12}k_1{k_0}^2{k_4}^2{k_3}^2}{5k_0{k_2}^3}\right)|e_0|^{-\frac{1}{5}}\right)e_2e_3$$
$$-(\epsilon-1)\gamma^2k_1(\frac{8\gamma^{15}{k_4}^2{k_3}^2}{k_2}|e_0|^{-\frac{1}{5}}){e_0}^2|e_0|^{-\frac{6}{5}} - (4\gamma^5k_4)|e_4|^{-1}{e_4}^2|e_4|^2 + (16\gamma^{10}{k_4}^2)|e_4|^{-1}e_3e_4|e_4|$$
$$+10\left(\frac{2\epsilon\gamma^{16}k_1{k_4}^2{k_3}^2}{k_0k_2}\right)|e_0|^{-\frac{3}{5}}e_0e_1|e_0|^{-\frac{3}{5}} - \frac{\gamma^{17}{k_4}^2{k_3}^2(60k_1+48{k_0}^2)}{k_2}|e_0|^{\frac{4}{5}}$$
$$-\frac{8\gamma^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}}$$

$$\dot{V}_0(z) = -\frac{16\gamma^{15}{k_4}^2{k_3}^2}{5k_2}|e_0|^{-1}{e_1}^2 - \frac{\gamma^{13}{k_4}^2{k_3}^2(60k_1+48{k_0}^2)}{5k_0{k_2}^2}|e_0|^{-\frac{2}{5}}{e_2}^2 - \frac{8\gamma^{11}{k_4}^2k_3}{k_2}|e_0|^{-\frac{1}{5}}{e_3}^2$$
$$+\frac{24\gamma^{14}k_0{k_4}^2{k_3}^2}{5k_2k_1}|e_0|^{-\frac{4}{5}}e_1e_2 + \left(\frac{12\gamma^{12}{k_1}^2{k_4}^2{k_3}^2}{k_0{k_2}^3}+\frac{48\gamma^{12}k_1{k_0}^2{k_4}^2{k_3}^2}{5k_0{k_2}^3}\right)|e_0|^{-\frac{1}{5}}e_2e_3$$
$$-(\frac{8(\epsilon-1)\gamma^{17}k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{-\frac{1}{5}}){e_0}^2|e_0|^{-\frac{6}{5}} - (4\gamma^5k_4|e_4|^{-1}){e_4}^2|e_4|^2 + (16\gamma^{10}{k_4}^2|e_4|^{-1})e_3e_4|e_4|$$
$$+(\frac{20\epsilon\gamma^{16}k_1{k_4}^2{k_3}^2}{k_0k_2}|e_0|^{-\frac{3}{5}})e_0e_1|e_0|^{-\frac{3}{5}} - \frac{\gamma^{17}{k_4}^2{k_3}^2(60k_1+48{k_0}^2)}{k_2}|e_0|^{\frac{4}{5}} - \frac{8\gamma^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}}$$

Posteriormente, se agregan los siguientes términos con la finalidad de que la función $\dot{V}_0(z)$ no dependa del parámetro desconocido γ: $\pm\frac{16\hat{\gamma}^{15}{k_4}^2{k_3}^2}{5k_2}|e_0|^{-1}{e_1}^2$, $\pm\frac{\hat{\gamma}^{13}{k_4}^2{k_3}^2(60k_1+48{k_0}^2)}{5k_0{k_2}^2}|e_0|^{-\frac{2}{5}}{e_2}^2$ $\pm\frac{8\hat{\gamma}^{11}{k_4}^2k_3}{k_2}|e_0|^{-\frac{1}{5}}{e_3}^2, \pm\frac{24\hat{\gamma}^{14}k_0{k_4}^2{k_3}^2}{5k_2k_1}|e_0|^{-\frac{4}{5}}e_1e_2, \pm\left(\frac{12\hat{\gamma}^{12}{k_1}^2{k_4}^2{k_3}^2}{k_0{k_2}^3}\right.$ $\left.+\frac{48\hat{\gamma}^{12}k_1{k_0}^2{k_4}^2{k_3}^2}{5k_0{k_2}^3}\right)|e_0|^{-\frac{1}{5}}e_2e_3, \pm(\frac{8(\epsilon-1)\hat{\gamma}^{17}k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{-\frac{1}{5}}){e_0}^2|e_0|^{-\frac{6}{5}}, \pm(4\hat{\gamma}^5k_4|e_4|^{-1}){e_4}^2|e_4|^2,$ $\pm(16\hat{\gamma}^{10}{k_4}^2|e_4|^{-1})e_3e_4|e_4|, \pm(\frac{20\epsilon\hat{\gamma}^{16}k_1{k_4}^2{k_3}^2}{k_0k_2}|e_0|^{-\frac{3}{5}})e_0e_1|e_0|^{-\frac{3}{5}}, \pm\frac{\hat{\gamma}^{17}{k_4}^2{k_3}^2(60k_1+48{k_0}^2)}{k_2}|e_0|^{\frac{4}{5}}$ y $\pm$ $\frac{8\hat{\gamma}^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}}$. Asimismo, los términos dependientes de γ son reemplazados por sus similares dependientes de $\hat{\gamma}$ a fin de formar la función $z^T\tilde{Q}(\hat{\gamma})z$ y el resto de términos son agrupados de la siguiente forma:

$$\dot{V}_0(z) \leq -z^T\tilde{Q}(\hat{\gamma})z - \frac{\hat{\gamma}^{17}{k_4}^2{k_3}^2(60k_1+48{k_0}^2)}{k_2} - \frac{8\hat{\gamma}^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}} - \frac{16\gamma^{15}{k_4}^2{k_3}^2}{5k_2}|e_0|^{-1}|e_1|^2$$
$$+\frac{16\hat{\gamma}^{15}{k_4}^2{k_3}^2}{5k_2}|e_0|^{-1}|e_1|^2 - \frac{\gamma^{13}{k_4}^2{k_3}^2(60k_1+48{k_0}^2)}{5k_0{k_2}^2}|e_0|^{-\frac{2}{5}}|e_2|^2$$
$$+\frac{\hat{\gamma}^{13}{k_4}^2{k_3}^2(60k_1+48{k_0}^2)}{5k_0{k_2}^2}|e_0|^{-\frac{2}{5}}|e_2|^2 - \frac{8\gamma^{11}{k_4}^2k_3}{k_2}|e_0|^{-\frac{1}{5}}|e_3|^2 + \frac{8\hat{\gamma}^{11}{k_4}^2k_3}{k_2}|e_0|^{-\frac{1}{5}}|e_3|^2$$
$$+\frac{24\gamma^{14}k_0{k_4}^2{k_3}^2}{5k_2k_1}|e_0|^{-\frac{4}{5}}|e_1||e_2| - \frac{24\hat{\gamma}^{14}k_0{k_4}^2{k_3}^2}{5k_2k_1}|e_0|^{-\frac{4}{5}}|e_1||e_2|$$
$$+\left(\frac{12\gamma^{12}{k_1}^2{k_4}^2{k_3}^2}{k_0{k_2}^3}+\frac{48\gamma^{12}k_1{k_0}^2{k_4}^2{k_3}^2}{5k_0{k_2}^3}\right)|e_0|^{-\frac{1}{5}}|e_2||e_3|$$
$$-\left(\frac{12\hat{\gamma}^{12}{k_1}^2{k_4}^2{k_3}^2}{k_0{k_2}^3}+\frac{48\hat{\gamma}^{12}k_1{k_0}^2{k_4}^2{k_3}^2}{5k_0{k_2}^3}\right)|e_0|^{-\frac{1}{5}}|e_2||e_3|$$
$$-\left(\frac{8(\epsilon-1)\gamma^{17}k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{-\frac{1}{5}}\right)|e_0|^2|e_0|^{-\frac{6}{5}} + \left(\frac{8(\epsilon-1)\hat{\gamma}^{17}k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{-\frac{1}{5}}\right)|e_0|^2|e_0|^{-\frac{6}{5}}$$
$$-(4\gamma^5k_4|e_4|^{-1})|e_4|^2|e_4|^2 + (4\hat{\gamma}^5k_4|e_4|^{-1})|e_4|^2|e_4|^2 + \left(16\gamma^{10}{k_4}^2|e_4|^{-1}\right)|e_3||e_4||e_4|$$

$$-\left(16\hat{\gamma}^{10}{k_4}^2|e_4|^{-1}\right)|e_3||e_4||e_4| + \left(\frac{20\epsilon\gamma^{16}k_1{k_4}^2{k_3}^2}{k_0k_2}|e_0|^{-\frac{3}{5}}\right)|e_0||e_1||e_0|^{-\frac{3}{5}}$$
$$-\left(\frac{20\epsilon\hat{\gamma}^{16}k_1{k_4}^2{k_3}^2}{k_0k_2}|e_0|^{-\frac{3}{5}}\right)|e_0||e_1||e_0|^{-\frac{3}{5}} - \frac{\gamma^{17}{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{k_2}|e_0|^{\frac{4}{5}}$$
$$+\frac{\hat{\gamma}^{17}{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{k_2}|e_0|^{\frac{4}{5}} - \frac{8\gamma^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}} + \frac{8\hat{\gamma}^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}}$$

$$\dot{V}_0(z) \le -z^T\tilde{Q}(\hat{\gamma})z - \frac{\hat{\gamma}^{17}{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{k_2} - \frac{8\hat{\gamma}^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}} - \frac{16\gamma^{17}{k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{3}{5}}$$
$$+\frac{16\hat{\gamma}^{17}{k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{3}{5}} - \frac{\gamma^{17}{k_1}^2{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{5k_0{k_2}^2}|e_0|^{\frac{4}{5}} + \frac{\hat{\gamma}^{17}{k_1}^2{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{5k_0{k_2}^2}|e_0|^{\frac{4}{5}}$$
$$-\frac{8\gamma^{17}{k_4}^2k_3{k_2}^2}{k_2}|e_0|^{\frac{1}{5}} + \frac{8\hat{\gamma}^{17}{k_4}^2k_3{k_2}^2}{k_2}|e_0|^{\frac{1}{5}} + \frac{24\gamma^{17}{k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{3}{5}}$$
$$-\frac{24\hat{\gamma}^{17}{k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{3}{5}} + \left(\frac{12\gamma^{12}{k_1}^3{k_4}^2{k_3}^2}{k_0{k_2}^2} + \frac{48\gamma^{12}{k_1}^2{k_0}^2{k_4}^2{k_3}^2}{5k_0{k_2}^2}\right)|e_0|^{\frac{4}{5}}$$
$$-\left(\frac{12\hat{\gamma}^{12}{k_1}^3{k_4}^2{k_3}^2}{k_0{k_2}^2} + \frac{48\hat{\gamma}^{12}{k_1}^2{k_0}^2{k_4}^2{k_3}^2}{5k_0{k_2}^2}\right)|e_0|^{\frac{4}{5}} - \frac{8(\epsilon-1)\gamma^{17}k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{3}{5}}$$
$$+\frac{8(\epsilon-1)\hat{\gamma}^{17}k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{3}{5}} - 4\gamma^{17}{k_3}^3k_4|e_0|^{\frac{3}{5}} + 4\hat{\gamma}^{17}{k_3}^3k_4|e_0|^{\frac{3}{5}}$$
$$+16\gamma^{17}k_2k_3{k_4}^2|e_0|^{\frac{3}{5}} - 16\hat{\gamma}^{17}k_2k_3{k_4}^2|e_0|^{\frac{3}{5}} + \frac{20\epsilon\gamma^{17}k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{2}{5}} - \frac{20\epsilon\hat{\gamma}^{17}k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{2}{5}}$$
$$-\frac{\gamma^{17}{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{k_2}|e_0|^{\frac{4}{5}} + \frac{\hat{\gamma}^{17}{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{k_2}|e_0|^{\frac{4}{5}} - \frac{8\gamma^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}}$$
$$+\frac{8\hat{\gamma}^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}}$$

$$\dot{V}_0(z) \le -z^T\tilde{Q}(\hat{\gamma})z - \frac{\hat{\gamma}^{17}{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{k_2} - \frac{8\hat{\gamma}^{17}k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}}$$
$$+(\gamma^{17}-\hat{\gamma}^{17})\left(-\frac{16{k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{3}{5}} - \frac{{k_1}^2{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{5k_0{k_2}^2}|e_0|^{\frac{4}{5}}\right.$$
$$-\frac{8{k_4}^2k_3{k_2}^2}{k_2}|e_0|^{\frac{1}{5}} + \frac{24{k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{3}{5}} + \left(\frac{12{k_1}^3{k_4}^2{k_3}^2}{k_0{k_2}^2} + \frac{48{k_1}^2{k_0}^2{k_4}^2{k_3}^2}{5k_0{k_2}^2}\right)|e_0|^{\frac{4}{5}}$$
$$-\frac{8(\epsilon-1)k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{3}{5}} - 4{k_3}^3k_4|e_0|^{\frac{3}{5}} + 16k_2k_3{k_4}^2|e_0|^{\frac{3}{5}} + \frac{20\epsilon k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{2}{5}}$$
$$\left.-\frac{{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{k_2}|e_0|^{\frac{4}{5}} - \frac{8k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}}\right)$$

$$\dot{V}_0(z) = -z^T\tilde{Q}(\hat{\gamma})z - \hat{\gamma}^{17}\Gamma + \tilde{\gamma}\beta_{17}\varphi$$

donde,

$$\beta_{17} = \gamma^{16} + \gamma^{15}\hat{\gamma} + \gamma^{14}\hat{\gamma}^2 + \gamma^{13}\hat{\gamma}^3 + \gamma^{12}\hat{\gamma}^4 + \gamma^{11}\hat{\gamma}^5 + \gamma^{10}\hat{\gamma}^6 + \gamma^9\hat{\gamma}^7 + \gamma^8\hat{\gamma}^8 + \gamma^7\hat{\gamma}^9 + \gamma^6\hat{\gamma}^{10} + \gamma^5\hat{\gamma}^{11} + \gamma^4\hat{\gamma}^{12}$$
$$+\gamma^3\hat{\gamma}^{13} + \gamma^2\hat{\gamma}^{14} + \gamma\hat{\gamma}^{15} + \hat{\gamma}^{16} > 1$$
$$\varphi = -\frac{16{k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{3}{5}} - \frac{{k_1}^2{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{5k_0{k_2}^2}|e_0|^{\frac{4}{5}} - \frac{8{k_4}^2k_3{k_2}^2}{k_2}|e_0|^{\frac{1}{5}} + \frac{24{k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{3}{5}}$$
$$+\left(\frac{12{k_1}^3{k_4}^2{k_3}^2}{k_0{k_2}^2} + \frac{48{k_1}^2{k_0}^2{k_4}^2{k_3}^2}{5k_0{k_2}^2}\right)|e_0|^{\frac{4}{5}} - \frac{8(\epsilon-1)k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{3}{5}} - 4{k_3}^3k_4|e_0|^{\frac{3}{5}}$$
$$+16k_2k_3{k_4}^2|e_0|^{\frac{3}{5}} + \frac{20\epsilon k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{2}{5}} - \frac{{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{k_2}|e_0|^{\frac{4}{5}} - \frac{8k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}}$$
$$\Gamma = \frac{{k_4}^2{k_3}^2\left(60k_1+48{k_0}^2\right)}{k_2} + \frac{8k_1{k_4}^2{k_3}^3}{k_2}|e_0|^{\frac{3}{5}} > 0$$

Luego, la función $\dot{V}(z)$ queda representada de la siguiente manera:

$$\dot{V}(z) = \dot{V}_0(z) - \frac{1}{w}\tilde{\gamma}\dot{\hat{\gamma}} \; = \; -z^T\tilde{Q}(\hat{\gamma})z - \hat{\gamma}^{17}\Gamma + \tilde{\gamma}(\beta_{17}\varphi \; - \; \frac{1}{w}\dot{\hat{\gamma}}) < 0$$

Luego, la ley de adaptación para el parámetro γ es igual a:

$$\dot{\hat{\gamma}} \; = \; w\beta_{17}\varphi \qquad (3.147)$$

Donde, la constante *w* puede ser escogida $w = \frac{1}{\beta_{17}}$ a fin de eliminar el efecto del parámetro desconocido β_{17} sobre la ley de adaptación $\dot{\hat{\gamma}}$. Por tanto, se comprueba que $V(z) \; > 0$ y $\dot{V}(z) \; = -z^T\tilde{Q}(\hat{\gamma})z \; < 0$ para todo $z \; \in \; \mathbb{R}^5, z \; \neq 0$.

De la misma forma, para el caso $\theta = 0$, se define la siguiente función de Lyapunov:

$$V(z) \; = V_0(z) \; + \frac{1}{2w}\,\tilde{\mu}^2 \qquad (3.148)$$

$V_0(z) \; = z^T P z$ es una forma cuadrática, donde $z = \left[\lfloor e_0 \rceil^{\frac{5+a}{10}} \quad e_1 \quad e_2 \quad e_3 \quad \lfloor e_4 \rceil^2\right]^T$, que no es localmente Lipschitz debido a que es diferenciable casi en cualquier punto a excepción del origen, debido al término $\lfloor e_0 \rceil^{\frac{5+a}{10}}$. Asimismo, se considera la matriz P>0, definida previamente para el caso $\theta = 1$, por lo que al igual que el caso anterior se debe hallar los valores de los coeficientes $p_1, p_2, p_3, p_4, p_5, p_{12}, p_{23}, p_{34}, p_{45}$ a fin de que $V(z) \; > 0$ y $\dot{V}(z) \; < 0$ para todo $z \; \in \; \mathbb{R}^5, z \; \neq 0$. Dado que se cumple la condición número uno del teorema 3.8, la función $V(z)$ es una función candidata de Lyapunov, siendo positiva definida pero radialmente desacotada. Asimismo, $\dot{V}(z)$ queda representada de la siguiente manera:

$$\dot{V}(z) = \dot{V}_0(z) - \frac{1}{w}\,\tilde{\mu}\dot{\hat{\mu}}$$

De acuerdo a la estructura del sistema pseudo lineal 3.142, las derivadas del vector $\dot{z}^T$ son las siguientes:

$$\dot{z}^T \; = \left[\dot{e}_0\left(\frac{-5+a}{10}|e_0|^{\frac{-25+a}{10}}{e_0}^2 + |e_0|^{\frac{-5+a}{10}}\right) \quad \dot{e}_1 \quad \dot{e}_2 \quad \dot{e}_3 \quad \dot{e}_4\left(\frac{{e_4}^2}{|e_4|} + |e_4|\right)\right]$$

donde, $\dot{e_0} = -k_0\mu^5|e_0|^{\frac{a}{5}}e_0 + e_1, \dot{e_1} = -k_1\mu^4|e_0|^{\frac{2a}{5}}e_0 + e_2, \dot{e_2} = -k_2\mu^3|e_0|^{\frac{3a}{5}}e_0 + e_3, \dot{e_3} = -k_3\mu^2|e_0|^{\frac{4a}{5}}e_0 + e_4$ y $\dot{e_4} = -k_4\mu|e_0|^a e_0$. Asimismo, se asume que en tiempo finito $e_{\mathrm{i}} \; = \; \dot{e}_{\mathrm{i}} \; = \; 0$, por tanto, de las ecuaciones dinámicas del error se obtiene la relación $|e_{\mathrm{i+1}}| \; = \; k_{\mathrm{i}}\mu^{5-i}|e_0|^{\frac{(\mathrm{i+1})\mathrm{a}}{5}}$.

Al expandir $V_0(z) = z^T P z$ y $\dot{V}_0(\mathrm{z}) = 2\dot{z}^T \mathrm{Pz}$ se obtiene:

$$V_0(z) \; = \; \frac{5p_2}{3}{e_1}^2 + \frac{1}{2}p_3{e_2}^2 + \frac{1}{2}p_4{e_3}^2 - p_{23}e_1e_2 - p_{34}e_2e_3 + 5p_1{e_0}^2|e_0|^{\frac{-5+a}{5}} \; + \frac{1}{3}p_5{e_4}^2|e_4|^2 - p_{45}e_3e_4|e_4| - p_{12}e_0e_1|e_0|^{\frac{-5+a}{10}}$$

$$\dot{V}_0(z) = \left(-p_{12}|e_0|^{\frac{-5+a}{10}} + \frac{5-a}{10}p_{12}{e_0}^2|e_0|^{\frac{-25+a}{10}}\right){e_1}^2 - p_{23}{e_2}^2 - p_{34}{e_3}^2 + \frac{10}{3}p_2e_1e_2 + p_3e_2e_3 + \left(-10\mu^5k_0p_1|e_0|^{\frac{a}{5}} + \mu^4k_1p_{12}|e_0|^{\frac{3a+5}{10}} + (5-a)\mu^5k_0p_1{e_0}^2|e_0|^{\frac{-10+a}{5}}\right){e_0}^2|e_0|^{\frac{-5+a}{5}} - p_{45}|e_4|^{-1}{e_4}^2|e_4|^2 + p_4|e_4|^{-1}e_3e_4|e_4| + (10p_1|e_0|^{\frac{-5+a}{10}})e_0e_1|e_0|^{\frac{-5+a}{10}} - p_{23}e_1e_3 - p_{34}e_2e_4 - (5-a)p_1{e_0}^3e_1|e_0|^{\frac{-15+a}{5}} - (\frac{5-a}{10})\mu^5k_0p_{12}{e_0}^3e_1|e_0|^{\frac{3a-25}{10}} - \frac{10}{3}\mu^4k_1p_2e_0e_1|e_0|^{\frac{2a}{5}} + \mu^3k_2p_{23}e_0e_1|e_0|^{\frac{3a}{5}} + \mu^5k_0p_{12}e_0e_1|e_0|^{\frac{-5+3a}{5}} - \mu^3k_2p_3e_0e_2|e_0|^{\frac{3a}{5}} + \mu^4k_1p_{23}e_0e_2|e_0|^{\frac{2a}{5}} + \mu^2k_3p_{34}e_0e_2|e_0|^{\frac{4a}{5}} - p_{12}e_0e_2|e_0|^{\frac{-5+a}{10}} - \mu^2k_3p_4e_0e_3|e_0|^{\frac{4a}{5}} + \mu^3k_2p_{34}e_0e_3|e_0|^{\frac{3a}{5}} + \mu k_4p_{45}e_0e_3|e_4||e_0|^a + \mu k_4p_{45}e_0e_3{e_4}^2|e_0|^a|e_4|^{-1} - \frac{2}{3}\mu k_4p_5e_0{e_4}^3|e_0|^a - \frac{2}{3}\mu k_4p_5e_0e_4|e_4|^2|e_0|^a + \mu^2k_3p_{45}e_0e_4|e_4||e_0|^{\frac{4a}{5}}$$

Como se puede observar, al igual que el caso $\theta = 1$, la función $\dot{V}_0(\mathrm{z})$ no es una forma cuadrática debido a que posee términos mixtos que impiden expresarla de similar forma que $V_0(z)$. Sin embargo, la citada función puede ser expresada de la siguiente manera:

$$\dot{V}_0(\mathrm{z}) = -z^T\tilde{Q}z - p_{23}e_1e_3 - p_{34}e_2e_4 - (5-a)p_1{e_0}^3e_1|e_0|^{\frac{-15+a}{5}} - \left(\frac{5-a}{10}\right)\mu^5k_0p_{12}{e_0}^3e_1|e_0|^{\frac{3a-25}{10}}$$
$$-\frac{10}{3}\mu^4k_1p_2e_0e_1|e_0|^{\frac{2a}{5}} + \mu^3k_2p_{23}e_0e_1|e_0|^{\frac{3a}{5}} + \mu^2k_0p_{12}e_0e_1|e_0|^{\frac{-5+3a}{10}} - \mu^3k_2p_3e_0e_2|e_0|^{\frac{3a}{5}}$$
$$+\mu^4k_1p_{23}e_0e_2|e_0|^{\frac{2a}{5}} + \mu^2k_3p_{34}e_0e_2|e_0|^{\frac{4a}{5}} - p_{12}e_0e_2|e_0|^{\frac{-5+a}{10}} - \mu^2k_3p_4e_0e_3|e_0|^{\frac{4a}{5}}$$
$$+\mu^3k_2p_{34}e_0e_3|e_0|^{\frac{3a}{5}} + \mu k_4p_{45}e_0e_3|e_4||e_0|^a + \mu k_4p_{45}e_0e_3{e_4}^2|e_0|^a|e_4|^{-1} - \frac{2}{3}\mu k_4p_5e_0{e_4}^3|e_0|^a$$
$$-\frac{2}{3}\mu k_4p_5e_0e_4|e_4|^2|e_0|^a + \mu^2k_3p_{45}e_0e_4|e_4||e_0|^{\frac{4a}{5}}$$

donde, $z^T\tilde{Q}z$ es en efecto una forma cuadrática positiva definida para $\tilde{Q} > 0$. De lo anterior, $\tilde{Q}$, de estructura similar a la matriz que figura en (3.146) queda representada de la siguiente forma:

$$\tilde{Q} = \begin{bmatrix} q_{11} & -5p_1|e_0|^{\frac{-5+a}{10}} & 0 & 0 & 0 \\ -5p_1|e_0|^{\frac{-5+a}{10}} & q_{22} & -\frac{10}{6}p_2 & 0 & 0 \\ 0 & -\frac{10}{6}p_2 & p_{23} & -\frac{1}{2}p_3 & 0 \\ 0 & 0 & -\frac{1}{2}p_3 & p_{34} & -\frac{1}{2}p_4|e_4|^{-1} \\ 0 & 0 & 0 & -\frac{1}{2}p_4|e_4|^{-1} & p_{45}|e_4|^{-1} \end{bmatrix}$$

donde,

$$q_{11} = 10\mu^5k_0p_1|e_0|^{\frac{a}{5}} - \mu^4k_1p_{12}|e_0|^{\frac{3a+5}{10}} - (5-a)\mu^5k_0p_1{e_0}^2|e_0|^{\frac{-10+a}{5}} > 0$$
$$q_{22} = p_{12}|e_0|^{\frac{-5+a}{10}} - \frac{5-a}{10}p_{12}{e_0}^2|e_0|^{\frac{-25+a}{10}} > 0$$

debe cumplir. Ciertamente, q_{22} es positivo definido para cualquier valor $p_{12} > 0$ y $e_0 \neq 0$. Si ${e_0}^2 \leq |e_0|^2, q_{22}(\gamma, e_0) = \frac{5+a}{10}p_{12}|e_0|^{\frac{-5+a}{10}}$. Por otro lado, para $q_{11} > 0$ se debe despejar p_1 de la siguiente manera para que $q_{11} > 0$ cumpla:

$$10\mu^5k_0p_1|e_0|^{\frac{a}{5}} - \mu^4k_1p_{12}|e_0|^{\frac{3a+5}{10}} - (5-a)\mu^5k_0p_1|e_0|^2|e_0|^{\frac{-10+a}{5}} = (5+a)\mu^5k_0p_1|e_0|^{\frac{a}{5}} - \mu^4k_1p_{12}|e_0|^{\frac{3a+5}{10}} > 0$$

$$p_1 = \frac{\epsilon k_1p_{12}}{(5+a)\mu k_0}|e_0|^{\frac{a+5}{10}}$$

Donde $\epsilon > 1$ debe cumplir para asegurar que $q_{11} = (\epsilon - 1)\mu^4k_1p_{12}|e_0|^{\frac{3a+5}{10}}$ sea positiva definida. Por otro lado, a fin de eliminar el efecto de los términos cuadráticos mixtos restantes de la función (3.146), es necesario plantear las siguientes igualdades:

$$\left(-(5-a)p_1{e_0}^2|e_0|^{\frac{-15+a}{5}} - \left(\frac{5-a}{10}\right)\mu^5k_0p_{12}{e_0}^2|e_0|^{\frac{3a-25}{10}} - \frac{10}{3}\mu^4k_1p_2|e_0|^{\frac{2a}{5}} + \mu^3k_2p_{23}|e_0|^{\frac{3a}{5}} + \mu^5k_0p_{12}|e_0|^{\frac{-5+3a}{10}}\right)e_0e_1 = 0$$
$$\left(-\mu^3k_2p_3|e_0|^{\frac{3a}{5}} + \mu^4k_1p_{23}|e_0|^{\frac{2a}{5}} + \mu^2k_3p_{34}|e_0|^{\frac{4a}{5}} - p_{12}|e_0|^{\frac{-5+a}{10}}\right)e_0e_2 = 0$$
$$(-\mu^2k_3p_4|e_0|^{\frac{4a}{5}} + \mu^3k_2p_{34}|e_0|^{\frac{3a}{5}} + \mu k_4p_{45}|e_4||e_0|^a + \mu k_4p_{45}{e_4}^2|e_0|^a|e_4|^{-1})e_0e_3 = 0$$
$$\left(-\frac{2}{3}\mu k_4p_5{e_4}^2|e_0|^a - \frac{2}{3}\mu k_4p_5|e_4|^2|e_0|^a + \mu^2k_3p_{45}|e_4||e_0|^{\frac{4a}{5}}\right)e_0e_4 = 0$$

Aplicando $e_i \leq |e_i|$, se resuelve el sistema de desigualdades, obteniendo las siguientes funciones:

$$p_1 = \frac{\epsilon k_1 p_{12}}{(5+a)\mu k_0}|e_0|^{\frac{a+5}{10}} = \frac{\epsilon k_1}{(5+a)\mu k_0}\left(\frac{8{k_4}^2{k_3}^2\mu^3}{k_2}|e_0|^{\frac{29a+5}{10}}\right)|e_0|^{\frac{a+5}{10}} = \frac{8\epsilon k_1{k_4}^2{k_3}^2\mu^2}{(5+a)k_0k_2}|e_0|^{\frac{30a+5}{10}}$$

$$p_2 = \frac{(15+3a)\mu k_0 p_{12}}{100k_1}|e_0|^{\frac{-a-5}{10}} = \frac{(30+6a)\mu k_0{k_4}^2{k_3}^2\mu^3}{25k_1k_2}|e_0|^{\frac{28a}{10}}$$

$$p_3 = \frac{\mu k_1 p_{23}}{k_2}|e_0|^{\frac{-a}{5}} = \frac{\mu k_1}{k_2}\left(\frac{8(5-a)\epsilon k_1{k_4}^2{k_3}^2}{(5+a)\mu k_0{k_2}^2}|e_0|^{\frac{28a}{10}}\right)|e_0|^{\frac{-a}{5}} = \frac{8(5-a)\epsilon{k_1}^2{k_4}^2{k_3}^2}{(5+a)k_0{k_2}^3}|e_0|^{\frac{27a}{10}}$$

$$p_4 = \frac{4k_4 p_{45}}{\mu k_3}|e_4||e_0|^{\frac{a}{5}}\ \frac{4k_4}{\mu k_3}(4k_4\mu|e_0|^a)(k_3\mu^2|e_0|^{\frac{4a}{5}})|e_0|^{\frac{a}{5}} = 16{k_4}^2\mu^2|e_0|^{2a}$$

$$p_5 = 3$$

$$p_{12} = \mu^2 k_3 p_{34}|e_0|^{\frac{7a+5}{10}} = \mu^2 k_3\left(\frac{8{k_4}^2k_3\mu}{k_2}|e_0|^{\frac{11a}{5}}\right)|e_0|^{\frac{7a+5}{10}} = \frac{8{k_4}^2{k_3}^2\mu^3}{k_2}|e_0|^{\frac{29a+5}{10}}$$

$$p_{23} = \frac{(5-a)p_1}{\mu^3 k_2}|e_0|^{\frac{-5-2a}{5}} = \frac{(5-a)}{\mu^3 k_2}\left(\frac{8\epsilon k_1{k_4}^2{k_3}^2\mu^2}{(5+a)k_0k_2}|e_0|^{\frac{30a+5}{10}}\right)|e_0|^{\frac{-5-2a}{5}}$$

$$= \frac{8(5-a)\epsilon k_1{k_4}^2{k_3}^2}{(5+a)\mu k_0{k_2}^2}|e_0|^{\frac{28a}{10}}$$

$$p_{34} = \frac{2k_4 p_{45}}{\mu^2 k_2}|e_4||e_0|^{\frac{2a}{5}} = \frac{2k_4}{\mu^2 k_2}(4k_4\mu|e_0|^a)(k_3\mu^2|e_0|^{\frac{4a}{5}})|e_0|^{\frac{2a}{5}} = \frac{8{k_4}^2k_3\mu}{k_2}|e_0|^{\frac{11a}{5}}$$

$$p_{45} = \frac{4k_4 p_5}{3\mu k_3}|e_4||e_0|^{\frac{a}{5}} = \frac{4k_4(3)}{3\mu k_3}(k_3\mu^2|e_0|^{\frac{4a}{5}})|e_0|^{\frac{a}{5}} = 4k_4\mu|e_0|^a$$

Al expandir $z^T\tilde{Q}z$ y reemplazar las funciones halladas se obtiene:

$$\dot{V}_0(z) = -z^T\tilde{Q}z - p_{23}e_1e_3 - p_{34}e_2e_4 \le -z^T\tilde{Q}z - p_{23}|e_1||e_3| - p_{34}|e_2||e_4| < 0$$

$$\dot{V}_0(z) = -z^T\tilde{Q}z - p_{23}\left(k_0\mu^5|e_0|^{\frac{a}{5}}\right)\left(k_2\mu^3|e_0|^{\frac{3a}{5}}\right) - p_{34}\left(k_1\mu^4|e_0|^{\frac{2a}{5}}\right)\left(k_3\mu^2|e_0|^{\frac{4a}{5}}\right) < 0$$

$$\dot{V}_0(z) = -z^T\tilde{Q}z - p_{23}k_0k_2\mu^8|e_0|^{\frac{4a}{5}} - p_{34}k_1k_3\mu^6|e_0|^{\frac{6a}{5}} < 0$$

$$\dot{V}_0(z) = -q22\mathrm{e_1}^2 - \mathrm{p_{23}e_2}^2 - \mathrm{p_{34}e_3}^2 + \frac{10}{3}\mathrm{p_2e_1e_2} + \mathrm{p_3e_2e_3} - q11\mathrm{e_0}^2|e_0|^{\frac{-5+a}{5}} - \mathrm{p_{45}}|\mathrm{e_4}|^{-1}\mathrm{e_4}^2|\mathrm{e_4}|^2$$

$$+\mathrm{p_4}|\mathrm{e_4}|^{-1}\mathrm{e_3e_4}|\mathrm{e_4}| + \left(10\mathrm{p_1}|e_0|^{\frac{-5+a}{10}}\right)\mathrm{e_0e_1}|e_0|^{\frac{-5+a}{10}} - p_{23}k_0k_2\mu^8|e_0|^{\frac{4a}{5}} - p_{34}k_1k_3\mu^6|e_0|^{\frac{6a}{5}} < 0$$

$$\dot{V}_0(z) = -\frac{4(5+a){k_4}^2{k_3}^2\mu^3}{5k_2}|e_0|^{3a}\mathrm{e_1}^2 - \frac{8(5-a)\epsilon k_1{k_4}^2{k_3}^2}{(5+a)\mu k_0{k_2}^2}|e_0|^{\frac{28a}{10}}\mathrm{e_2}^2 - \frac{8{k_4}^2k_3\mu}{k_2}|e_0|^{\frac{11a}{5}}\mathrm{e_3}^2$$

$$+\frac{(20+4a)k_0{k_4}^2{k_3}^2\mu^4}{5k_1k_2}|e_0|^{\frac{28a}{10}}\mathrm{e_1e_2} + \frac{8(5-a)\epsilon{k_1}^2{k_4}^2{k_3}^2}{(5+a)k_0{k_2}^3}|e_0|^{\frac{27a}{10}}\mathrm{e_2e_3}$$

$$-\frac{8(\epsilon-1)\mu^7k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{32a+10}{10}}{e_0}^2|e_0|^{\frac{-5+a}{5}} - 4k_4\mu|e_0|^a|\mathrm{e_4}|^{-1}\mathrm{e_4}^2|\mathrm{e_4}|^2$$

$$+16{k_4}^2\mu^2|e_0|^{2a}\,|\mathrm{e_4}|^{-1}\mathrm{e_3e_4}|\mathrm{e_4}| + \frac{80\epsilon k_1{k_4}^2{k_3}^2\mu^2}{(5+a)k_0k_2}|e_0|^{\frac{31a}{10}}\mathrm{e_0e_1}|e_0|^{\frac{-5+a}{10}} - \frac{8(5-a)\epsilon\mu^7k_1{k_4}^2{k_3}^2}{(5+a)k_2}|e_0|^{\frac{32a}{10}}$$

$$-\frac{8k_1{k_4}^2{k_3}^2\mu^7}{k_2}|e_0|^{\frac{17a}{5}} < 0$$

Posteriormente, se agregan los siguientes términos con la finalidad de que la función $\dot{V}_0(z)$ no dependa del parámetro desconocido γ: $\pm\frac{4(5+a){k_4}^2{k_3}^2\hat{\mu}^3}{5k_2}|e_0|^{3a}\mathrm{e_1}^2$, $\pm\frac{8(5-a)\epsilon k_1{k_4}^2{k_3}^2}{(5+a)\hat{\mu}k_0{k_2}^2}|e_0|^{\frac{28a}{10}}\mathrm{e_2}^2$ $\pm\frac{8{k_4}^2k_3\hat{\mu}}{k_2}|e_0|^{\frac{11a}{5}}\mathrm{e_3}^2, \pm\frac{(20+4a)k_0{k_4}^2{k_3}^2\hat{\mu}^4}{5k_1k_2}|e_0|^{\frac{28a}{10}}\mathrm{e_1e_2}, \pm\frac{8(5-a)\epsilon{k_1}^2{k_4}^2{k_3}^2}{(5+a)k_0{k_2}^3}|e_0|^{\frac{27a}{10}}\mathrm{e_2e_3}$ $, \pm\frac{8(\epsilon-1)\hat{\mu}^7k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{32a+10}{10}}{e_0}^2|e_0|^{\frac{-5+a}{5}}, \pm 4k_4\hat{\mu}|e_0|^a|\mathrm{e_4}|^{-1}\mathrm{e_4}^2|\mathrm{e_4}|^2, \pm 16{k_4}^2\hat{\mu}^2|e_0|^{2a}\,|\mathrm{e_4}|^{-1}\mathrm{e_3e_4}|\mathrm{e_4}|$, $\pm\frac{80\epsilon k_1{k_4}^2{k_3}^2\hat{\mu}^2}{(5+a)k_0k_2}|e_0|^{\frac{31a}{10}}\mathrm{e_0e_1}|e_0|^{\frac{-5+a}{10}}, \pm\frac{8(5-a)\epsilon\hat{\mu}^7k_1{k_4}^2{k_3}^2}{(5+a)k_2}|e_0|^{\frac{32a}{10}}$ y $\pm\frac{8k_1{k_4}^2{k_3}^2\hat{\mu}^7}{k_2}|e_0|^{\frac{17a}{5}}$. Asimismo, los términos dependientes de μ son reemplazados por sus similares dependientes de $\hat{\mu}$ a fin de formar la función $z^T\tilde{Q}(\hat{\mu})z$ y se aplica la desigualdad $e_i \le |e_i|$:

$$\dot{V}_0(z) \le -z^T\tilde{Q}(\hat{\mu})z - \frac{8(5-a)\epsilon\hat{\mu}^7 k_1 {k_4}^2 {k_3}^2}{(5+a)k_2}|e_0|^{\frac{32a}{10}} - \frac{8k_1{k_4}^2{k_3}^2\hat{\mu}^7}{k_2}|e_0|^{\frac{17a}{5}}$$
$$-\frac{4(5+a){k_4}^2{k_3}^2\mu^3}{5k_2}|e_0|^{3a}|e_1|^2 + \frac{4(5+a){k_4}^2{k_3}^2\hat{\mu}^3}{5k_2}|e_0|^{3a}|e_1|^2 - \frac{8(5-a)\epsilon k_1{k_4}^2{k_3}^2}{(5+a)\mu k_0{k_2}^2}|e_0|^{\frac{28a}{10}}|e_2|^2$$
$$+\frac{8(5-a)\epsilon k_1{k_4}^2{k_3}^2}{(5+a)\hat{\mu}k_0{k_2}^2}|e_0|^{\frac{28a}{10}}|e_2|^2 - \frac{8{k_4}^2k_3\mu}{k_2}|e_0|^{\frac{11a}{5}}|e_3|^2 + \frac{8{k_4}^2k_3\hat{\mu}}{k_2}|e_0|^{\frac{11a}{5}}|e_3|^2$$
$$+\frac{(20+4a)k_0{k_4}^2{k_3}^2\mu^4}{5k_1k_2}|e_0|^{\frac{28a}{10}}|e_1||e_2| - \frac{(20+4a)k_0{k_4}^2{k_3}^2\hat{\mu}^4}{5k_1k_2}|e_0|^{\frac{28a}{10}}|e_1||e_2|$$
$$+\frac{8(5-a)\epsilon{k_1}^2{k_4}^2{k_3}^2}{(5+a)k_0{k_2}^3}|e_0|^{\frac{27a}{10}}|e_2||e_3| - \frac{8(5-a)\epsilon{k_1}^2{k_4}^2{k_3}^2}{(5+a)k_0{k_2}^3}|e_0|^{\frac{27a}{10}}|e_2||e_3|$$
$$-\frac{8(\epsilon-1)\mu^7k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{32a+10}{10}}|e_0|^2|e_0|^{\frac{-5+a}{5}} + \frac{8(\epsilon-1)\hat{\mu}^7k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{32a+10}{10}}|e_0|^2|e_0|^{\frac{-5+a}{5}}$$
$$-4k_4\mu|e_0|^a|e_4|^{-1}|e_4|^2|e_4|^2 + 4k_4\hat{\mu}|e_0|^a|e_4|^{-1}|e_4|^2|e_4|^2 + 16{k_4}^2\mu^2|e_0|^{2a}\,|e_4|^{-1}|e_3||e_4||e_4|$$
$$-16{k_4}^2\hat{\mu}^2|e_0|^{2a}\,|e_4|^{-1}|e_3||e_4||e_4| + \frac{80\epsilon k_1{k_4}^2{k_3}^2\mu^2}{(5+a)k_0k_2}|e_0|^{\frac{31a}{10}}|e_0||e_1||e_0|^{\frac{-5+a}{10}}$$
$$-\frac{80\epsilon k_1{k_4}^2{k_3}^2\hat{\mu}^2}{(5+a)k_0k_2}|e_0|^{\frac{31a}{10}}|e_0||e_1||e_0|^{\frac{-5+a}{10}}$$

El resto de términos son agrupados para formar los términos dependientes del error $\tilde{\mu}$ y también se reemplaza $|e_{i+1}| = k_i\mu^{5-i}|e_0|^{\frac{(i+1)a}{5}}$ cuando corresponda, obteniéndose lo siguiente:

$$\dot{V}_0(z) \le -z^T\tilde{Q}(\hat{\mu})z - \hat{\mu}^7\left(\frac{8(5-a)\epsilon k_1{k_4}^2{k_3}^2}{(5+a)k_2}|e_0|^{\frac{32a}{10}} + \frac{8k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{17a}{5}}\right)$$
$$-\tilde{\mu}\beta_{13}\frac{4(5+a){k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{17a}{5}} - \tilde{\mu}\beta_7\frac{8(5-a)\epsilon{k_1}^3{k_4}^2{k_3}^2}{(5+a)k_0{k_2}^2}|e_0|^{\frac{36a}{10}} - \tilde{\mu}\beta_7\frac{8{k_2}^2{k_4}^2k_3}{k_2}|e_0|^{\frac{17a}{5}}$$
$$+\tilde{\mu}\beta_{13}\frac{(20+4a){k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{34a}{10}} + \tilde{\mu}\beta_7\frac{8(5-a)\epsilon{k_1}^3{k_4}^2{k_3}^2}{(5+a)k_0{k_2}^2}|e_0|^{\frac{37a}{10}}$$
$$-\tilde{\mu}\beta_7\frac{8(\epsilon-1)k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{34a+20}{10}} - \tilde{\mu}\beta_7 4k_4{k_3}^3|e_0|^{\frac{17a}{5}} + \tilde{\mu}\beta_7 16k_2k_3{k_4}^2|e_0|^{\frac{17a}{5}}$$
$$+\tilde{\mu}\beta_7\frac{80\epsilon k_1{k_4}^2{k_3}^2}{(5+a)k_2}|e_0|^{\frac{34a+5}{10}}$$

$$\dot{V}_0(z) = -z^T\tilde{Q}(\hat{\gamma})z - \hat{\mu}^{17}\Gamma + \tilde{\mu}(\beta_{13}\varphi_1 + \beta_7\varphi_2)$$

donde,

$\beta_{13} = \mu^{12} + \mu^{11}\hat{\mu} + \mu^{10}\hat{\mu}^2 + \mu^9\hat{\mu}^3 + \mu^8\hat{\mu}^4 + \mu^7\hat{\mu}^5 + \mu^6\hat{\mu}^6 + \mu^5\hat{\mu}^7 + \mu^4\hat{\mu}^8 + \mu^3\hat{\mu}^9 + \mu^2\hat{\mu}^{10} + \mu\hat{\mu}^{11} + \hat{\mu}^{12}$

$\beta_7 = \mu^6 + \mu^5\hat{\mu} + \mu^4\hat{\mu}^2 + \mu^3\hat{\mu}^3 + \mu^2\hat{\mu}^4 + \mu\hat{\mu}^5 + \hat{\mu}^6$

$$\varphi_1 = (-1 + |e_0|^{\frac{17a}{5}})\frac{(20+4a){k_0}^2{k_4}^2{k_3}^2}{5k_2}|e_0|^{\frac{17a}{5}}$$

$$\varphi_2 = -\frac{8(5-a)\epsilon{k_1}^3{k_4}^2{k_3}^2}{(5+a)k_0{k_2}^2}|e_0|^{\frac{36a}{10}} - \frac{8{k_2}^2{k_4}^2k_3}{k_2}|e_0|^{\frac{17a}{5}} + \frac{8(5-a)\epsilon{k_1}^3{k_4}^2{k_3}^2}{(5+a)k_0{k_2}^2}|e_0|^{\frac{37a}{10}}$$
$$-\frac{8(\epsilon-1)k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{34a+20}{10}} - 4k_4{k_3}^3|e_0|^{\frac{17a}{5}} + 16k_2k_3{k_4}^2|e_0|^{\frac{17a}{5}} + \frac{80\epsilon k_1{k_4}^2{k_3}^2}{(5+a)k_2}|e_0|^{\frac{34a+5}{10}}$$

$$\Gamma = \frac{8(5-a)\epsilon k_1{k_4}^2{k_3}^2}{(5+a)k_2}|e_0|^{\frac{32a}{10}} + \frac{8k_1{k_4}^2{k_3}^2}{k_2}|e_0|^{\frac{17a}{5}} > 0$$

Luego, la función $\dot{V}(z)$ queda representada de la siguiente manera:

$$\dot{V}(z) = \dot{V}_0(z) - \frac{1}{w}\tilde{\gamma}\dot{\hat{\gamma}} = -z^T\tilde{Q}(\hat{\gamma})z - \hat{\mu}^{17}\Gamma + \tilde{\mu}(\beta_{13}\varphi_1 + \beta_7\varphi_2 - \frac{1}{w}\dot{\hat{\mu}}) < 0$$

Luego, la ley de adaptación para el parámetro γ es igual a:

$$\dot{\hat{\mu}} = w_\mu(\beta_{13}\varphi_1 + \beta_7\varphi_2)$$

donde, la constante w puede ser escogida $w_\mu = max\{\frac{1}{\beta_{13}}, \frac{1}{\beta_7}\}$ a fin de eliminar el efecto de los parámetros desconocidos β_{13} y β_7 sobre la ley de adaptación $\dot{\hat{\mu}}$. Como se puede observar, la ley de adaptación para el parámetro μ posee la misma estructura que para el parámetro γ, haciendo que $V(z) > 0$ y $\dot{V}(z) = -z^T \tilde{Q}(\hat{\gamma}) z < 0$ cumpla para todo $z \in \mathbb{R}^5, z \neq 0$.

Finalmente, se propone el siguiente sistema de estructura variable adaptativo para el diferenciador:

Para $\lvert e_0 \rvert \geq 1$	Para $\varepsilon \leq \lvert e_0 \rvert < 1$	Para $\lvert e_0 \rvert < \varepsilon$
$\theta = 0$	$\theta = 1$	$\theta = 1$
$\dot{\hat{\gamma}} = 0$	$\dot{\hat{\gamma}} = w_\gamma \beta_{17} \varphi$	$\dot{\hat{\gamma}} = -w_\gamma \beta_{17} \varphi$
$\dot{\hat{\mu}} = w_\mu(\beta_{13}\varphi_1 + \beta_7 \varphi_2)$	$\dot{\hat{\mu}} = 0$	$\dot{\hat{\mu}} = 0$

La presente ley de adaptación, para $\lvert e_0 \rvert \geq 1$, activa los términos de orden alto del diferenciador para la convergencia uniforme en tiempo finito de este mientras mantiene el parámetro $\hat{\gamma}$ constante. Esto es especialmente útil cuando se requiere una convergencia rápida ante la presencia de perturbaciones senoidales y cosenoidales que varían su frecuencia, amplitud y fase en el tiempo. Por otro lado, cuando $\varepsilon \leq \lvert e_0 \rvert < 1$, el error se encuentra dentro de la capa límite de ancho $\delta - \varepsilon$, donde ε es un parámetro de diseño ligado al valor promedio de la amplitud del ruido que se espera que posea la señal $f(t)$. Por ello, la ley de adaptación, con la finalidad de ofrecer una convergencia exacta, aumenta el valor del parámetro $\hat{\gamma}$, mientras los términos de orden alto son eliminados al hacer $\mu = 0$. Esto es útil cuando el blanco se encuentra en trayectos lineales o efectúa cambios de rumbo a fuerzas "g" bajas. Por último, si $\lvert e_0 \rvert < \varepsilon$ se reduce el parámetro $\hat{\gamma}$ a fin de evitar una sobre estimación de la perturbación, mientras los términos de orden alto no tienen ningún efecto dado que permanece $\mu = 0$. Cabe resaltar que según lo descrito por Shtessel et al. [93], las leyes de adaptación con reducción de ganancia no poseen la capacidad de converger al origen del plano de estados. Sin embargo, la ventaja que posee es tipo de leyes de adaptación es evitar la sobre estimación del límite superior de la perturbación.

Para la implementación en MATLAB®, se propone la siguiente versión discretizada del diferenciador para la coordenada "x" (coordenada "y" y "z" son idénticas):

$$z_{x_0 k+1} = z_{x_0 k} + Tk_0(\theta\hat{\gamma}\left|z_{x_0 k} - f_{x_k}\right|^{\frac{4}{5}} + (1-\theta)\hat{\mu}^5\left|z_{x_0 k} - f_{x_k}\right|^{\frac{(5+a)}{5}}) + \sum_{j=1}^{4}\frac{T^j}{j!}z_{x_{j_k}}$$

$$z_{x_1 k+1} = z_{x_1 k} + Tk_1(\theta\hat{\gamma}^2\left|z_{x_0 k} - f_{x_k}\right|^{\frac{3}{5}} + (1-\theta)\hat{\mu}^4\left|z_{x_0 k} - f_{x_k}\right|^{\frac{(5+2a)}{5}}) + \sum_{j=1}^{3}\frac{T^j}{j!}z_{x_{j+1_k}}$$

$$z_{x_2 k+1} = z_{x_2 k} + Tk_2(\theta\hat{\gamma}^3\left|z_{x_0 k} - f_{x_k}\right|^{\frac{2}{5}} + (1-\theta)\hat{\mu}^3\left|z_{x_0 k} - f_{x_k}\right|^{\frac{(5+3a)}{5}}) + \sum_{j=1}^{2}\frac{T^j}{j!}z_{x_{j+2_k}}$$

$$z_{x_3 k+1} = z_{x_3 k} + Tk_3(\theta\hat{\gamma}^4\left|z_{x_0 k} - f_{x_k}\right|^{\frac{1}{5}} + (1-\theta)\hat{\mu}^2\left|z_{x_0 k} - f_{x_k}\right|^{\frac{(5+4a)}{5}}) + \sum_{j=1}^{1}\frac{T^j}{j!}z_{x_{j+3_k}}$$

$$z_{x_4 k+1} = z_{x_4 k} + Tk_4(\theta\hat{\gamma}^5 + (1-\theta)\hat{\mu}\left|z_{x_0 k} - f_{x_k}\right|^{1+a})sign(z_{x_0 k} - f_{x_k})$$

donde $\sum_{j=1}^{4}\frac{T^j}{j!}z_{x_{j_k}} = Tz_{x_1 k} + \frac{T^2}{2}z_{x_2 k} + \frac{T^3}{6}z_{x_3 k} + \frac{T^4}{24}z_{x_4 k}$, $\sum_{j=1}^{3}\frac{T^j}{j!}z_{x_{j+1_k}} = Tz_{x_2 k} + \frac{T^2}{2}z_{x_3 k} + \frac{T^3}{6}z_{x_4 k}$, $\sum_{j=1}^{2}\frac{T^j}{j!}z_{x_{j+2_k}} = Tz_{x_3 k} + \frac{T^2}{2}z_{x_4 k}$ y $\sum_{j=1}^{1}\frac{T^j}{j!}z_{x_{j+3_k}} = Tz_{x_4 k}$ son las aproximaciones de Taylor del diferenciador y f_{x_k} es la posición medida del blanco discretizada. A continuación, se presentan los parámetros utilizados en el diseño del presente diferenciador.

ARED	ε	w_γ	w_μ	β_{17}	β_{13}	β_7	a	$K0$	$K1$	$K2$	$K3$	$K4$
C. x	0.3	5e-4	1.5e-7	17	17.8	9.6	0.08	4.62	8.42	14.88	7.31	1.10
C. y	0.4	5e-4	2e-7	17	13	7	0.08	5.50	10.0	9.30	4.57	1.10
C. z	0.3	1.5e-4	2e-7	17	13	7	0.08	5.50	9.03	17.02	4.57	1.21

Tabla 3.4.: Parámetros sintonizados del diferenciador robusto exacto adaptativo (ARED).
Fuente: Elaboración propia.

3.9.3 Algoritmo del diferenciador robusto exacto adaptativo (ARED)

A continuación, se presenta el algoritmo del programa del apéndice 2.11 el cual permite la implementación del diferenciador robusto exacto adaptativo:

Algoritmo 3.9: Diferenciador Robusto Exacto Adaptativo (ARED)

Entradas: Vectores $\mathbf{y_k}$ y tiempo de muestreo ***T***
Salidas: VE estimadas $\boldsymbol{xh_k}$, entrada de control estimada del modelo $\boldsymbol{uh_k}$, Derivada de entrada de control estimada del modelo $\boldsymbol{duh_k}$, colectores deslizantes $\boldsymbol{sig_k}$, constantes de Lipschitz adaptadas $\mathbf{L_k}$ $\boldsymbol{y}$ constantes uniformes adaptadas $\boldsymbol{\mu_k}$
Inicio del Programa: Invocado por el programa sim_ARED.m
1: Declarar e inicializa variables de estado estimadas de posición, velocidad, aceleración, sobre aceleración y derivada de la sobre aceleración de la coordenada "x" ($\boldsymbol{z0x, z1x, z2x, z3x}$ $\boldsymbol{y}$ $\boldsymbol{z4x}$), de la coordenada "y" ($\boldsymbol{z0y, z1y, z2y, z3y}$ $\boldsymbol{y}$ $\boldsymbol{z4y}$) y coordenada "z" ($\boldsymbol{z0z, z1z, z2z, z3z}$ $\boldsymbol{y}$ $\boldsymbol{z4z}$)**.**
2: Definir constante de Lipschitz inicial de las coordenadas "x", "y" y "z".
3: Definir el orden ***n=4*** del diferenciador de las coordenadas "x","y" y "z"
4: Definir parámetros para ley de adaptación de las coordenadas "x","y" y "z" (según tabla 3.4)
5: Definir las ganancias homogéneas del diferenciador de la coordenada "x", "y" y "z" **k0x, k1x, k2x, k3x** y **k4y, k0y, k1y, k2y, k3y** y **k4y y k0z, k1z, k2z, k3z** y **k4z.**
6: Establecer las series de Taylor para discretización del diferenciador
7: Establecer las variables deslizantes **sigx, sigy y sigz y sigdx, sigdy y sigdz (estás solo para gráfica de plano de estados)**
8: Adaptación de constante de Lipschitz y constante uniforme (según ley 3.147)
9: Hallar variables de estado estimadas futuras **z0x_1, z1x_1, z2x_1, z3x_1** y **z4x_1, z0y_1, z1y_1, z2y_1, z3y_1** y **z4y_1,** y **z0z_1, z1z_1, z2z_1, z3z_1** y **z4z_1.**
10: Guardar variables para siguiente iteración.
11: Entrega $\boldsymbol{xh_k} = [\boldsymbol{z0x; z1x; z2x; z0y; z1y; z2y; z0z; z1z; z2z}]$
12: Entrega $\boldsymbol{uh_k} = [\boldsymbol{z3x; z3y; z3z}]$
13: Entrega $\boldsymbol{duh_k} = [\boldsymbol{z4x; z4y; z4z}]$
14: Entrega $\boldsymbol{sig_k} = [\mathbf{sigx; sigy; sigz; sigdx; sigdy; sigdz}]$
15: Entrega $\boldsymbol{L_k} = [\boldsymbol{Lx0\_1; Lx0\_1; Lx0\_1}]$
16: Entrega $\boldsymbol{\mu_k} = [\boldsymbol{mu0\_1; mu0\_1; mu0\_1}]$
17: Fin de programa

3.9.4 Simulación de la trayectoria del misil

Mediante el programa del apéndice 2.12 se efectuó las simulaciones con y sin ruido de la trayectoria del misil, siendo el algoritmo correspondiente detallado a continuación:

Algoritmo 3.10: Simulación de trayectoria del blanco con diferenciador ARED

Entradas: Señal de entrada de control $\boldsymbol{\mu_k}$, perturbaciones $\boldsymbol{\xi_k}$ y modelo mixto gaussiano (GMM)
Salidas: VE reales del modelo $\boldsymbol{x_k}$, VE medidas del modelo $\boldsymbol{y_k}$, **VE estimadas** $\boldsymbol{xh_k}$, entrada de control al modelo (sobre aceleración) estimado $\boldsymbol{uh_k}$ **y derivada** $\boldsymbol{duh_k}$, variables deslizantes $\boldsymbol{sig_k}$, constantes adaptadas de Lipschitz $\boldsymbol{L_k}$, constantes uniformes adaptadas $\boldsymbol{\mu_k}$, errores cartesianos $\boldsymbol{error\_c_k}$, errores polares $\boldsymbol{error\_c_k}$, error RMSE y gráficos
Inicio del programa:
1: Declara tiempo de muestreo ***T***, gravedad $\boldsymbol{g}$, constante de mach **M**, sobre aceleración máxima ***load_factor*** tiempo inicial de simulación $\boldsymbol{t_i}$, tiempo final de simulación t_{ff} y número total de muestras $\boldsymbol{nt}$.
2: Declara condiciones iniciales de traqueo al blanco: distancia $\boldsymbol{Ad}$, acimut verdadero $\boldsymbol{Bdn}$, elevación $\boldsymbol{Ed}$, velocidad $\boldsymbol{Vm}$ y rumbo verdadero $\boldsymbol{Rv}$.
3: Convierte posición de coordenadas polares a cartesianas usando $\boldsymbol{Ad, Bdn\ y\ Ed}$
4: Declara el vector de estados inicial $\boldsymbol{x_k(0)}$
5: Declara matrices del espacio de estados $\boldsymbol{F_k}$, $\boldsymbol{G_k}$, $\boldsymbol{D_k}$ y $\boldsymbol{C_k}$.
6: Obtén $\boldsymbol{u_k}$ (invoca a función ***gen_jerk_sea_skimming3.m*** **(programa del apéndice 2.5))**
7: Obtén $\boldsymbol{\xi_k}$ **(invoca** a función ***perturb_1.m*** **(programa del apéndice 2.6)**
8: Cargar ruido angular ***glint_puntos.mat*** **(datos guardados del programa del apéndice 2.7)**
9: Ingrese "1" para simulación con ruido y "0" para simulación sin ruido.
10: Para tt = ti:T:tff **Hacer**
11: Ejecuta el modelo de transición de estados para obtener $\boldsymbol{x_k}$ futuro.
12: Invoca ***c2p.m*** para convertir $\boldsymbol{x_k}$ a coordenadas polares.
13: Invoca ***yk_noise.m*** para obtener el vector de mediciones y_k
14: Invoca **ARED.m (programa del apéndice 2.11)** para obtener $\boldsymbol{xh_k}$, $\boldsymbol{uh_k}$, $\boldsymbol{duh_k}$, $\boldsymbol{sig_k}$, $\boldsymbol{L_k}$ $\boldsymbol{y}$ $\boldsymbol{\mu_k}$
15: Invoca **c2p.m** para convertir $\boldsymbol{xh_k}$ a coordenadas polares.
16: Extrae errores polares, errores cartesianos y RMSE
17: Almacena vectores para gráficas
18: Fin Para
19: Conversión de unidades del vector de estados a Mn, Mach y g's.
20: Graficar
21: Fin de programa

Al efectuar las simulaciones se observó que el diferenciador robusto exacto adaptativo en la ausencia de ruido (figura 3.47) estima correctamente las variables de estado del blanco, pero presenta sensibilidad al ruido (figura 3.42).

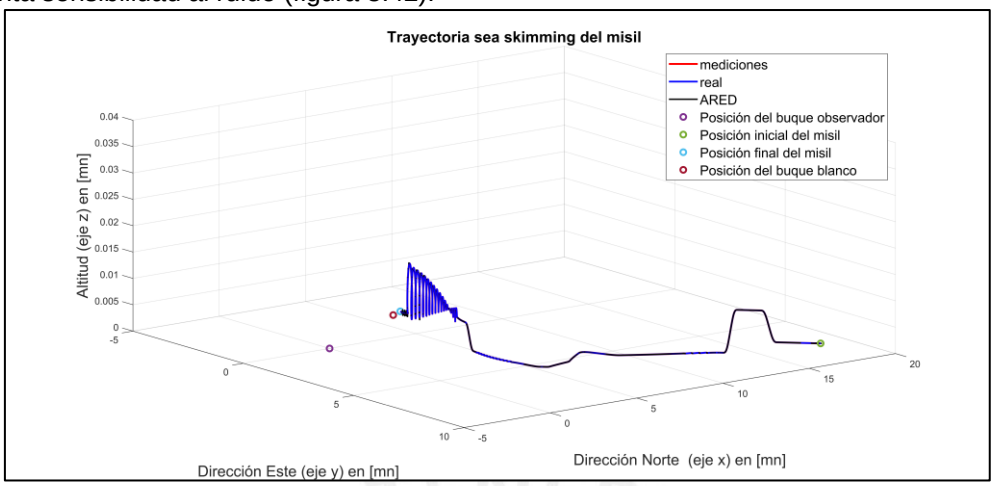

Figura 3.47.: Trayectoria completa del misil sin ruido (ARED).
Fuente: Elaboración propia.

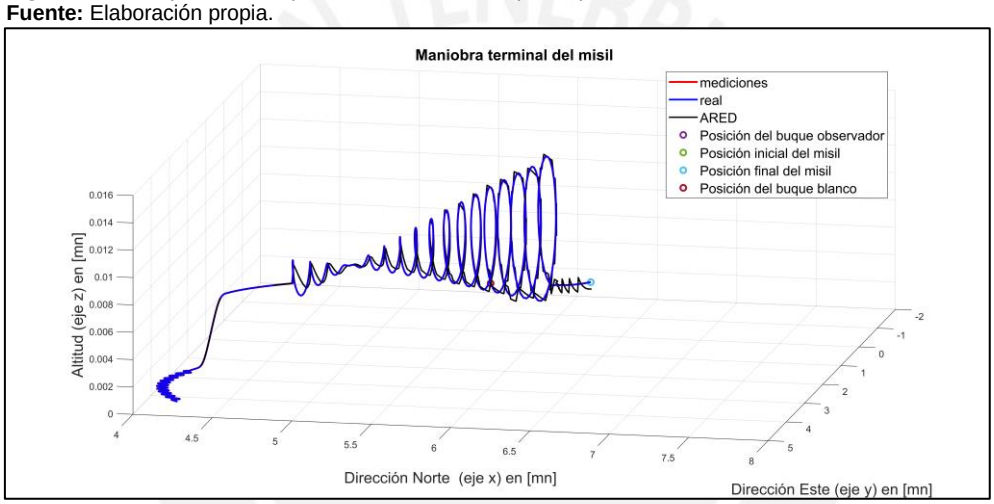

Figura 3.48.: Maniobra terminal del misil sin ruido (ARED).
Fuente: Elaboración propia.

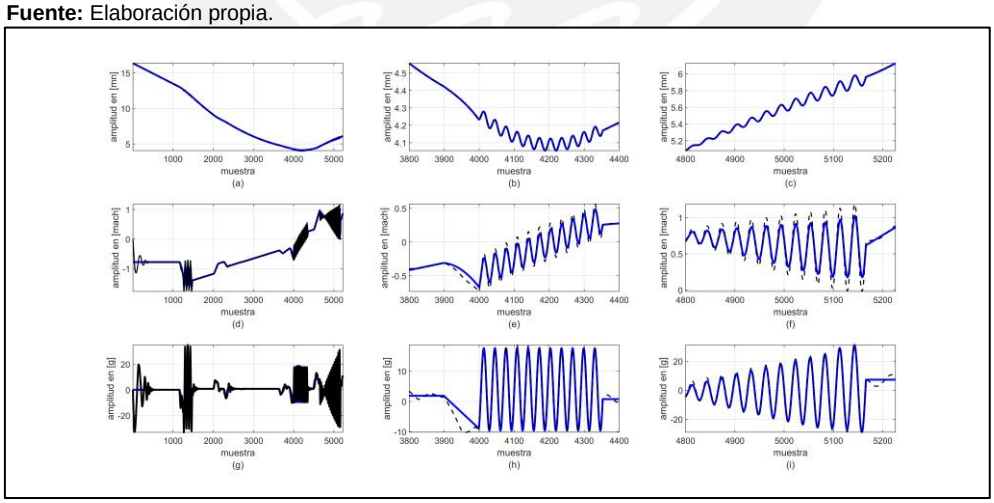

Figura 3.49.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "x" en ausencia de ruido (RED). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [3800 4400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [3800 4400]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [3800 4400]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

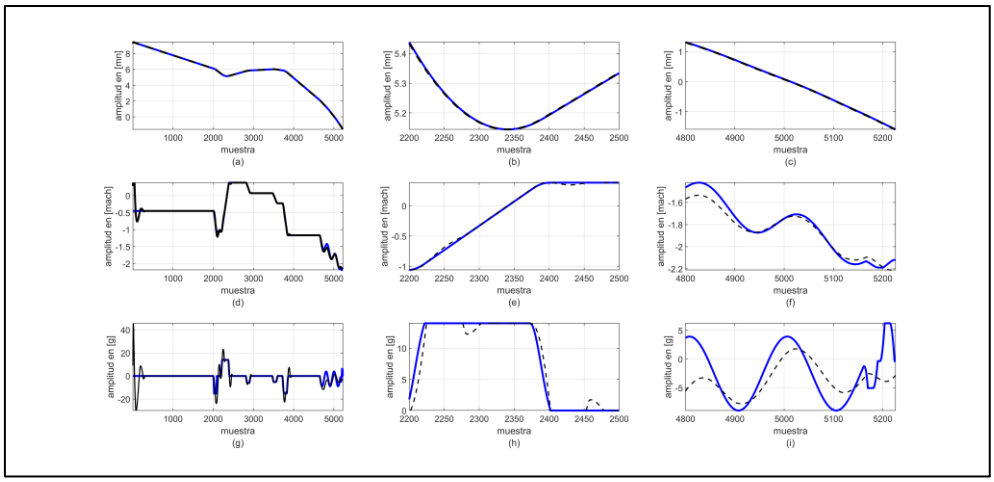

Figura 3.50.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "y" en la ausencia de ruido (RED). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [2200 2500]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [2200 2500]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [2200 2500]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

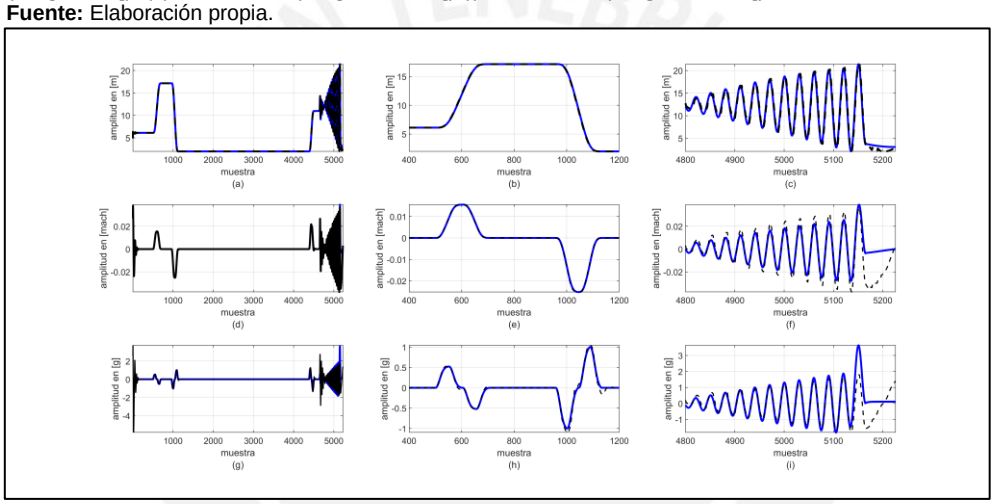

Figura 3.51.: Variables de estado de posición, velocidad y aceleración del misil en la coordenada "z" en la ausencia de ruido (RED). Variable de estado real (línea azul). Variable de estado estimada (línea negra punteada) (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Velocidad (k = [400 1200]) (f) Velocidad (k = [4800 5226]) (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [400 1200]) (i) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

Dado que el diferenciador diseñado posee la capacidad de estimar la sobre aceleración, esto permite que se pueda obtener el vector estimado de entrada de control $\boldsymbol{u_k}$ del modelo, debido a que la magnitud física de este vector es la sobre aceleración. Por ello, en la figura 3.48 se contrasta el vector de entrada de control real $\boldsymbol{u_k}$ y con la sobre aceleración estimada del diferenciador, denominada $\boldsymbol{\hat{u}_k}$, observándose que este detecta correctamente los monopulsos gaussianos del vector de control. Sin embargo, se observa que existe una perturbación de la estimación del diferenciador entre las muestras 1200 y 1500 y posteriormente a partir de la muestra 4000. Esto sucede debido a que tanto la señal de control como las perturbaciones consideradas para el modelo poseen unidades de sobre aceleración (m/s3) y el diferenciador no tiene la capacidad de discernir entre una u otra. En color magenta se graficó el vector de perturbaciones reales en la coordenada "y", teniendo un desempeño similar en las otras coordenadas. Por otro lado, en las figuras 3.49 y 3.50 se observa que los errores cartesianos y errores polares, respectivamente, se mantienen dentro de los límites establecidos, siendo menores que los errores del diferenciador robusto exacto estándar.

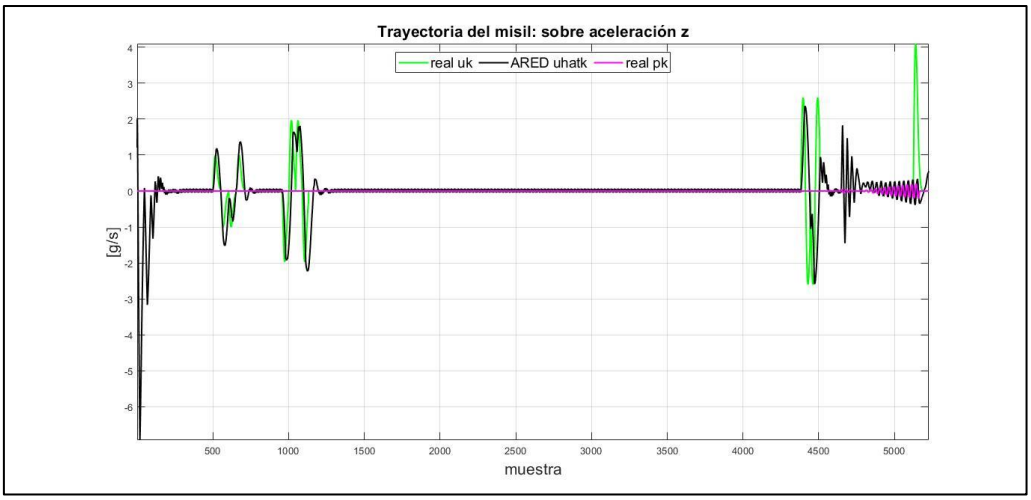

Figura 3.52.: Sobre aceleración estimada del misil en la coordenada “z” sin ruido (ARED).
Fuente: Elaboración propia.

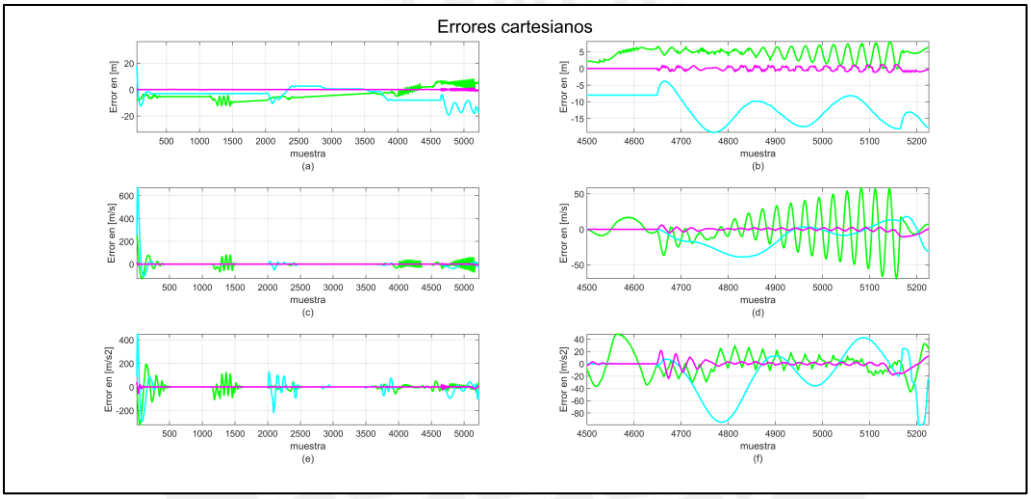

Figura 3.53.: Errores cartesianos (ARED). Coordenada “x” (línea verde). Coordenada “y” (línea cyan). Coordenada “z” (línea magenta). (a) Posición k = [0 5226] (b) Posición k = [4800 5226] (c) Velocidad k = [0 5226] (d) Velocidad k = [4800 5226] (e) Aceleración k = [0 5226] (f) Aceleración k = [4800 5226].
Fuente: Elaboración propia.

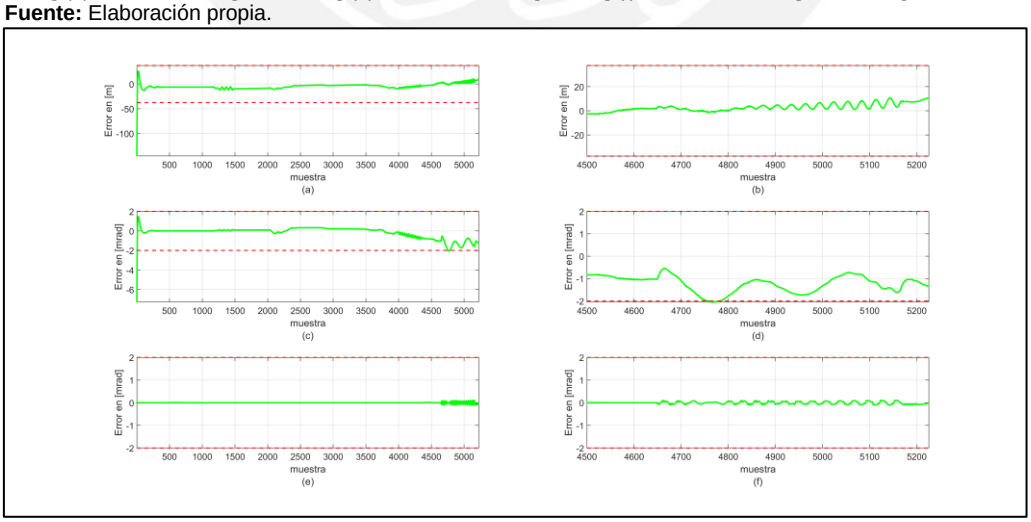

Figura 3.54.: Errores polares (ARED). (a) Error en alcance Ad (k = [0 5226]). (b) Error en alcance Ad (k = [4500 5226]). (c) Error en azimuth verdadero Bdn (k = [0 5226]). (d) Error en azimuth verdadero Bdn (k = [4500 5226]). (e) Error en elevación Ed (k = [0 5226]). (f) Error en elevación Ed (k = [4500 5226]).
Fuente: Elaboración propia.

Por último, en la figura 3.55 se muestra la adaptación de constante Lipschitz γ y la constante de uniformidad μ, así como la función de conmutación θ.

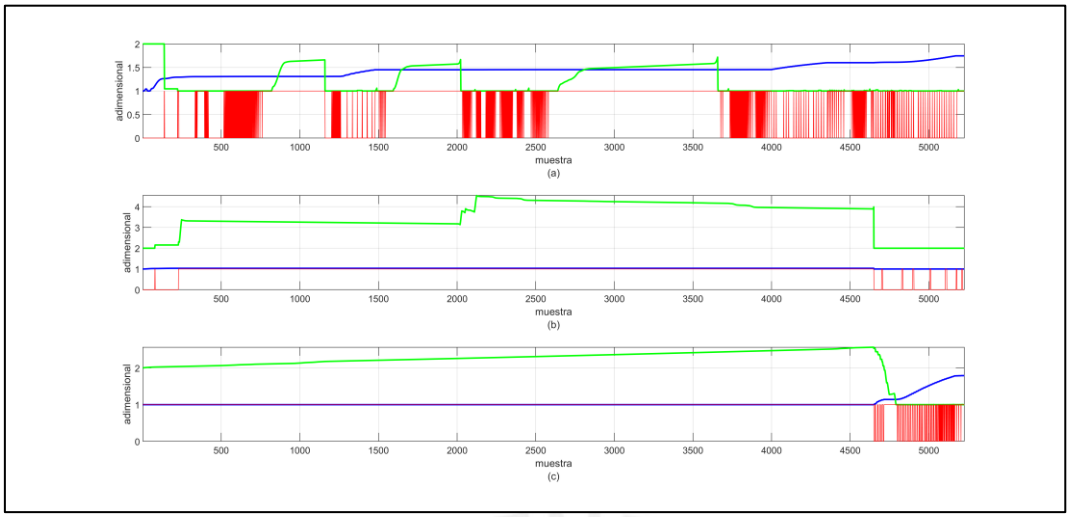

Figura 3.55.: Adaptación de constantes (ARED). Constante de Lipschitz (línea verde). Constante de Uniformidad (línea azul). Función de conmutación θ (línea roja). (a) Coordenada "x". (b) Coordenada "y". (c) Coordenada "z".
Fuente: Elaboración propia.

Se observó a partir de simulaciones que la adaptación de constante de Lipschitz e introducción de términos de orden alto mejora el desempeño del diferenciador robusto exacto en la ausencia de ruido. El diferenciador robusto exacto adaptativo mostró mejor exactitud y uniformidad en tiempo finito que en su versión estándar, logrando también aproximar el vector de entrada de control al modelo al obtener la sobre aceleración por medio de la operación de diferenciación robusta. Sin embargo, el diferenciador posee similar sensibilidad al ruido angular que el *observador Super Twisting de orden superior*. Por tanto, queda pendiente el diseño de un mecanismo de filtrado si es que se pretende utilizar cualquiera de los diferenciadores u observadores anteriormente expuestos para la tarea de estimación de variables de estado de un blanco aéreo de alta maniobrable un ambiente de ruido angular.

3.10 Diferenciador Robusto Exacto de filtrado estándar (REDF)

Como se ha mencionado anteriormente, la arquitectura de modelo múltiple interactivo (IMM) a base de filtros kalman es la solución del estado del arte para el problema en estudio, dado que provee una solución óptima al minimizar el error medio cuadrático cuando el proceso bajo observación es representado por un modelo exacto. Sin embargo, la efectividad de esta solución del estado del arte dependerá si los modelos utilizados son exactos respecto a todas las cinemáticas posibles del blanco aéreo de alta maniobrabilidad [33]. Por ello, si existen incertidumbres estructuradas o paramétricas en los modelos los filtros implementados no presentarán buen desempeño. Xie y Soh [94] estudian el problema de robustez que presenta el filtro Kalman y proveen una solución robusta al garantizar que la matriz de covarianza del error de estimación se encuentre siempre dentro de un límite de diseño establecido para todas las incertidumbres contempladas para el modelo. Otras soluciones al problema de robustez del filtro Kalman son propuestas en Agamennoni et al. [95], Luo y Bosch [96] y Zorzi [97].

Otro problema del uso de los filtros Kalman en aplicaciones de traqueo de blancos aéreos es que se asume que el ruido de medición está representado por una distribución gaussiana de media cero [33]. Claramente esta asunción no se acerca a la realidad debido a que el ruido angular posee una distribución no gaussiana, siendo aproximado de acuerdo a lo detallado en el capítulo II como una distribución de "cola larga" o *Long-Tailed distribution*. Por ello, para contrarrestar el efecto del ruido angular en el desempeño del filtro Kalman Masreliez [98,99] en el año de 1975 y 1977, respectivamente, propone teóricamente el uso de una función Score o *Score Function* efectuando la convolución de dos funciones de densidad, encontrando dificultades en su implementación en tiempo real. Posteriormente, Wu y Kunda [100] y Wu [101] proponen propone una función basada en una distribución normal adaptativa para la evaluación eficiente de la función *Score*, aplicándola a problemas de traqueo de blancos. Posteriormente, Daeipour y Bar-Shalom [32] en un intento de utilizar el algoritmo de traqueo IMM efectúa el modelamiento del ruido angular o glint por medio de una distribución mixta gaussiana, obteniendo resultados

satisfactorios. En lo que respecta al algoritmo Kalman lineal, este considera un modelo de transición de estados lineal,

$$x_{k+1} = \Phi_k X_k + w_k$$

Donde X_k es el vector de estados estimados, Φ_k es la matriz de transición del modelo del instante k al $k+1$, y w_k es el ruido de proceso del sistema asumido como una distribución gaussiana blanca de media cero. Por otro lado, asume que el modelo de observación es lineal respecto al vector de estados estimados [33] de la siguiente forma:

$$y_k = H_k X_k + v_k$$

Donde H_k es la matriz de observación y v_k es el ruido de medición asumido como una distribución gaussiana de media cero y que posee correlación cruzada igual a cero con el ruido de proceso [33]. Por tanto, las ecuaciones del filtro Kalman lineal son las siguientes:

$$\begin{aligned}
\tilde{X} &= \Phi\hat{X} \\
\tilde{P} &= \Phi\hat{P}\Phi^T + Q \\
K &= \tilde{P}\,H^T\left(H\tilde{P}H^T + R\right)^{-1} \\
\hat{X} &= \tilde{X} + K\left(y_k - H\tilde{X}\right) \\
\hat{P} &= (I - KH)\tilde{P}
\end{aligned}$$

Donde $\hat{X}$ es el vector de estados estimados predicha, $\tilde{X}$ es el vector de estados filtrados, $\hat{P}$ es la matriz de covarianza filtrada, $\tilde{P}$ es la matriz de covarianza predicha, Q es la matriz de ruido de proceso, K es la matriz de ganancias Kalman, R es la covarianza del ruido de medición y I es la matriz identidad. A partir del filtro Kalman lineal se propone el filtro de Kalman extendido (EKF) para la estimación de las variables de estado de un sistema no lineal, utilizándose el modelo linealizado de la planta o proceso al calcular la matriz jacobiana en cada iteración del algoritmo [102]; este algoritmo cuenta con las mismas ecuaciones que el algoritmo Kalman lineal, pero con una transición de estados $\tilde{X} = f(\hat{X})$., donde $f(\hat{X})$ es la función del modelo linealizada respecto a un punto de operación. Según Wan y Van Der Merwe [103] el filtro Kalman extendido posee problemas en la propagación de la variable aleatoria gaussiana a través de la dinámica del sistema cuando esta es linealizada respecto a un punto de operación, introduciendo errores que llevan a un desempeño sub óptimo y en algunos casos a la completa divergencia de los estados reales [103]. Por ello, Julier y Uhlman [104] proponen el filtro Kalman Uscented, el cual utiliza un muestreo determinístico seleccionando cuidadosamente algunos puntos denominados sigma que describen completamente la media y covarianza de la distribución gaussiana que, al ser propagados por el modelo no lineal del sistema, permiten estimar de manera exacta la media covarianza posterior [103]. Asimismo, existe el filtro Kalman Cubature el cual provee una solución sistemática al problema de filtrado de sistemas no lineales de dimensiones altas al utilizar una regla de cubículo esférico-radial, siendo aplicado a problemas de traqueo de aeronaves satisfactoriamente en Arasaratnam y Haykin [105]; el filtro Kalman Cubature es igual al filtro Kalman Uscented cuando *sigma=1, beta=1 y kappa=1* [105]. Cabe resaltar que ninguno de estos filtros provee soluciones al problema de ruido no gaussiano, por ello el filtro de partículas o *particle filter,* siendo su nombre propuesto por primera vez por Del Moral [106] en base a los algoritmos genéticos desarrollados por Holland [107], Cerf [108], Kunita [109] y Stettner [110] en la década de los años sesenta, provee una solución de estimación de un modelo dinámico no lineal y modelo de medición no gaussiano por medio de la aplicación de métodos secuenciales de Montecarlo [111].

Otro método de filtrado en tiempo real es el algoritmo de ventana adaptativa de primer orden (First Order Adaptive Window o FOAW por sus siglas en inglés), el cual fue propuesto por primera vez por Janabi-Sharifi et al. [112] como una técnica para efectuar el filtrado de las estimaciones de velocidad de un diferenciador de posición y así poder obtener una buena relación filtrado-retardo de tiempo por filtrado [113]. Si se considera los valores pasados y valor presente de la variable ruidosa a filtrar u_k iguales a $S = \{u_k, u_{k-1}, u_{k-2}, \ldots u_{k-n-1}, u_{k-n}\}$, donde n es la longitud de la ventana deslizante del algoritmo, y x_k es la derivada de la variable de entrada ruidosa u_k, el FOAW verifica en cada iteración que los puntos del conjunto *S* se encuentren dentro de una región rectangular de dimensiones $(u_{k-n} - u_k)\; x\; 2e_{max}$, donde e_{max} es la máxima desviación

permisible respecto a la línea que une los puntos u_{k-n} y u_k, aumentando la longitud de la ventana solo si y solo sí todos los puntos hasta el paso de verificación estén localizados dentro de la región; al encontrarse el primer punto dentro de la región aumenta en uno la longitud, al encontrarse el segundo punto dentro de la región aumenta en dos la longitud y así sucesivamente, permitiendo obtener estimaciones más suaves de la variable diferenciada x_k [112]. Por otro lado, si hay algún punto que durante la verificación se encuentra fuera de la región, el algoritmo se trunca y la longitud de la ventana se mantiene hasta que un nuevo grupo de puntos es analizado; al truncarse la ventana el algoritmo efectúa la diferenciación del último punto del conjunto *S* que se encontraba dentro de la región para así poder retornar un valor x_k apropiado. En la figura se puede observar que el punto u_{k-i} del conjunto *S* se encuentra fuera de la región rectangular, por ello, el algoritmo trunca la longitud de la ventana en un valor de $n - i$. Cabe resaltar que esta técnica ha mostrado buenos resultados en [113] pero no existen estudios que verifiquen su desempeño en casos de filtrado de ruido angular.

Dado que se requiere filtrar en todo momento las mediciones brindadas por el radar, en el campo de filtrado por modos deslizantes solo se conoce el filtro parabólico de modos deslizantes con FOAW propuesto por Jin et al. [113], el diferenciador de primer orden con capacidad de filtrado de modos deslizantes de primer orden de Kikuuwe et. al [114] y el diferenciador robusto exacto de filtrado estándar de Levant [80]. Si bien es cierto en [113] y [114] se logra el filtrado de señales con ruido gaussiano satisfactoriamente, este tipo de filtros no ofrecerían buenos resultados para el problema en estudio debido a que implementan filtros de paso bajo que al ser aplicados a las mediciones de posición de radar no tendrían ningún efecto sobre el ruido angular, dado que este se encuentra por debajo de los 10 Hz según los datos gráficos provistos en el capítulo 9 de Skolnik [1]. Por ello, para el filtrado de señales por modos deslizantes se ha enfocado el estudio del diferenciador robusto exacto de filtrado estándar de Levant.

3.10.1 Fundamentos teóricos

Asunción 3.1 (Señales filtradas muestreadas continuamente [80]): *La función desconocida $f_0(t), t \geq 0$, se encuentra disponible en tiempo real por medio de su aproximación ruidosa $f(t) = f_0(t) + n(t) + n_c(t)$. Se conoce que la derivada ${f_0}^k(t)$ existe y posee la constante de Lipschitz conocida $L > 0$. El primer componente de ruido $n(t)$ puede ser medido por Lebesgue y esencialmente acotado, esto es, $|n(t)| \leq \delta$, para un parámetro desconocido $\delta \geq 0$.*

Asunción 3.2 (Señales filtradas muestreadas continuamente [80]): *La segunda componente del ruido $n_c(t)$ es medible por Lebesgue y centrada aproximadamente en cero. Por tanto, $n_c(t)$ debe ser una función integrable que cumpla la desigualdad $\left|\int_0^t n_c(s)ds\right| \leq \varepsilon$ para cualquier $t \geq 0$ y un parámetro desconocido $\varepsilon \geq 0$.*

Asunción 3.3 (Señales filtradas muestreadas discretamente [80]): *La función $f(t)$ es muestreada en los instantes $t_0, t_1, t_2, ..$ donde $t_0 = 0, 0 < t_{j+1} - t_j < \tau_j \leq \tau$. El ruido $n_c(t_j)$ se encuentra aproximadamente centrado en cero, cumpliéndose $\left|\sum_{s=0}^{j} n_c(t_s)\tau_s\right| \leq \varepsilon$ para cualquier $j \geq 0$ y un parámetro desconocido $\varepsilon \geq 0$.*

Definición 3.21 (Función globalmente filtrable [80]): *Una función $v(t), v: [0,\infty) \to \mathbb{R}$ es llamada globalmente filtrable o una señal del orden global de filtrado $k \geq 0$, si v es una función de medida de Lebesgue localmente integrable y existe también una solución $\xi(t)$, $\xi: [0,\infty) \to \mathbb{R}$ local absolutamente continua y uniformemente acotada de la ecuación $\xi^{(k)} = v$. Cualquier número que excede al valor correspondiente a $sup|\xi(t)|$ es denominado una magnitud integral de v de orden k.*

Definición 3.22 (Función localmente filtrable [80]: *Una función de medida de Lebesgue local integrable $v(t), v: [0,\infty) \to \mathbb{R}$ es llamada localmente filtrable si existe un entero $k > 0$ y números $T > 0$ y $a_0, a_1, a_2, \ldots, a_{k-1} \geq 0$, tal que para cualquier $t_1 \geq 0$ existe una solución $\xi(t)$, $t \in [t_1, t_1 + T]$, de la ecuación $\xi^{(k)} = v$ que satisface $|\xi^l(t)| \leq a_l$ para $l = 0,1,\ldots,k-1$. Los números a_l son denominados las magnitudes integrales de v de orden k. Las señales de orden de filtrado local 0 son trivialmente definidas como señales acotadas de magnitud a_0*

De las definiciones anteriores, la estructura no recursiva del diferenciador robusto exacto de filtrado estándar en su versión continua es la siguiente:

$$\begin{aligned}
\dot{w}_1 &= -\tilde{\lambda}_{n+n_f} L^{\frac{1}{n+n_f+1}} \lfloor w_1 \rceil^{\frac{n+n_f}{n+n_f+1}} + w_2 \\
\dot{w}_{n_f-1} &= -\tilde{\lambda}_{n+2} L^{\frac{n_f-1}{n+n_f+1}} \lfloor w_1 \rceil^{\frac{n+2}{n+n_f+1}} + w_{n_f} \\
\dot{w}_{n_f} &= -\tilde{\lambda}_{n+1} L^{\frac{n_f}{n+n_f+1}} \lfloor w_1 \rceil^{\frac{n+1}{n+n_f+1}} + z_0 - f(t) \\
\dot{z}_0 &= -\tilde{\lambda}_n L^{\frac{n_f+1}{n+n_f+1}} \lfloor w_1 \rceil^{\frac{n}{n+n_f+1}} + z_1 \\
\dot{z}_{n-1} &= -\tilde{\lambda}_1 L^{\frac{n+n_f}{n+n_f+1}} \lfloor w_1 \rceil^{\frac{1}{n+n_f+1}} + z_n \\
\dot{z}_n &= -\tilde{\lambda}_0 L sign(w_1)
\end{aligned} \tag{3.149}$$

donde w_1 es una variable virtual, $\dot{w}_{n_f}$ es el error de posición filtrado, $z_0 - f(t)$ es el error de posición sin filtrado, $\mathrm{n_f} \geq 0$ es el orden de filtrado y n es el orden diferenciación. Cabe resaltar que para la elección de las ganancias del diferenciador se puede utilizar la tabla 3.1 con un orden del diferenciador igual a $\mathrm{n} + \mathrm{n_f}$. Asimismo, la versión discretizada es similar al sistema discretizado que figura en 3.141. Cabe resaltar que el sistema 3.149 del orden de filtrado $n_f \geq 0$ es robusto respecto a posibles ruidos desacotados de los ordenes de filtrado que no excedan el valor n_f.

3.10.2 Diseño del diferenciador robusto exacto de filtrado estándar (REDF)

El objetivo es efectuar el filtrado de las mediciones ruidosas de posición del blanco, a fin de lograr obtener variables de estado estimadas libre de ruido, y lograr compensar adecuadamente las incertidumbres estructuradas introducidas al modelo lineal incierto del blanco. Sin embargo, en el apartado anterior se observó que es necesario efectuar la adaptación de la constante de Lipschitz e introducir términos de super torsión o *Super Twisting* de orden alto para poder obtener derivadas exactas y uniformes de la posición. Por tanto, si para el diseño del diferenciador robusto exacto de filtrado estándar no se contempla la adaptación de la constante Lipschitz y se introducen los términos de orden alto, no se logrará obtener variables de estado filtradas exactas y uniformes de la velocidad, aceleración y sobre aceleración del blanco para la aplicación en estudio. Con la finalidad de comprobar que, para el problema particular en estudio, el diferenciador robusto exacto de filtrado estándar de Levant no posee una buena relación de filtrado de ruido - robustez ante incertidumbres estructuradas con constante de Lipschitz variante en el tiempo, se procedió a diseñar el diferenciador de la siguiente forma.

Si se considera el sistema dinámico lineal incierto en espacio de estados del sistema discreto planteado en (2.23) y sea $f_x(t)$, $f_y(t)$ y $f_z(t)$ la posición medida del blanco en la coordenada "x", coordenada "y" y coordenada "z", respectivamente, obtenidas del modelo de medición $y_k = H_k x_k + v_k$, el diferenciador robusto exacto de filtrado estándar con orden de filtrado $n_f = 3$ y orden de diferenciación $n = 4$ continuo en la coordenada "x" se deduce de acuerdo al siguiente detalle:

$$\begin{aligned}
\dot{w}_{x_1} &= -\tilde{\lambda}_7 L^{\frac{1}{8}} \lfloor w_1 \rceil^{\frac{7}{8}} + w_{x_2} \\
\dot{w}_{x_2} &= -\tilde{\lambda}_6 L^{\frac{2}{8}} \lfloor w_1 \rceil^{\frac{6}{8}} + w_{x_3} \\
\dot{w}_{x_3} &= -\tilde{\lambda}_5 L^{\frac{3}{8}} \lfloor w_1 \rceil^{\frac{5}{8}} + z_{x_0} - f_x(t) \\
\dot{z}_{x_0} &= -\tilde{\lambda}_4 L^{\frac{4}{8}} \lfloor w_1 \rceil^{\frac{4}{8}} + z_{x_1} \\
\dot{z}_{x_1} &= -\tilde{\lambda}_3 L^{\frac{5}{8}} \lfloor w_1 \rceil^{\frac{3}{8}} + z_{x_2} \\
\dot{z}_{x_2} &= -\tilde{\lambda}_2 L^{\frac{6}{8}} \lfloor w_1 \rceil^{\frac{2}{8}} + z_{x_3} \\
\dot{z}_{x_3} &= -\tilde{\lambda}_1 L^{\frac{7}{8}} \lfloor w_1 \rceil^{\frac{1}{8}} + z_{x_4} \\
\dot{z}_{x_4} &= -\tilde{\lambda}_0 L sign(w_1)
\end{aligned}$$

Donde, $f_x(t)$ es la versión continua de posición medida de la coordenada x, z_{x_0} es la estimación de la posición en la coordenada x, L es la constante de Lipschitz, los parámetros $\tilde{\lambda}_0$, $\tilde{\lambda}_1$, $\tilde{\lambda}_2$, $\tilde{\lambda}_3$, $\tilde{\lambda}_4$, $\tilde{\lambda}_5$, $\tilde{\lambda}_6$ y $\tilde{\lambda}_7$ son las ganancias homogéneas del diferenciador de acuerdo a la tabla 3.3 y w_{x_1}, w_{x_2} y w_{x_3} son variables virtuales de filtrado del error de posición. Cabe resaltar que los diferenciadores para la coordenada "z" e "y" son idénticos al diferenciador de la coordenada "x", a excepción de la constante L de Lipschitz que se modificará de acuerdo al límite superior de la perturbación en cada coordenada. Asimismo, de acuerdo a lo descrito en Livne y Levant [86] el diferenciador en su forma discreta tiene la siguiente forma:

$$
\begin{aligned}
w_{x_{1_{k+1}}} &= w_{x_{1_k}} + T(-\tilde{\lambda}_7 L^{\frac{1}{8}} \lfloor w_1 \rceil^{\frac{7}{8}} + w_{x_2}) \\
w_{x_{2_{k+1}}} &= w_{x_{2_k}} + T(-\tilde{\lambda}_6 L^{\frac{2}{8}} \lfloor w_1 \rceil^{\frac{6}{8}} + w_{x_3}) \\
w_{x_{1_{k+1}}} &= w_{x_{1_k}} + T(-\tilde{\lambda}_5 L^{\frac{3}{8}} \lfloor w_1 \rceil^{\frac{5}{8}} + z_{x_0} - f_x(t)) \\
z_{x_{0_{k+1}}} &= z_{x_{0_k}} + T\tilde{\lambda}_4 L^{\frac{4}{8}} \lfloor w_1 \rceil^{\frac{4}{8}} + \sum_{j=1}^{4} \frac{T^j}{j!} z_{x_{j+1_k}} \\
z_{x_{1_{k+1}}} &= z_{x_{1_k}} + T\tilde{\lambda}_3 L^{\frac{5}{8}} \lfloor w_1 \rceil^{\frac{3}{8}} + \sum_{j=1}^{3} \frac{T^j}{j!} z_{x_{j+1_k}} \\
z_{x_{2_{k+1}}} &= z_{x_{2_k}} + T\tilde{\lambda}_2 L^{\frac{6}{8}} \lfloor w_1 \rceil^{\frac{2}{8}} + \sum_{j=1}^{2} \frac{T^j}{j!} z_{x_{j+2_k}} \\
z_{x_{3_{k+1}}} &= z_{x_{3_k}} + T\tilde{\lambda}_1 L^{\frac{7}{8}} \lfloor w_1 \rceil^{\frac{1}{8}} + \sum_{j=1}^{1} \frac{T^j}{j!} z_{x_{j+3_k}} \\
z_{x_{4_{k+1}}} &= z_{x_{4_k}} + T\tilde{\lambda}_0 L sign(w_1)
\end{aligned}
$$

Los parámetros seleccionados para los diferenciadores de las tres coordenadas fueron los siguientes:

Diferenciador RED filtrado	L	$\tilde{\lambda}_7$	$\tilde{\lambda}_6$	$\tilde{\lambda}_5$	$\tilde{\lambda}_4$	$\tilde{\lambda}_3$	$\tilde{\lambda}_2$	$\tilde{\lambda}_1$	$\tilde{\lambda}_0$
Coordenada x	Varias	12	84.12	281.37	455.40	295.74	88.78	14.13	1.1
Coordenada y	Varias	12	84.12	281.37	455.40	295.74	88.78	14.13	1.1
Coordenada z	Varias	12	84.12	281.37	455.40	295.74	88.78	14.13	1.1

Tabla 3.5.: Parámetros sintonizados del diferenciador robusto exacto de filtrado estándar (REDF).
Fuente: Elaboración propia.

Cabe resaltar que la constante de Lipschitz fue elegida empíricamente en base de simulaciones, efectuando un aumento progresivo de la constante en cada simulación a fin de evaluar sus efectos sobre la filtración del ruido y la robustez del diferenciador, con la finalidad de encontrar una relación óptima filtrado-robustez. Las constantes elegidas para comparar el desempeño fueron $L = \{1,5,10,20\}$.

3.10.3 Algoritmo del diferenciador robusto exacto de filtrado estándar (REDF)

A continuación, se presenta el algoritmo del programa del apéndice 2.13 correspondiente a la implementación del diferenciador robusto exacto estándar filtrado:

Algoritmo 3.11: Diferenciador Robusto Exacto estándar filtrado (RED_F)
Entradas: Vectores $\mathbf{y_k}$ y tiempo de muestreo ***T***
Salidas: VE estimados $\boldsymbol{xh_k}$, entrada de control estimada del modelo $\boldsymbol{uh_k}$ y derivada $\boldsymbol{duh_k}$ ***y*** colectores deslizantes $\boldsymbol{sig_k}$
Inicio del Programa: Invocado por el programa sim_RED_filt_estandar.m
1: Declara e inicializa VE estimadas en la coordenada "x" ($\boldsymbol{z0x, z1x, z2x, z3x\ y\ z4x}$), coordenada "y" ($\boldsymbol{z0y, z1y, z2y, z3y\ y\ z4y}$) y coordenada "z" ($\boldsymbol{z0z, z1z, z2z, z3z\ y\ z4z}$).
2: Declara e inicializa variables virtuales del filtro de la coordenada "x" ($\boldsymbol{w1x, w2x\ y\ w3x}$), coordenada "y" ($\boldsymbol{w1y, w2y\ y\ w3y}$) y coordenada "z" ($\boldsymbol{w1z, w2z\ y\ w3z}$).
3: Definir constante de Lipschitz (fijas) de las coordenadas "x", "y" y "z".
4: Definir el orden ***n=4*** del diferenciador de la coordenadas "x","y" y "z"

Algoritmo 3.11: Diferenciador Robusto Exacto estándar filtrado (RED_F) (continuación)

5: Definir las ganancias homogéneas del diferenciador de la coordenada "x", "y" y "z" **k0x, k1x, k2x, k3x, k4x, k5x, k6x y k7x, k0y, k1y, k2y, k3y, k4y, k5y, k6y y k7y** y **k0z, k1z, k2z, k3z, k4z, k5z, k6z y k7z,** respectivamente.
6: Establecer series de Taylor para discretización.
7: Establecer las variables deslizantes **sigx, sigy y sigz y sigdx, sigdy y sigdz (estás solo para gráfica de plano de estados)**
8: Filtrar variables deslizantes **sigx, sigy y sigz** en filtros de las coordenadas "x", "y" y "z".
9: Estimar VE filtradas futuras $\boldsymbol{xh_k}$, $\boldsymbol{uh_k}$ y $\boldsymbol{duh_k}$
15: Guardar variables para siguiente iteración.
16: Entrega $\boldsymbol{xh_k} = [\boldsymbol{z0x}; \boldsymbol{z1x}; \boldsymbol{z2x}; \boldsymbol{z0y}; \boldsymbol{z1y}; \boldsymbol{z2y}; \boldsymbol{z0z}; \boldsymbol{z1z}; \boldsymbol{z2z}]$
17: Entrega $\boldsymbol{uh_k} = [\boldsymbol{z3x}; \boldsymbol{z3y}; \boldsymbol{z3z}]$
18: Entrega $\boldsymbol{duh_k} = [\boldsymbol{z4x}; \boldsymbol{z4y}; \boldsymbol{z4z}]$
19: Entrega $\boldsymbol{sig_k} = [\mathbf{sigx}; \mathbf{sigy}; \mathbf{sigz}; \mathbf{sigdx}; \mathbf{sigdy}; \mathbf{sigdz}]$
20: Fin de programa

3.10.4 Simulación de la trayectoria del misil

Algoritmo 3.12: Diferenciador Robusto Exacto estándar filtrado (RED_F)

Entradas: Señal de entrada de control $\boldsymbol{\mu_k}$, perturbaciones $\boldsymbol{\xi_k}$ y modelo mixto gaussiano (GMM)
Salidas: VE reales del modelo $\boldsymbol{x_k}$, VE medidas del modelo $\boldsymbol{y_k}$, **VE filtradas** $\boldsymbol{xh_k}$, entrada de control al modelo (sobre aceleración) filtrada $\boldsymbol{uh_k}$ **y derivada** $\boldsymbol{duh_k}$, variables deslizantes $\boldsymbol{sig_k}$, errores cartesianos $\boldsymbol{error\_c_k}$, errores polares $\boldsymbol{error\_c_k}$, error RMSE y gráficos.
Inicio del programa:
1: Declara tiempo de muestreo $\boldsymbol{T}$, gravedad $\boldsymbol{g}$, constante de mach **M**, sobre aceleración máxima ***load_factor*** tiempo inicial de simulación $\boldsymbol{t_i}$, tiempo final de simulación t_{ff} y número total de muestras $\boldsymbol{nt}$.
2: Declara condiciones iniciales de traqueo al blanco: distancia $\boldsymbol{Ad}$, acimut verdadero $\boldsymbol{Bdn}$, elevación $\boldsymbol{Ed}$, velocidad $\boldsymbol{Vm}$ y rumbo verdadero $\boldsymbol{Rv}$.
3: Convierte posición de coordenadas polares a cartesianas usando $\boldsymbol{Ad}$, $\boldsymbol{Bdn}$ y $\boldsymbol{Ed}$
4: Declara el vector de estados inicial $\boldsymbol{x_k}(\mathbf{0})$
5: Declara matrices del espacio de estados $\boldsymbol{F_k}$, $\boldsymbol{G_k}$, $\boldsymbol{D_k}$ y $\boldsymbol{C_k}$.
6: Obtén $\boldsymbol{u_k}$ (invoca a función ***gen_jerk_sea_skimming3.m*** **(programa del apéndice 2.5))**
7: Obtén $\boldsymbol{\xi_k}$ **(invoca** a función ***perturb_1.m*** **(programa del apéndice 2.6))**
8: Cargar ruido angular ***glint_puntos.mat*** **(datos guardados del programa del apéndice 2.7)**
9: Ingrese "1" para simulación con ruido y "0" para simulación sin ruido.
10: Para tt = ti:T:tff **Hacer**
11: Ejecuta el modelo de transición de estados para obtener $\boldsymbol{x_k}$ futuro.
12: Invoca ***c2p.m*** para convertir $\boldsymbol{x_k}$ a coordenadas polares.
13: Invoca ***yk_noise.m*** para obtener el vector de mediciones y_k
14: Invoca **RED_filt_estandar.m (programa del apéndice 2.11)** para obtener $\boldsymbol{xh_k}$, $\boldsymbol{uh_k}$, $\boldsymbol{duh_k}$, $\boldsymbol{sig_k}$ filtrados.
15: Invoca ***c2p.m*** para convertir $\boldsymbol{xh_k}$ a coordenadas polares.
16: Extrae errores polares, errores cartesianos y RMSE
17: Almacena vectores para gráficas
18: Fin Para
19: Conversión de unidades del vector de estados a Mn, Mach y g's.
20: Graficar
21: Fin de programa

Se efectuaron simulaciones con diferentes valores de la constante de Lipschitz, observándose en la figura 3.56 que el diferenciador robusto exacto de filtrado estándar presenta muy buena filtración del ruido angular para constantes de Lipschitz $L = \{1{,}5{,}10\}$, sin embargo, carece de la exactitud y uniformidad necesaria requerida para compensar las perturbaciones del modelo lineal incierto del blanco aéreo de alta maniobrabilidad, siendo esto evidente durante la maniobra terminal (sub figura (d)). Por otro lado, cuando se utilizaron constantes de Lipschitz por encima de $L = 5$, se observó que la propiedad de filtración del ruido se perdía progresivamente y aumentaba la robustez. Asimismo, se observó un sesgo de la estimación a partir de $L = 15$. Por lo tanto, el REDF logra filtrar adecuadamente el ruido de cola larga o glint noise para constantes de Lipschitz bajas, pero carece de la uniformidad y exactitud durante trayectorias senoidales y cosenoidales del blanco siendo necesario proponer una solución para el problema en estudio que permite obtener una relación aceptable de filtrado, exactitud y uniformidad ante perturbaciones.

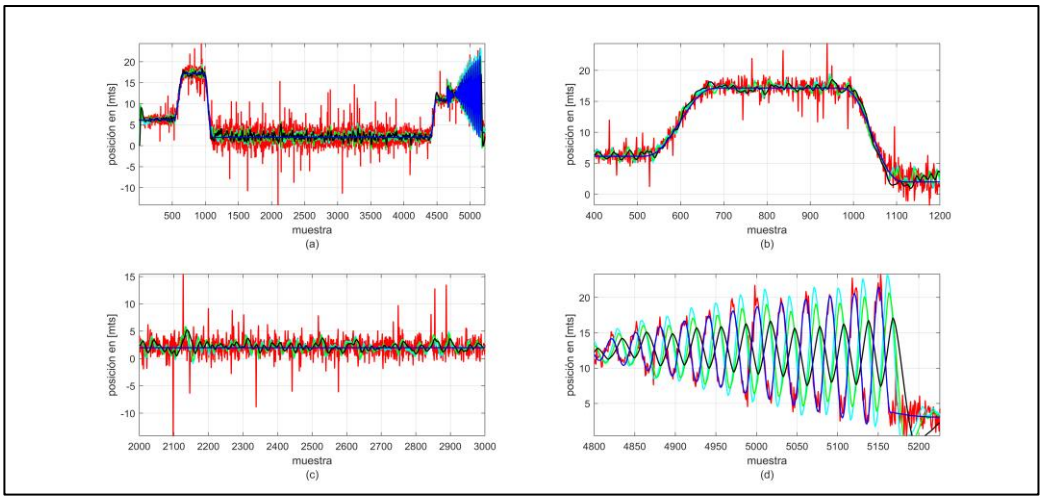

Figura 3.56.: Variable de estado de posición del misil en la coordenada "z" (REDF). Variable de estado real (línea azul). Variable de estado medida (línea roja) Variable de estado estimada constante L=1 (línea negra). Variable de estado estimada constante L=5 (línea verde). Variable de estado estimada constante L=10 (línea cyan). (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [2000 3000]) (d) Posición (k = [4800 5226]).
Fuente: Elaboración propia.

3.10.5 Diseño del diferenciador robusto exacto y uniforme filtrado (UREDF)

El problema que se tiene para la aplicación en estudio es filtrar la posición medida del blanco en un ambiente de alto ruido angular y mantener la mejor exactitud y uniformidad posible durante maniobras bruscas y maniobra terminal del misil. Cabe resaltar que al obtener una variable filtrada de posición esta puede ser utilizada por el diferenciador robusto exacto adaptativo (ARED) para obtener las derivadas restantes con mínimo ruido angular. Asimismo, de los resultados obtenidos en el apartado anterior se observó que el diferenciador robusto exacto filtrado ofrece resultados prometedores respecto al filtrado del ruido angular aproximado mediante una distribución mixta gaussiana (GMM) pero carece de la robustez necesaria durante la maniobra terminal del misil, dado que si se aumenta la constante de Lipschitz se amplifica el ruido de medición. Por ello, como solución al problema en estudio se propone el siguiente diferenciador correspondiente a la coordenada "x":

$$
\begin{aligned}
\dot{w}_{x_1} &= -\tilde{\lambda}_4 L^{\frac{1}{5}} \theta_x(t) \lfloor w_{x_1} \rceil^{\frac{4}{5}} + \theta_x(t) w_{x_2} \\
\dot{w}_{x_2} &= -\tilde{\lambda}_3 L^{\frac{2}{5}} \theta_x(t) \lfloor w_{x_1} \rceil^{\frac{3}{5}} + \theta_x(t) w_{x_3} \\
\dot{w}_{x_3} &= -\tilde{\lambda}_2 L^{\frac{3}{5}} \theta_x(t) \lfloor w_{x_1} \rceil^{\frac{2}{5}} + z_{x_0} - f_x(t) \\
\dot{z}_{x_0} &= -\tilde{\lambda}_1 L^{\frac{4}{5}} (\theta_x(t) \lfloor w_{x_1} \rceil^{\frac{1}{5}} + (1 - \theta_x(t)) \mu_{x_1} \lfloor \dot{w}_{x_3} \rceil^{\frac{5}{4}}) + z_{x_1} \\
\dot{z}_{x_1} &= -\tilde{\lambda}_0 L (\theta_x(t) sign(w_{x_1}) + (1 - \theta_x(t)) \mu_{x_2} \lfloor \dot{w}_{x_3} \rceil^{\frac{3}{2}})
\end{aligned}
$$

Donde, $f_x(t)$ es la versión continua de la posición medida del blanco en la coordenada "x", z_{x_0} es la estimación de la posición en la coordenada x, z_{x_1} es la estimación de la velocidad en la coordenada x, L es la constante de Lipschitz, los parámetros $\tilde{\lambda}_0$, $\tilde{\lambda}_1$, $\tilde{\lambda}_2$, $\tilde{\lambda}_3$ y $\tilde{\lambda}_4$ son las ganancias homogéneas del diferenciador de acuerdo a la tabla 3.3, w_{x_1}, w_{x_2} y w_{x_3} son variables virtuales de filtrado del error de posición, μ_{x_1} y μ_{x_2} son los pesos de diferenciación uniforme y $\theta_x(t)$ es una función de conmutación entre los términos de filtrado estándar y los términos de orden alto del diferenciador, los cuales son dependientes de la variable deslizante filtrada $\dot{w}_{x_3}$. Cabe resaltar que para obtener una estimación suave de la posición fue necesario obtener la velocidad, dado que al elevar el orden del diferenciador la discontinuidad del término $sign(w_1)$ es asumida por z_{x_1}.

El diferenciador propuesto efectúa el filtrado de la función $f_x(t)$ por medio del diferenciador robusto exacto de filtrado estándar para $\lfloor \dot{w}_{x_3} \rceil \leq 1$, siendo el valor de $\theta_x(t) = 1$. Por ello, para

todo valor $|\dot{w}_{x_3}| \leq 1$ el efecto de los términos de orden alto del diferenciador uniforme es cancelados mediante el término $1 - \theta_x(t)$. Si el valor de la variable deslizante filtrada $\dot{w}_{x_3}$ cumple $|\dot{w}_{x_3}| \geq 1$, que normalmente esta condición se da en maniobras de cambio de rumbo o maniobra terminal del misil, la función $\theta_x(t)$ toma el valor de $\theta_x(t) = 0$ y anula el filtrado estándar y activa a los términos de diferenciación uniforme, a fin de que en un tiempo finito lleven a la variable deslizante filtrada $\dot{w}_{x_3}$ a un conjunto compacto. Dado que para $\theta_x(t) = 0$ se cumple $\dot{w}_{x_3} = z_{x_0} - f_x(t)$ y con la finalidad de no introducir ruido angular a las estimaciones por medio de la variable deslizante ruidosa $f_x(t)$, se propone filtrar en tiempo real a esta variable por medio de un filtro no lineal de mediana o *median filter*; el filtrado no lineal de mediana ha sido utilizado ampliamente en procesamiento de imágenes de radares de apertura sintética (SAR), dadas sus características de filtrado de ruidos con características no gaussianas. Este filtro no lineal procesa las mediciones efectuadas por el radar en cepas o *batches* y aplica a cada *batch* la operación de mediana, basándose en las siguientes definiciones:

Definición 3.23 (Invarianza de escala y de translación [115]): *Para cualquier secuencia aleatoria de entrada* $x^n = \{x_1, x_2, \ldots, x_n\}$ *donde* $n = 2m - 1$, *se tiene que* $mediana(c \cdot x^n + d\{1\}) = c \cdot mediana(x^n) + d = c \cdot x_m + d$.

Definición 3.24 (Eficiencia de filtrado [116]): *Sea* $x^n = \{x_i\}$ *y* $x_i = c + v_i$, *donde c es una señal deseada constante y* v_i *es ruido blanco de media cero. Si* v_i *tiene una distribución Laplaciana, entonces* $mediana(x^n)$ *es la estimación óptima de c en el sentido de la máxima similitud o Maximum Likelihood sense.*

Chang y Wu [115] proponen un filtro de mediana con retroalimentación de velocidad predicha utilizando como algoritmo de traqueo al filtro Kalman, obteniendo buenos resultados de filtración de ruido angular por medio de una técnica pre procesamiento de pulsos de radar. Según lo detallado por Chang y Wu [115], cuando la frecuencia de repetición de pulsos (Pulse Repetition Frequency o PRF por sus siglas en inglés) es mayor que el período de muestreo del algoritmo de filtrado, un sistema de traqueo tendrá un mayor número de datos medidos de posición de los que puede procesar. Por ejemplo, se conoce que el período de muestreo del algoritmo de estimación del sistema de control de tiro WM-28 o en su versión americana MK92 FCS es de 0.02 segundos, pero su frecuencia de repetición de pulsos (PRF) es seleccionable para un valor de 1800 o 3600 pulsos por segundo. Esto quiere decir que, para 1800 pulsos por segundo, el radar de control de tiro en un período de 0.02 segundos obtendrá 36 retornos o *paints* de la posición del blanco. En la figura 3.57 se muestran el pulso y_{k-1} y y_k, los cuales son procesados por el observador de estados a cada instante de muestreo T=0.02 segundos. Sin embargo, entre estos pulsos se observa una cantidad de pulsos determinada por el PRF del radar que no son procesados por un radar de control de tiro sin pre procesamiento en secuencias o batches. Por ello, para el ordenamiento de estas muestras por batches se denomina a z_k como la muestra central del batch ${z_k}^n = \{z_{k-l}, \ldots, z_k, \ldots z_{k+l}\}$, cumpliéndose $n = 2l - 1$, donde n es la longitud total del batch y l es la cantidad de muestras antes y después de la muestra central z_k. Inmediatamente antes que inicie el batch ${z_k}^n$ se obtuvo por medición de radar la muestra y_{k-1} e inmediatamente después la muestra actual y_k. Posteriormente, se obtendrá el batch ${z_{k+n}}^n = \{z_{k+n+-l}, \ldots, z_{k+n}, \ldots z_{k+n+l}\}$. Por tanto, se forman secuencias o batches para posteriormente aplicar el filtro de mediana $y_k = mediana({z_k}^n)$ en el instante actual de muestreo $k = 1$, permitiendo brindar una posición filtrada al estimador u observador de estados sin excesiva varianza.

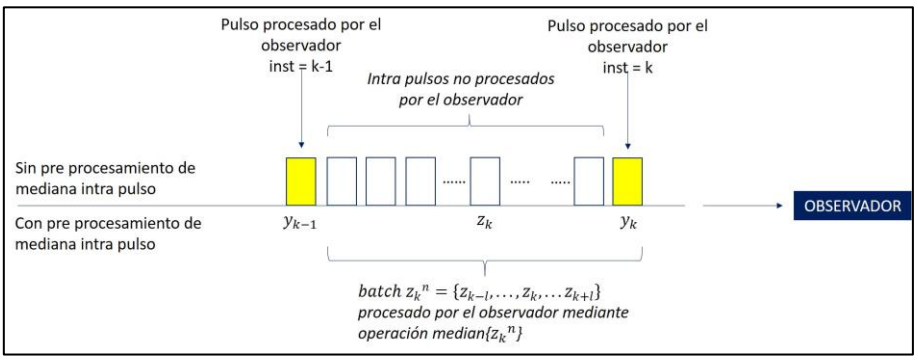

Figura 3.57.: Pre procesamiento intra pulso de pulsos de radar.
Fuente: Elaboración propia.

El filtro no lineal de mediana puede ser utilizado mediante la técnica intra pulso y en su defecto, si no se cuenta con capacidad intra pulso, considerando a la muestra central z_k de la secuencia $z_k{}^n = \{z_{k-l}, \dots, z_k, \dots z_{k+l}\}$ igual a y_k. Al colocar a y_k en la posición central de la secuencia sería necesario esperar a obtener por radar l muestras posteriores para poder filtrar la muestra y_k; esto genera un retardo de tiempo de $l * T$ segundos debido a que el observador de estados debe esperar ese tiempo para procesar el batch.

Para el diseño del diferenciador robusto exacto y uniforme filtrado se consideraron ambas opciones; para el caso de filtrado de mediana intra pulso solo se consideró cuando $\theta_x(t) = 0$ cumple. Por otro lado, para el filtrado de mediana sin intra pulso se consideró una longitud de ventana $n = 3$ para $\theta_x(t) = 0$ y $n = 5$ para $\theta_x(t) = 1$. A continuación, se presentan los parámetros escogidos para el diferenciador en estudio.

Diferenciador Filt. Unif.	**L**	$\boldsymbol{\mu_1}$	$\boldsymbol{\mu_2}$	$\boldsymbol{n}$		$\tilde{\boldsymbol{\lambda}}_4$	$\tilde{\boldsymbol{\lambda}}_3$	$\tilde{\boldsymbol{\lambda}}_2$	$\tilde{\boldsymbol{\lambda}}_1$	$\tilde{\boldsymbol{\lambda}}_0$
				$(\boldsymbol{\theta(t) = 1})$	$(\boldsymbol{\theta(t) = 0})$					
Coordenada x	1	4.56	1.86	5	3	5.5	10.03	9.30	4.57	1.1
Coordenada y	1	3.86	1.57	5	3	5.5	10.03	9.30	4.57	1.1
Coordenada z	1	4.56	1.86	5	3	5.5	10.03	9.30	4.57	1.1

Tabla 3.6.: Parámetros sintonizados del diferenciador robusto exacto y uniforme filtrado (UREDF)
Fuente: Elaboración propia.

3.10.6 Algoritmo del diferenciador robusto exacto y uniforme filtrado

A continuación, se presenta el algoritmo del programa del apéndice 2.14 correspondiente al diferenciador robusto exacto estándar y uniforme filtrado:

Algoritmo 3.13: Diferenciador Robusto exacto y uniforme filtrado

Entradas: Vectores $\mathbf{y_k}, \mathbf{y_{k\_log}}, \mathbf{batch_{ykx}}, \mathbf{batch_{yky}}, \mathbf{batch_{ykz}}$ *e intra, muestra actual* k y tiempo de muestreo ***T***
Salidas: Vector de variables de estado estimadas de posición $\boldsymbol{ykfu\_filt}$
Inicio del Programa: Invocado por el programa sim_RED_filtrado.m
1: Declarar e inicializar VE estimadas de posición y velocidad en coordenada "x" ($\boldsymbol{z0x, z1x}$), coordenada "y" ($\boldsymbol{z0y, z1y}$) **Y** coordenada "z" ($\boldsymbol{z0z, z1z}$).
2: Declarar e inicializar variables virtuales de filtrado en coordenada "x" ($\boldsymbol{w1x, w2x\ y\ w3x}$), coordenada "y" ($\boldsymbol{w1y, w2y\ y\ w3y}$) ***y*** coordenada "z" ($\boldsymbol{w1z, w2z\ y\ w3z}$).
3: Declarar ventanas de batch en coordenadas "x", "y" y "z" ($\boldsymbol{mx, my\ y\ mz}$).
4: Declarar parámetros del diferenciador de la coordenada "x", coordenada "y" y coordenada "z" (según tabla 3.4).
5: Switch intra
6: **caso** 0 (sin intra pulso)
7: **Si** k **mayor** 4*mx
8: **Si** mod(k,2) **igual a** 0
9: n = k – (2*mx +1)
10: **else** n = k - 2*mx
11: **Fin para**
12: Aplica filtrado de mediana ykx_filt = median(Sx), donde Sx = yk_log(1,n-mx:n+mx), tomándo ***mx*** muestras hacia atrás y ***mx*** muestras hacia adelante respecto de la muestra central ***n***
13: **else** ykx_filt = yk(1,1)
14: **Fin caso**
15: **caso** 1 (con intra pulso)
16: Aplica filtrado de mediana Sx = batch_ykx(k,:) a la secuencia intra pulso k
17: **end**
18: Si $|\boldsymbol{w3x\_dot}|$ **mayor a** 1
19: $\boldsymbol{\theta_x(t) = 0}$ (diferenciador uniforme)
20: Establece **mx = 1** (solo para caso 0 de switch)
21: Computa variable deslizante **sigx = -(ykx_filt – z0x)**
22: else
23: $\boldsymbol{\theta_x(t) = 1}$ (diferenciador filtrado)
24: Establece **mx = 2** (solo para caso 1 de switch)
25: Computa variable deslizante **sigx = -(yk(1,1) – z0x)** (variable sin filtrar)
26: Fin Si
27: Efectuar la transición de estados del diferenciador de la coordenada "x"
28: Integrar estados del diferenciador de la coordenada "x"
29: Efectuar mismo procedimiento de las líneas (12 a 34) para los diferenciadores de las coordenadas "y" y "z".
30: Guardar variables para siguiente iteración.
31: Entrega $\boldsymbol{ykfu\_filt} = [\boldsymbol{z0x; z0y; z0z}]$;
32: Fin de programa

3.10.7 Simulación de la trayectoria del misil

Para la simulación de la trayectoria con ruido fue necesario implementar el programa del apéndice 2.15, el cual genera las muestras intra pulso necesarias para demostrar la efectividad del algoritmo propuesto para el filtrado.

Algoritmo 3.14: Generación de muestras intra pulso para filtrado por mediana

Entradas: Vectores $\mathbf{x_{k\_log}, y_{k\_log}, polar_{log}}$, $\boldsymbol{Xgm12t, Xgm34t}$ y ***nro_intra_pulsos***
Salidas: Vector de muestras intra pulso $\boldsymbol{batch_{ykx}, batch_{yky}}$ $\boldsymbol{y}$ $\boldsymbol{batch_{ykz}}$
Inicio del Programa: Invocado por el programa ***sim_RED_filtrado.m***
1: Declarar número de puntos de simulación $\boldsymbol{nt}$**,** longitudes de subsecuencias del modelo mixto Gaussiano $\boldsymbol{nt\_set}$ *y* $\boldsymbol{nt\_set2}$**,** y constantes $\boldsymbol{a}$ y $\boldsymbol{j}$**.**
2: Para k **igual a** 2 **hasta** nt
3: Declara medición anterior en alcance last_paint_Ad = polar_log(1,k-1), medición actual en alcance actual_paint_Ad = polar_log(1,k), "nro_intra_pulsos" puntos entre last_paint_Ad y actual_paint_Ad.
4: Efectúa el procedimiento de la línea 3 para el acimut Bdn y elevación Ed.
5: **Si** altura del misil **mayor o igual a** 5
6: **Si** k **menor o igual a** nt_set*j
7: Declara ruido **Ad_noise, Bdn_noise** y **Ed_noise** con los primeros **"nt_set"** puntos de la distribución **Xgm12t.**
8: **Si** k **igual a** nt_set*j
9: **Si** k **igual a** 5200
10: Suma 26 al índice **"a"**
11: Haz nt_set = nt_set2 (para no pasar de 5226 muestras)
12: **de otro modo**
13: Suma 52 al índice **"a"**
14: **Fin Si**
15: **Fin Si**
16: Suma 1 al índice "j" (Para poder abarcar la siguiente secuencia de ruido)
17: **Fin Si**
18: Adh = (batch_Ad + Ad_noise).*cosd(batch_Ed + Ed_noise)
19: batch_ykx(k,:) = Adh.*cosd(batch_Bdn + Bdn_noise)
20: batch_yky(k,:) = Adh.*sind(batch_Bdn + Bdn_noise)
21: batch_ykz(k,:) = (batch_Ad + 0.05*Ad_noise).* sind(batch_Ed + 0.05*Ed_noise)
22: **de otro modo**
23: Efectuar mismo procedimiento de las líneas 8 a 23 pero cambiando Xgm12t por Xgm34t
24: **Fin Si**
25: Fin Para
26: Hacer elementos de primera fila de los batches igual a la primera medición (dado que para la primera medición no se cuenta con mediciones anteriores
27: Retornar batch_ykx batch_yky batch_ykz
28: Fin Programa

Por otro lado, el desempeño de filtrado del diferenciador robusto exacto y uniforme filtrado (UREDF) se contrastó con el desempeño de los filtros *Kalman Filter* (KF), *Cubature Kalman Filter* (CKF) y *Particle Filter (PF), implementados* mediante las funciones disponibles en el toolbox de *Sensor Fusion and Tracking* de MATLAB®. Cabe resaltar que el criterio de elección fue contrastar la robustez y capacidad de filtrado del diferenciador diseñado con un filtro con modelo lineal y filtrado gaussiano (KF), un filtro con modelo no lineal y filtrado gaussiano (CKF) y otro con modelo no lineal y filtrado no gaussiano (PF). A continuación, se muestra el algoritmo del programa del apéndice 2.16 que implementa la simulación.

Algoritmo 3.15: Simulación de trayectoria del blanco con diferenciador filtrado (UREDF)

Entradas: Señal de entrada de control $\boldsymbol{\mu_k}$**,** perturbaciones $\boldsymbol{\xi_k}$ y modelo mixto gaussiano de ruido
Salidas: VE reales del modelo $\boldsymbol{x_k}$*,* VE medidas del modelo $\boldsymbol{y_k}$*,* VE estimadas de posición "x","y" y "z" $\boldsymbol{xhURED\_F}$ de UREDF, VE estimadas $\boldsymbol{xhKF}$ de KF, VE estimadas $\boldsymbol{xhUKF}$ de UKF VE estimadas $\boldsymbol{xhCKF}$ de CKF, errores cartesianos, errores polares, error RMSE, tiempo de cómputo de los algoritmos en cada iteración TCPU y gráficos.
Inicio del programa:
1: Declara tiempo de muestreo $\boldsymbol{T}$*,* gravedad $\boldsymbol{g}$**,** constante de mach **M**, sobre aceleración máxima ***load_factor*** tiempo inicial de simulación $\boldsymbol{t_i}$**,** tiempo final de simulación t_{ff} y número total de muestras $\boldsymbol{nt}$**.**
2: Declara condiciones iniciales de traqueo al blanco: distancia $\boldsymbol{Ad}$**,** acimut verdadero $\boldsymbol{Bdn}$*,* elevación $\boldsymbol{Ed}$**,** velocidad $\boldsymbol{Vm}$ y rumbo verdadero $\boldsymbol{Rv}$**.**
3: Convierte posición de coordenadas polares a cartesianas usando $\boldsymbol{Ad, Bdn}$ $\boldsymbol{y}$ $\boldsymbol{Ed}$
4: Declara el vector de estados inicial $\boldsymbol{x_k(0)}$
5: Declara matrices del espacio de estados $\boldsymbol{F_k}$, $\boldsymbol{G_k}$, $\boldsymbol{D_k}$ y $\boldsymbol{C_k}$.

Algoritmo 3.15: Simulación de trayectoria del blanco con diferenciador filtrado (UREDF) (continuación)

6: Obtén $\boldsymbol{u_k}$ (invoca a función ***gen_jerk_sea_skimming3.m*** **(programa del apéndice 2.5))**
7: Obtén $\boldsymbol{\xi_k}$ **(invoca** a función ***perturb_1.m*** **(programa del apéndice 2.6))**
8: Cargar ruido angular ***glint_puntos.mat*** **(datos guardados del programa del apéndice 2.7)**
9: Generar datos intra pulso **(programa del apéndice 2.16*) batch_ykx batch_yky batch_ykz***
10: Inicializar filtro KF = trackingKF('MotionModel','3D Constant Acceleration','State',0.5*$\boldsymbol{x_k}(\mathbf{0})$)
11: Inicializar filtro CKF = trackingCKF(@constacc,@cameas, 0.5*$\boldsymbol{x_k}(\mathbf{0})$);
12: Inicializar filtro PF. stateCov = diag([0.5 0.001 0.001 0.0001 0.001 0.001 0.7 0.001 0.001]),
myPF = particleFilter(@contaccParticleFilterStateFcn,@contaccMeasurementLikelihoodFcn),
initialize(myPF,2000,xk,stateCov), myPF.StateEstimationMethod = 'mean',
myPF.ResamplingMethod = 'residual'.
13: prompt('Simular con filtrado intrapulso [1], sin filtrado intrapulso [0]')
14: intra = input('prompt')
15: Para tt = ti:T:tff **Hacer**
11: Obtén $\boldsymbol{x_k}$ futuro.
12: Invoca función ***c2p.mat*** para convertir $\boldsymbol{x_k}$ en coordenadas polares.
13: Invoca función ***yk_noise.m*** para obtener y_k
14: Tic. Ejecuta **RED_unif_filt.m (programa del apéndice 2.15)** para obtener $\mathbf{xhURED\_F}$.
TCPU_xh_UREDF = toc;
15: Tic. Ejecuta [xhKF,phKF] = predict(KF,T) para predecir estimados futuros, xhKF=xhKF' y
[xcKF,pcKF] = correct(KF,[yk(1,1),yk(3,1),yk(5,1)']) para corregir con la medición
disponible. TCPU_xh_KF = toc.
16: Tic. Ejecuta [xhCKF,phCKF] = predict(CKF,T) para predecir estimados futuros y
[xcCKF,pcCKF] = correct(CKF,[yk(1,1),yk(3,1),yk(5,1)']) para corregir con la medición
disponible. TCPU_xh_CKF = toc;
17: Tic. Ejecuta [xhPF,phPF] = predict(PF,T) para predecir estimados futuros y
[xcPF,pcPF] = correct(PF,[yk(1,1),yk(3,1),yk(5,1)']) para corregir con la medición
disponible. TCPU_xh_PF = toc;
18: Extrae errores polares, extrae error RMSE, extrae errores cartesianos, extrae tiempo de
cómputo de los algoritmos en cada iteración TCPU y almacena vectores para gráficas.
19: Fin Para
20: Conversión de unidades del vector de estados a Mn, Mach y g's.
21: Graficar
22: Fin de programa

En la primera simulación se comparó el desempeño del UREDF sin mediana intrapulso con el REDF, a fin de comprobar la mejora en exactitud y robustez, obteniéndose los resultados de la figura 3.58. En esta figura se puede observar que el diferenciador diseñado muestra similar capacidad de filtrado que el diferenciador de filtrado estándar, pero con una mejor robustez durante los cambios de rumbo y maniobra terminal. Sin embargo, existe un retardo de tiempo visible en la maniobra terminal de 5 muestras, siendo esto predecible porque el filtro de mediana debe esperar 5 muestras hacia adelante para poder procesar la muestra actual.

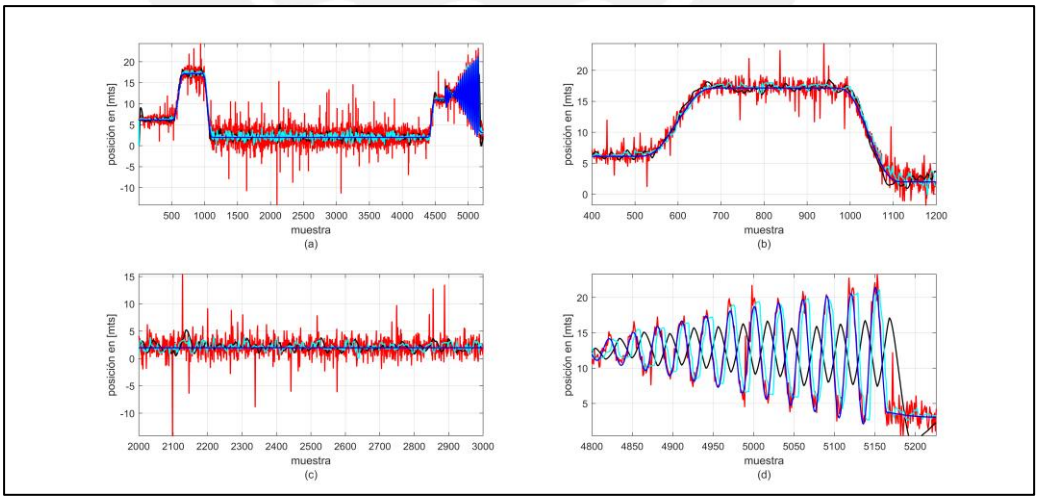

Figura 3.58.: Variable de estado de posición del misil en la coordenada "z" (UREDF vs REDF). Variable de estado real (línea azul). Variable de estado medida (línea roja) Variable de estado estimada REDF para una constante de Lipschitz L=1 (línea negra). Variable de estado estimada UREDF (línea cyan). (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [2000 3000]) (d) Posición (k = [4800 5226]).
Fuente: Elaboración propia.

Posteriormente, se efectuó la simulación según el programa del apéndice 2.17 para contrastar el desempeño del diferenciador robusto exacto y uniforme filtrado (UREDF) y los filtros Kalman (KF), Cubature Kalman Filter (UKF) y Particle Filter (PF). El diferenciador robusto exacto y

uniforme filtrado fue configurado de acuerdo a los parámetros de la tabla 3.5, activándose el filtrado de mediana intra pulso durante las convergencias uniformes. En la figura 3.59 se observa que los observadores que mejor capacidad de filtrado poseen son el filtro de partículas (PF) y el diferenciador robusto exacto y uniforme filtrado (UREDF), siendo esto verificado en la figura 3.60 durante la maniobra terminal; es evidente que los filtros Kalman no poseen capacidad de filtrar el ruido angular salvo se implementen funciones adicionales. Esto se constata en la figura 3.60, en la cual se observa que durante la maniobra terminal el CKF posee buena robustez, pero sin filtrado eficiente del ruido, el PF posee buena robustez y filtra adecuadamente el ruido y el filtro Kalman (KF) no posee buena robustez y filtrado de ruido. Cabe resaltar que las estimaciones de velocidad efectuadas por el diferenciador robusto exacto y uniforme filtrado son inexactas dado que, como se ha explicado anteriormente, la constante de Lipschitz no es adaptada en tiempo real. Sin embargo, dado que el objetivo es lograr el filtrado exacto y uniforme de la posición en este caso en particular no es necesario obtener variables de estado de velocidad exactas.

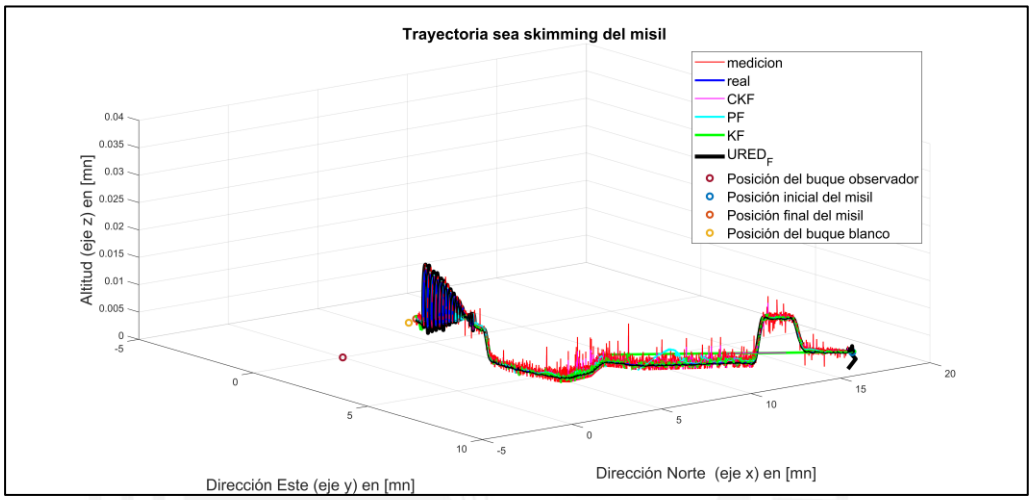

Figura 3.59.: Trayectoria completa en tres dimensiones del misil (UREDF vs otros).
Fuente: Elaboración propia.

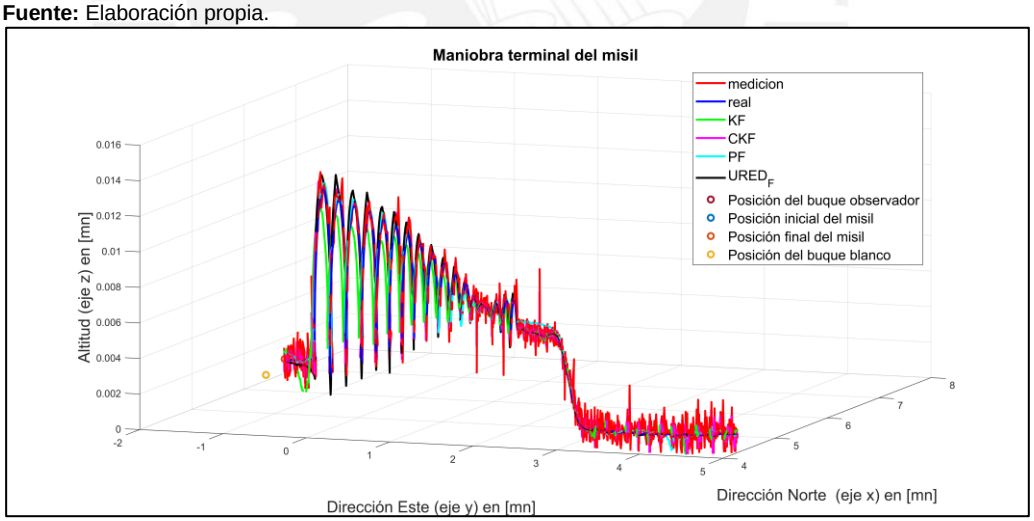

Figura 3.60.: Maniobra terminal: Filtrado de posición en la coordenada “z” (UREDF vs otros).
Fuente: Elaboración propia.

En la figuras 3.61,3.62 y 3.63 se aprecia que todos los observadores, a excepción del filtro Kalman (KF), realizan un buen traqueo de la posición del blanco en la coordenada “x”,”y”, y “z”, sin embargo, los observadores que poseen mejores estimaciones con mínimo ruido son el filtro de partículas (PF) y el diferenciador robusto exacto y uniforme filtrado (UREDF), siendo este último más rápido en la convergencia que el filtro de partículas. En la sub figura inferior derecha se observa la capacidad de filtrado del filtro de partículas (PF) y diferenciador robusto exacto y uniforme filtrado (UREDF).

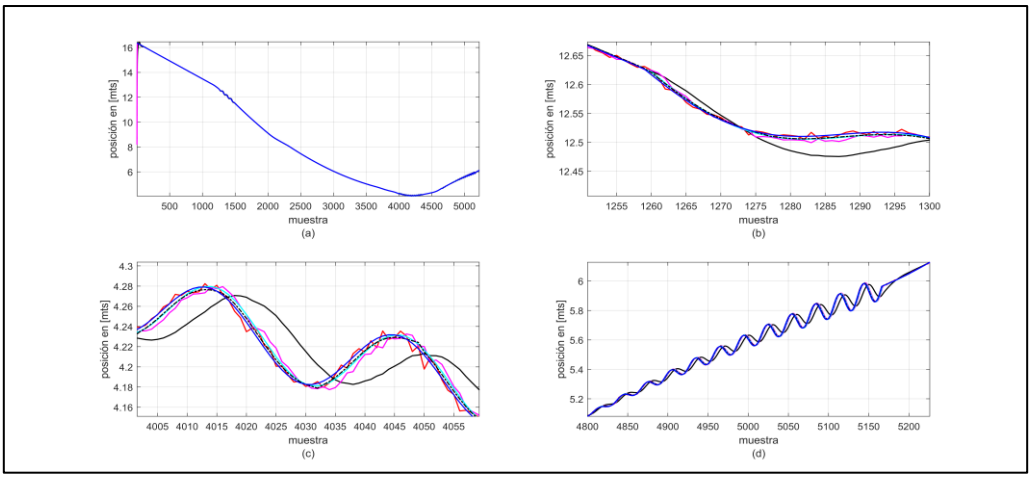

Figura 3.61.: Variable de estado de posición del misil en la coordenada “x” (UREDF vs otros). Variable de estado real (línea azul). Variable de estado medida (línea roja) Variable de estado estimada KF (línea negra). Variable de estado estimada CKF (línea magenta). Variable de estado estimada PF (línea cyan). Variable de estado estimada UREDF (línea negra punto-raya) (a) Posición (k = [0 5226]). (b) Posición (k = [1250 1300]) (c) Posición (k = [4000 4060]) (d) Posición (k = [4800 5226]).
Fuente: Elaboración propia.

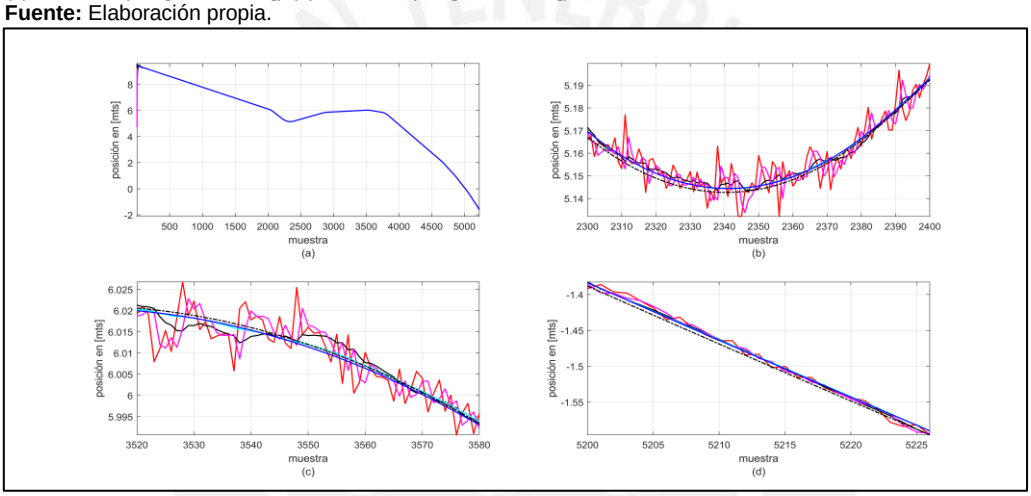

Figura 3.62.: Variable de estado de posición del misil en la coordenada “y” (UREDF vs otros). Variable de estado real (línea azul). Variable de estado medida (línea roja) Variable de estado estimada KF (línea negra). Variable de estado estimada CKF (línea magenta). Variable de estado estimada PF (línea cyan). Variable de estado estimada UREDF (línea negra punto-raya) (a) Posición (k = [0 5226]). (b) Posición (k = [2300 2400]) (c) Posición (k = [3520 3580]) (d) Posición (k = [5200 5226]).
Fuente: Elaboración propia.

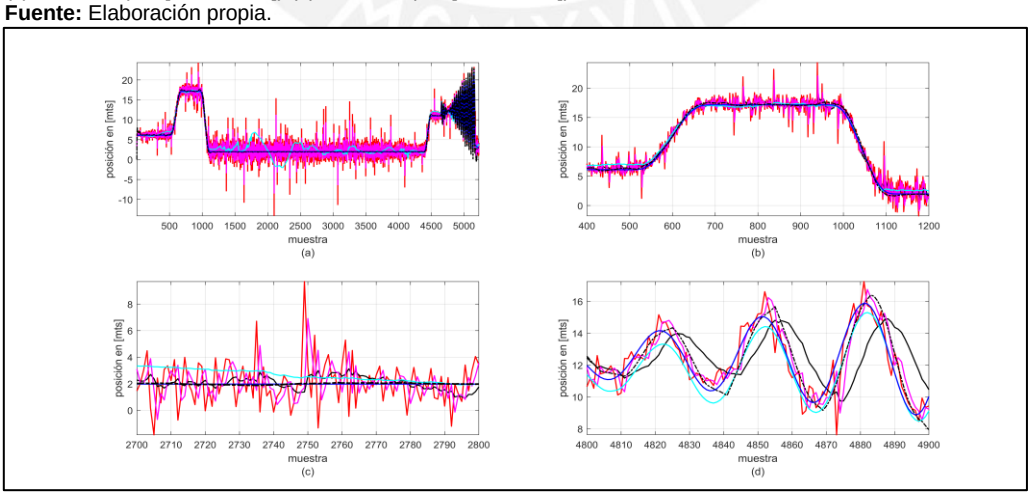

Figura 3.63.: Variable de estado de posición del misil en la coordenada “z” (UREDF vs otros). Variable de estado real (línea azul). Variable de estado medida (línea roja) Variable de estado estimada KF (línea negra). Variable de estado estimada CKF (línea magenta). Variable de estado estimada PF (línea cyan). Variable de estado estimada UREDF (línea negra punto-raya) (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [2700 2800]) (d) Posición (k = [4800 5226]).
Fuente: Elaboración propia.

En las figuras 3.64 y 3.65 se observa que los menores errores en coordenadas cartesianas y polares durante toda la trayectoria y maniobra terminal lo obtienen el filtro de partículas (PF) y diferenciador UREDF, mostrando una buena relación filtrado-robustez.

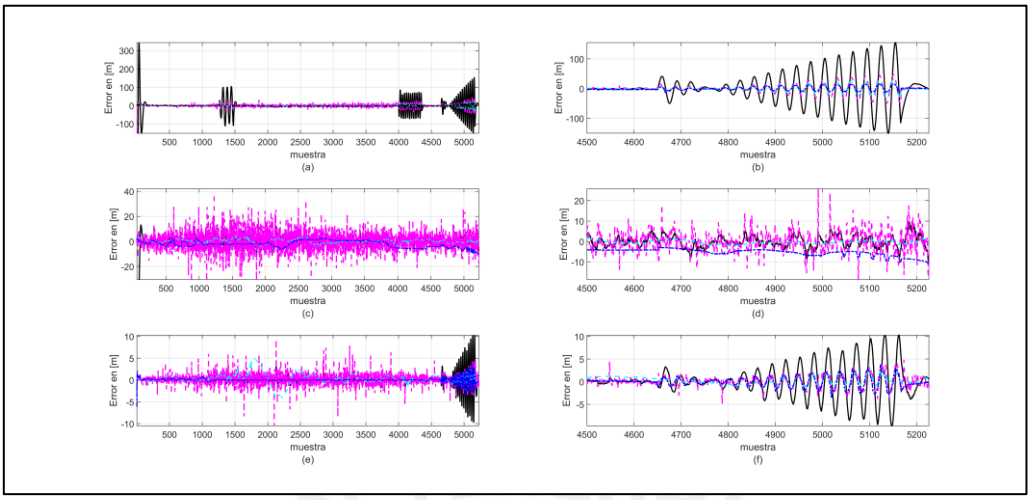

Figura 3.64.: Errores cartesianos (UREDF vs otros). KF (línea negra). CKF (línea magenta). PF (línea cyan). UREDF (línea negra punto-raya). (a) Error de posición coordenada "x" (k = [0 5226]). (b) Error de posición coordenada "x" (k = [4800 5226]).(c) Error de posición coordenada "y" (k = [0 5226]). (d) Error de posición coordenada "y" (k = [4800 5226]).(e) Error de posición coordenada "z" (k = [0 5226]). (f) Error de posición coordenada "z" (k = [4800 5226]).
Fuente: Elaboración propia.

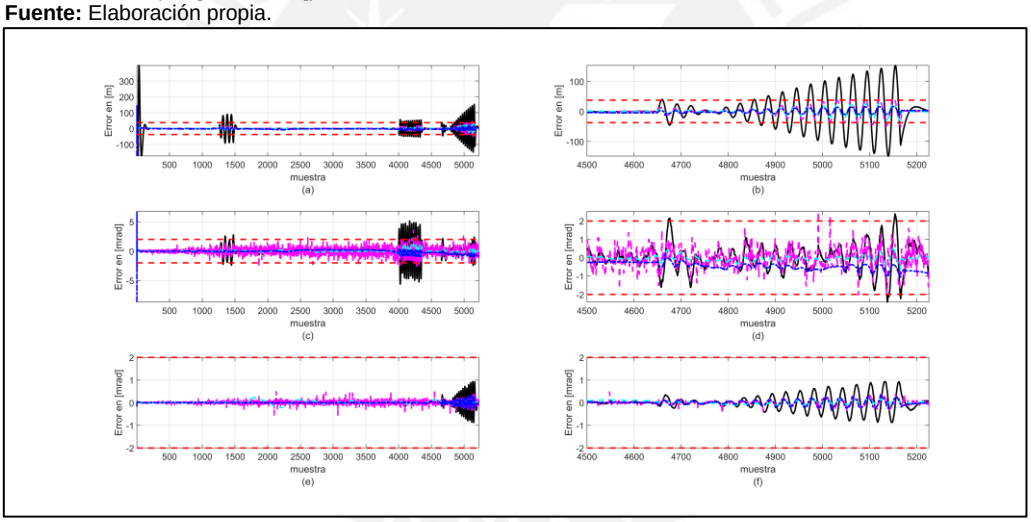

Figura 3.65.: Errores polares (UREDF vs otros). KF (línea negra). CKF (línea magenta). PF (línea cyan). UREDF (línea azul punto-raya) (a) Error en alcance Ad (k = [0 5226]). (b) Error en alcance Ad (k = [4500 5226]). (c) Error en azimuth verdadero Bdn (k = [0 5226]). (d) Error en azimuth verdadero Bdn (k = [4500 5226]). (e) Error en elevación Ed (k = [0 5226]). (f) Error en elevación Ed (k = [4500 5226]).
Fuente: Elaboración propia.

Finalmente, en la tabla 3.7 se muestran los tiempos de cómputo promedio y máximo de todos los algoritmos, observándose que el UREDF y KF obtuvieron los tiempos de cómputo promedio más bajos. Por otro lado, se observó que en lo que respecta al tiempo de cómputo máximo los filtros KF, CKF y PF sobrepasaron el tiempo de muestreo programado para el algoritmo, siendo esto un comportamiento indeseable.

Filtros	KF	CKF	PF	UREDF
Tiempo de cómputo promedio (seg)	5.4508e-04	0.0031	0.0045	9.5943e-05
Tiempo de cómputo máximo (seg)	0.0209	0.0861	0.0467	0.0175

Tabla 3.7.: Comparación de tiempos de cómputo de los algoritmos.
Fuente: Elaboración propia.

A partir de los resultados obtenidos en la presente tesis, Aranda y Pérez-Zuñiga [117] efectúan la comparación de desempeño del UREDF con filtros Kalman basados en la teoría bayesiana variacional por medio de simulaciones Montecarlo. Al respecto, se efectúa la comparación del UREDF con el Robust Student's t-based Kalman filter (RSTKF) propuesto por Huang et al. [118], siendo este último diseñado especialmente para el filtrado de ruido no guassiano modelado con distribución de tipo Student-t, encontrando que el algoritmo UREDF posee una robustez superior que el filtro RSTKF y filtrado de ruido no gaussiano similar.

3.11 Comparación de los observadores de modos deslizantes propuestos

En este apartado se compara el desempeño de los observadores de modos deslizantes desarrollados en la presente tesis, brindando una puntación del 0 al 5 a los siguientes criterios de desempeño:

- Nivel de robustez ante perturbaciones N_ρ
- Nivel de castañeo o "chattering" N_ε
- Nivel de sensibilidad al ruido de medición N_{ruido}
- Calidad de reconstrucción de perturbaciones ξ_{est}
- Tiempo de establecimiento en el colector deslizante T_{es}
- Número de variables de estado medibles necesarias para el cálculo del algoritmo N_{var}
- Número de variables de estado estimables N_{est}

Para el caso del criterio de desempeño "Error medio cuadrático polar promedio $RMSEp_{avg}$", se consideró una puntuación de 5 para $RMSEp_{avg} < 0.1$, puntuación de 4 para $0.1 \leq RMSEp_{avg} < 0.5$, puntuación de 3 para $0.5 \leq RMSEp_{avg} < 1.0$, puntuación de 2 para $1.0 \leq RMSEp_{avg} < 1.5$, puntuación de 1 para $1.5 \leq RMSEp_{avg} < 2.0$ y puntuación de 0 para $2.0 \leq RMSEp_{avg}$. Al respecto, se utilizó la siguiente tabla resumen de errores, donde se presentan tanto los errores cartesianos y polares de todos los observadores diseñados:

RMSE	ESSMO	WZSMO	HOSMO	RED	ARED	FRED	UREDF
Posición "x" (m)	0.2360	0.2135	0.3448	1.9095	0.2065	3.1692	0.2808
Velocidad "x" (m/s)	0.1623	0.1361	0.5835	3.3221	1.0594	3.6667	4.0622
Aceleración "x" (m/s2)	0.2552	0.0702	1.1155	5.4280	0.5930	5.3930	NA
Posición "y" (m)	0.5021	0.5034	0.0051	0.5032	0.5722	1.0099	0.1045
Velocidad "y" (m/s)	0.0446	0.0371	0.1747	1.4767	0.8104	3.3409	5.8036
Aceleración "y" (m/s2)	0.1282	0.7335	0.6688	3.3623	1.7896	4.1192	NA
Posición "z" (m)	0.0088	0.0079	0.0156	0.1073	0.0254	0.2320	0.0469
Velocidad "z" (m/s)	0.0111	0.0093	0.1044	0.5664	0.1290	0.2824	0.1428
Aceleración "z" (m/s2)	0.0186	0.1491	0.1506	1.3439	0.2716	0.4071	NA
$RMSEc_{avg}$	0.1519	0.2067	0.3514	2.0022	0.6063	2.4023	1.7401
Ad (m)	0.2240	0.2001	0.3408	1.8934	0.1920	3.1776	0.2797
Bdn (mrad)	0.0493	0.0494	0.0049	0.0410	0.0566	0.0952	0.0101
Ed (mrad)	0.0008	0.0007	0.0015	0.0100	0.0024	0.0218	0.0045
$RMSEp_{avg}$	0.0914	0.0834	0.1157	0.6481	0.0837	1.0982	0.0981
Puntaje Final	5	5	4	3	5	2	5

Tabla 3.8.: Tabla resumen de errores RMSE cartesianos y polares.
Fuente: Elaboración propia.

Para el criterio de "Tiempo de cómputo del algoritmo de estimación T_{cpu}" se tomó el tiempo de cómputo medio de los algoritmos para un total de 5226 iteraciones, considerando una puntuación de 5 para $T_{cpu_{avg}} < 0.00001$, puntuación de 4 para $0.00001 \leq RMSEp_{avg} < 0.0001$, puntuación de 3 para $0.0001 \leq RMSEp_{avg} < 0.001$, puntuación de 2 para $0.001 \leq RMSEp_{avg} < 0.01$, puntuación de 1 para $0.01 \leq RMSEp_{avg} < 0.02$ y puntuación de 0 para $0.02 \leq RMSEp_{avg}$.

Tiempo de cómputo	ESSMO	WZSMO	HOSMO	RED	ARED	FRED	UREDF
T_{cpu_avg} (seg)	0.000035	0.00042	0.000036	0.00004	0.00007	0.00007	0.0001
Puntaje Final	4	3	4	4	4	4	3

Tabla 3.9.: Tiempos de cómputo promedios.
Fuente: Elaboración propia.

Al asignar los puntajes por criterio de desempeño que corresponden y efectuar la suma total, se obtuvieron los siguientes resultados:

Criterios	ESSMO	WZSMO	HOSMO	RED	ARED	FRED	UREDF
N_ρ	5	4	2	2	5	3	5
N_ε	1	2	3	4	4	5	4
N_{ruido}	1	4	2	2	2	4	5
ξ_{est}	5	0	4	0	0	0	0
T_{es}	5	5	1	3	5	2	5
$RMSEp_{avg}$	5	5	4	3	5	2	5
$T_{cpu\_}avg$ (seg)	4	3	4	4	4	4	3
N_{var}	3	3	5	5	5	5	5
N_{est}	2	2	3	4	4	4	1
Total	31	28	28	29	34	29	33

Tabla 3.8.: Comparación de desempeño de observadores por criterios.
Fuente: Elaboración propia.

- Observador de Edwards-Spurgeon (ESSMO): 33 puntos
- Observador de Walcott-Zak (WZSMO): 32 puntos
- Observador Super twisting de orden superior (HOSMO): 30 puntos
- Diferenciador Robusto exacto (RED): 25.5 puntos
- Diferenciador Robusto exacto Adaptativo (ARED): 37 puntos
- Diferenciador Robusto exacto de filtrado estándar (FRED): 32 puntos
- Diferenciador Robusto exacto y uniforme filtrado (UREDF): 37 puntos

De los resultados obtenidos los observadores que obtuvieron los mayores puntajes fueron el diferenciador robusto exacto adaptativo (ARED) y el diferenciador robusto exacto y uniforme filtrado (UREDF), seguido del observador de Edwards-Spurgeon (ESSMO).

El diferenciador robusto exacto adaptativo (ARED) obtuvo el mayor puntaje porque posee la capacidad de brindar estimaciones exactas y uniformes hasta la cuarta derivada de la posición, solo teniendo como entrada la posición medida del blanco, siendo un algoritmo que sirve para mantener el traqueo del misil y estimar el vector de entrada de control al modelo. Sin embargo, se observó una ligera falta de exactitud para las variables de estado de velocidad y aceleración en la coordenada “y” durante la maniobra terminal. Esto se atribuye a las características de la perturbación “y” durante esos instantes de tiempo y no a un error del algoritmo propiamente dicho. Sin embargo, la estimación de posición “y” fue exacta y se mantuvo el traqueo adecuado del blanco.

El diferenciador robusto exacto y uniforme filtrado (UREDF) propuesto no es un diferenciador que permite obtener una buena estimación de la velocidad del blanco, sin embargo, provee una estimación exacta y uniforme de la posición mientras efectúa el filtrado de la señal, siendo esto una ventaja sobre el diferenciador robusto exacto de filtrado estándar, el cual se observó que no cuenta con buena robustez para la aplicación en estudio. Por tanto, este diferenciador da la posibilidad de trabajar como filtro de cualquier variable de estado medida, ya sea la posición o velocidad, mientras el orden de filtrado sea $n_f = 3$ y el orden de diferenciación $n = 1$, permitiendo brindar variables de estado filtradas para otros observadores de modos deslizantes, tales como el observador de Edwards-Spurgeon (ESSMO) y Walcott-Zak.

El observador de Edwards-Spurgeon obtuvo el mayor puntaje de los algoritmos “clásicos” debido a su excelente robustez y capacidad de reconstrucción de perturbaciones en la ausencia de ruido angular. Sin embargo, se verificó que este observador, sintetizándolo para ofrecer la mayor robustez posible, es muy sensible al ruido y genera castañeo. Esto fue posible de comprobar numéricamente al verificar la matriz de ganancias G_n. Por ejemplo, se puede apreciar que el valor numérico de los elementos $G_n(3,1)$, $G_n(3,2)$ y $G_l(3,1)$, $G_l(3,2)$ correspondientes a la fila de la estimación de la aceleración “x” son muy grandes, lo cual efectúan una amplificación del ruido considerable, siendo esto lo que ocasiona la sensibilidad al ruido de la variable de aceleración; esto sucede también para las coordenada “y” y “z”; se trató de modificar la síntesis del observador

para que el valor numérico de estos elementos disminuyera, sin embargo, la robustez del observador fue perjudicada.

Por otro lado, el observador de Walcott-Zak posee un mejor desempeño en un ambiente de ruido no gaussiano debido a su síntesis LQG/LMI. Este observador lograr atenuar el ruido no gaussiano pero no es una solución efectiva por si solo para el modelo mixto gaussiano planteado. Sin embargo, es un observador que ofrece muy buena robustez con menor sensibilidad al ruido no gaussiano.

3.12 Conclusiones del capítulo

De lo desarrollado en el presente capítulo se llegó a las siguientes conclusiones:

1) Los observadores de modos deslizantes "clásicos" poseen una excelente robustez en la ausencia de ruido.
2) El observador de Edwards-Spurgeon posee una alta sensibilidad al ruido angular, siendo necesario que este observador trabaje conjuntamente con un filtro de las variables medidas de ingreso al algoritmo para lograr un mejor desempeño.
3) El observador de Walcott-Zak ofrece una solución al problema de ruido debido a que sintoniza las ganancias lineales del observador por medio del algoritmo LQG. Sin embargo, es evidente que no atenúa el ruido angular completamente, pero ofrece mejores prestaciones de sensibilidad de ruido que el observador de Edwards-Spurgeon. Asimismo, su diseño permite que el usuario pueda sintonizar las ganancias para obtener una buena relación de filtrado de ruido y robustez ante perturbaciones.
4) De los observadores "clásicos" el más robusto es el observador de Edwards-Spurgeon, sin embargo, es altamente sensible al ruido. Por ello, el observador de Walcott-Zak es más versátil dado que ofrece cierta atenuación del ruido angular mientras mantiene una buena robustez.
5) Los observadores de Edwards-Spurgeon y Super Twisting de orden superior poseen la capacidad de reconstruir o estimar las perturbaciones al modelo, con grado alto de exactitud, siendo una ventaja comparativa respecto al observador de Walcott-Zak y estimadores Kalman. Sin embargo, esta ventaja no es muy relevante para el problema en estudio, pero interesante para aplicaciones de diagnóstico orientadas al control donde se puede efectuar la compensación de la señal de control a la planta o proceso.
6) Los algoritmos super twisting permiten la eliminación de castañeo o chattering obteniendo estimaciones más suaves de las variables de estado, a costa de un mayor tiempo de establecimiento en el colector deslizante. Asimismo, los algoritmos super twisting poseen la capacidad de estimar las variables de estado solo conociendo la posición medida del blanco. Esto es una ventaja respecto a los observadores de modos deslizantes clásicos estudiados debido a que estos necesitan mediciones de posición y velocidad para implementar sus respectivos algoritmos.
7) El diferenciador robusto exacto (RED) posee la capacidad de obtener una estimación del vector de entrada de control u_k, dado que obtiene mediante su algoritmo la derivada de la aceleración, pudiendo trabajar como el estimador de entrada de control que se necesita para el modelo dinámico planteado del blanco. Sin embargo, su robustez es limitada si es que se trabaja con una constante de Lipschitz fija y sus términos de orden bajo hacen que las convergencias no sean uniformes en un tiempo finito.
8) Se logró obtener una ley de adaptación de la constante de Lipschitz mediante una función de Lyapunov débil y añadir términos de orden alto al diferenciador robusto exacto, permitiendo obtener una mejor robustez ante perturbaciones variantes en el tiempo.
9) Se propuso el diferenciador robusto exacto y uniforme filtrado, el cual permite obtener una excelente capacidad de filtrado del ruido angular manteniendo la exactitud y uniformidad requerida durante maniobras senoidales y cosenoidales, integrando las capacidades de filtrado del diferenciador robusto exacto de filtrado estándar de Levant y las del filtro no lineal de mediana intra pulso. Esto permitió obtener mejores resultados de filtrado que el filtro de Partículas (PF), usualmente utilizado para traqueo de modelos dinámicos no lineales con mediciones no gaussianas.

Capítulo IV

Propuesta de solución de modos deslizantes

4.1 Introducción

A partir del estudio, diseño y comparación de los observadores de variables de estado basados en la teoría de control robusto de sistemas de estructura variable con modos deslizantes, en el presente capítulo se busca plantear la solución de modos deslizantes que permitirá efectuar el traqueo robusto de un blanco aéreo de alta maniobrabilidad. Al plantear la estructura teórica de la solución, se podrá efectuar la comparación de su desempeño con la arquitectura de modelo múltiple interactivo (IMM) en el capítulo V por medio de simulaciones.

4.2 Propuesta de solución de modos deslizantes

Para el problema de traqueo de un blanco aéreo de alta maniobrabilidad en un ambiente de alto ruido angular se ha visto proponer dos soluciones alternativas al traqueador de modelo múltiple interactivo (IMM), detalladas en las figuras 4.1 y 4.2.

En la primera solución se plantea el uso del diferenciador robusto exacto y uniforme filtrado con filtrado no lineal de mediana intra pulso para efectuar el filtrado de las posiciones ruidosas de radar y_k, para el caso de un radar a pulsos que no cuenta con medición de velocidad doppler. En la figura 4.1 se visualiza que el vector de mediciones de posición $y_k = [y_{k_x}\, y_{k_y}\, y_{k_z}]'$, previamente convertido de coordenadas polares a cartesianas, ingresa al algoritmo de pre procesamiento en secuencias o "batches", considerando un PRF de radar de 1800 Hz y un tiempo de muestreo de 0.02 segundos del algoritmo de estimación, lo que permite obtener 36 pulsos cada 0.02 segundos. A la salida del pre procesador se obtienen las secuencias o "batches" que serán utilizadas por el algoritmo uniforme del diferenciador robusto exacto y uniforme para filtrar las posiciones de radar por medio de la operación de mediana (UREDF). De la misma manera, ingresa el vector y_k (no filtrado) al UREDF para ser usado por la parte de diferenciación exacta de filtrado estándar del UREDF. A la salida del UREDF se obtiene la posición y velocidad filtrada del blanco en coordenadas cartesianas, sin embargo, la velocidad filtrada no es utilizada posteriormente debido a que este diferenciador fue diseñado solo para proveer una estimación suave, exacta y uniforme solo de la posición ruidosa del blanco.

La posición filtrada en coordenadas cartesianas ingresa al diferenciador robusto exacto adaptativo (ARED) para que este obtenga el vector de posiciones estimadas $z0_k = [z0x\ z0y\ z0z]'$, vector de velocidades estimadas $z1_k = [z1x\ z1y\ z1z]'$, aceleraciones estimadas $z2_k = [z2x\ z2y\ z2z]'$ y sobre aceleraciones estimadas $z3_k = [z3x\ z3y\ z3z]'$. Hasta este punto comprende la solución número uno de observación al problema en estudio, dado que el control de posición del arma, control de posición de radar y calculador de solución de tiro no son parte del estudio de esta tesis.

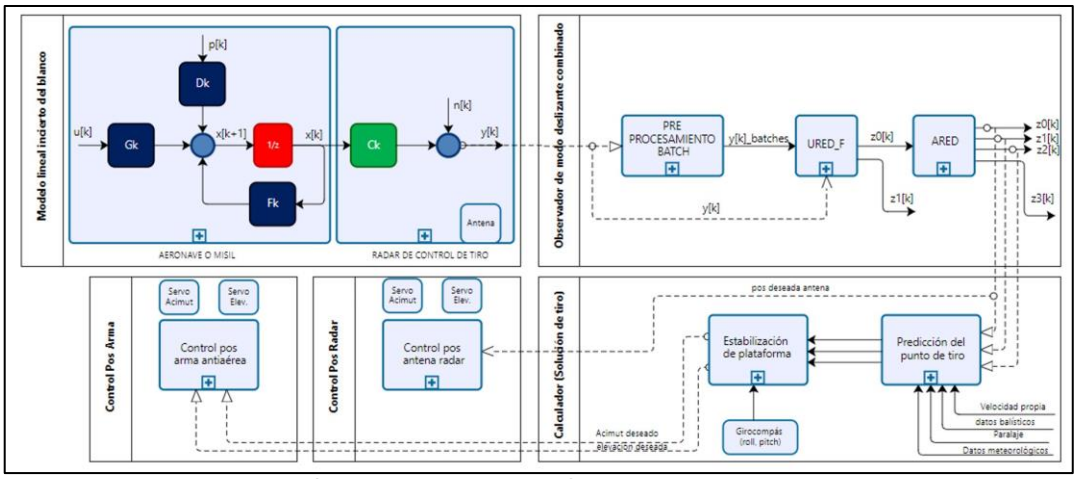

Figura 4.1.: Propuesta de solución de modos deslizantes número uno.
Fuente: Elaboración propia.

En la segunda propuesta de solución se asume que se cuenta con un radar de control de tiro pulse-doppler, con capacidad de medición de la posición y velocidad del blanco en coordenadas esféricas y se posee el vector de entrada de control estimado. Por ello, esta propuesta involucra el uso de un observador de Walcott-Zak por síntesis LQG/LMI (WZSMO) para la estimación de las variables de estado de posición, velocidad y aceleración del blanco, con la finalidad de verificar el buen funcionamiento de este observador cuando se le provee el vector de entrada de control estimado del blanco. Si bien es cierto, el observador de Edwards-Spurgeon obtuvo un mayor puntaje, el ruido remanente de las posiciones filtradas del blanco serían amplificados por las ganancias del observador, generando que las estimaciones de aceleración se pierdan. Por ello, se tuvo a bien seleccionar al observador de Walcott-Zak porque ofrece muy buena robustez y es menos sensible al ruido angular.

De acuerdo a lo señalado en la figura 4.2, el vector de medición de la posición y velocidad del blanco $y_k = \left[y_{k_x} y_{k_y} y_{k_z}\, y_{k_{vx}} y_{k_{vy}} y_{k_{vz}}\right]'$ ingresa al pre procesador batch y posteriormente las secuencias o batches de posición, conjuntamente con las mediciones de posición $y_{k_x}, y_{k_y}, y_{k_z}$ ingresan a un diferenciador robusto exacto y uniforme filtrado (UREDF). De la misma forma, las secuencias o batches de velocidad, conjuntamente con las mediciones de velocidad $y_{k_{vx}}, y_{k_{vy}}, y_{k_{vz}}$ ingresan a un segundo diferenciador robusto exacto y uniforme filtrado. A la salida de estos diferenciadores se obtienen los vectores de posición y velocidad filtrados z0_UREDF[k]=[z0x z0y z0z]' y z1_UREDF[k], respectivamente, los cuales ingresan al observador de Walcott-Zak (WZSMO) para que este pueda calcular las variables deslizantes de posición $\sigma = z0\_ESSMO[K] - z0\_URED\_F[K]$ y velocidad $\dot{\sigma} = z1\_ESSMO[K] - z1\_URED\_F[K]$, necesarias para cumplir con los requisitos de existencia del observador.

Asimismo, el vector de posición filtrado z0_UREDF[k] es utilizado por un diferenciador robusto exacto adaptativo para obtener el vector de entrada de control estimado al modelo $\hat{u}_k$. Por ello, mediante esta estructura el observador de Walcott-Zak logra obtener todas las señales que necesita para ejecutar su algoritmo de estimación. Sin embargo, cabe resaltar que la máxima exactitud del WZSMO estará supeditada a la máxima exactitud de diferenciación obtenible por el diferenciador UREDF, según la ecuación 3.136 del capítulo III:

$$sup_{f_{o,n}} sup_{t \geq t_k} \left| z_i - f_0^{(i)} \right| \geq K_{i,n} (2L)^{\frac{i}{n-1}} \varepsilon_0^{\frac{n+1-i}{n+1}}$$

Por otro lado, de acuerdo a los resultados obtenidos en el capítulo III, dado que el vector de entrada de control u_k posee las mismas unidades que las perturbaciones al modelo ξ_k (unidades de sobre aceleración o *m/s3*), al efectuar la diferenciación de la posición del blanco se observó que la sobre aceleración estimada, o tercera derivada de la posición, se aproximaba correctamente al vector de entrada de control real pero contando con la estimación de la perturbación real superpuesta, siendo esto un comportamiento no deseable. Por tanto, se deberá

evaluar el efecto de este comportamiento del vector de entrada de control estimado sobre las estimaciones efectuadas por el WZSMO.

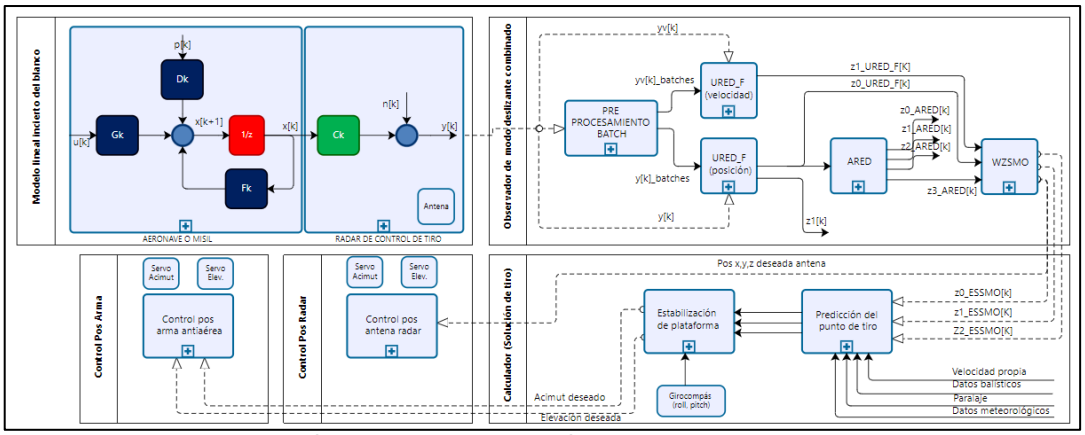

Figura 4.2.: Propuesta de solución de modos deslizantes número dos.
Fuente: Elaboración propia.

4.3 Conclusiones del capítulo

De lo desarrollado en el presente capítulo se llega a las siguientes conclusiones:

1) Existen posibles soluciones al problema en estudio mediante el uso de observadores de modos deslizantes de forma combinada, aprovechando la capacidad de filtrado de señales con ruido angular del observador robusto exacto y uniforme filtrado y la capacidad de exactitud y uniformidad con la que cuenta el diferenciador robusto exacto adaptativo y el observador de Walcott-Zak (WZSMO).
2) La exactitud máxima de estimación de ambas soluciones dependerá de la exactitud de diferenciación máxima que se pueda obtener por medio del diferenciador robusto exacto y uniforme filtrado (UREDF).
3) La efectividad de retro alimentación de la perturbación reconstruida para lograr mejores estimaciones de posición, velocidad y aceleración debe ser comprobada mediante simulaciones en el siguiente capítulo.
4) La efectividad de ambas soluciones en comparación al algoritmo interactivo de modelos múltiples (IMM) deberá ser verificada por medio de simulaciones en el capítulo V.

Capítulo V

Comparación de la propuesta de solución con el algoritmo IMM

5.1 Introducción

En el presente capítulo se efectúa la comparación de las dos soluciones propuestas de modos deslizantes con el algoritmo interactivo de modelos múltiples (IMM) por medio de simulaciones en MATLAB®. Cabe resaltar que existen diversas configuraciones del algoritmo interactivo de modelos múltiples, por lo que para la presente tesis se utilizó la función trackingIMM del toolbox de Sensor Fusion and Tracking de MATLAB® con la finalidad de obtener el mejor desempeño posible del citado algoritmo.

5.2 Simulación de la trayectoria del misil

La comparación de las soluciones propuestas de modos deslizantes y el algoritmo interactivo de modelos múltiples (IMM) fue realizada por medio de simulaciones, siendo el programa correspondiente detallado en el apéndice 5.1. Las condiciones de simulación fueron las siguientes:

- Modelo dinámico del blanco lineal incierto (2.23a) con correlación en el tiempo de la sobre aceleración $\alpha = 0$, ruido de proceso $w_k = 0$ y tiempo de muestreo *T=0.02 seg.*
- Modelo de medición del blanco lineal incierto (2.23b) que representa a un radar en banda X, PRF = 1800 Hz, con capacidad solo de medición de posición para el observador de modos deslizantes correspondiente a la solución uno (SMO1) y medición de posición y velocidad Doppler para el observador de modos deslizantes correspondiente a la solución dos (SMO2).
- Se consideró ruido de medición v_k simulado de acuerdo al modelo mixto gaussiano planteado en el capítulo II.
- Tiempo de simulación: 104.5 segs (5226 muestras adquiridas por el radar).
- Datos de traqueo iniciales:
  - Distancia inicial al blanco: 35 km
  - Elevación inicial al blanco: 0.01°
  - Acimut inicial al blanco: 030°
  - Velocidad inicial del blanco: 0.9 Mach
  - Rumbo verdadero inicial del blanco: 210°
- Tipo de trayectoria: Sea skimming con maniobra terminal helicoidal.

En lo que respecta al algoritmo interactivo de modelos múltiples, este fue configurado de la siguiente manera:

- Número de filtros del modelo múltiple interactivo: 4
- Filtro número 1: *Unscented Kalman Filter* (UKF) con modelo de aceleración constante.
- Filtro número 2: Cubature Kalman Filter (CKF) con modelo CT (Constante turn)
- Filtro número 3: Particle Filter (PF) con modelo de aceleración constante.

- Filtro número 4: Extended Kalman Filter con modelo de velocidad constante.
- Matriz de covarianza de estados inicial: eye(9)
- Matriz de covarianza del ruido inicial: 10*eye(9)
- Vector de estados iniciales: [11347.6 296.3 70.8 -2944.6 -726.15 -3.47 3.06 0.03 1.04]'
- Matriz de transición de probabilidades:

$$\prod ji = \begin{bmatrix} 0.99 & 0.01 & 0.00 & 0.00 \\ 0.01 & 0.98 & 0.01 & 0.00 \\ 0.00 & 0.01 & 0.98 & 0.01 \\ 0.00 & 0.00 & 0.01 & 0.99 \end{bmatrix}$$

La configuración de la solución número 1 (SMO1) fue la siguiente:

- Pre procesador de secuencia de pulsos (batches) de medición de posición: 36 pulsos por secuencia (solo para parte uniforme del diferenciador UREDF).
- Filtro para ruido angular: Diferenciador Robusto Exacto y uniforme filtrado (UREDF) de orden de filtrado $n_f = 3$ y orden de diferenciación $n = 1$.
- Observador de variables de estado del blanco aéreo de alta maniobrabilidad: Diferenciador robusto exacto adaptativo (ARED) de orden de diferenciación $n = 4$ (hasta la derivada de la sobre aceleración).

La configuración de la solución número 2 (SMO2) fue la siguiente:

- Pre procesador de secuencia de pulsos (batches) de medición de posición y velocidad doppler: 36 pulsos por secuencia (solo para parte uniforme del diferenciador UREDF de posición y el UREDF de velocidad).
- Filtro para ruido angular de posición: Diferenciador Robusto Exacto y uniforme filtrado (UREDF) de orden de filtrado $n_f = 3$ y orden de diferenciación $n = 1$.
- Filtro para ruido angular de velocidad: Diferenciador Robusto Exacto y uniforme filtrado (UREDF) de orden de filtrado $n_f = 3$ y orden de diferenciación $n = 1$.
- Estimador del vector de entrada de control al modelo: Diferenciador Robusto Exacto Adaptativo (ARED) de orden de diferenciación $n = 4$, con la finalidad de obtener la sobre aceleración del blanco.
- Observador de variables de estado del blanco aéreo de alta maniobrabilidad: Observador de Walcott-Zak (WZSMO) sintetizado de la siguiente forma:

$$G_l = \begin{bmatrix} 0.2801 & 0.0001 & 0 & 0 & 0 & 0 \\ 0.0001 & 0.3106 & 0 & 0 & 0 & 0 \\ 0.0053 & 0.5287 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2901 & 0.0001 & 0 & 0 \\ 0 & 0 & 0.0001 & 0.3106 & 0 & 0 \\ 0 & 0 & 0.0053 & 0.5287 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2800 & 0.0001 \\ 0 & 0 & 0 & 0 & 0.0001 & 0.3106 \\ 0 & 0 & 0 & 0 & 0.0053 & 0.5287 \end{bmatrix}$$

$W = diag([0.28\ 0.3\ 0.9\ 0.29\ 0.3\ 0.9\ 0.28\ 0.3\ 0.9])$
$Y = diag([7.98e-054.56e-051.39e-044.56e-055.18e-064.56e-05])$
$\sigma = [1000\ 0.1\ 0.1\ 0.01\ 100\ 0.01]^T$
$n = 1$

$$A_o = \begin{bmatrix} 0.7199 & 0.0199 & 2e-04 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0001 & 0.6894 & 0.02 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0053 & -0.5287 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.7099 & 0.0199 & 2e-04 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0001 & 0.6894 & 0.02 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0053 & -0.5287 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.7200 & 0.0199 & 2e-04 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.0001 & 0.6894 & 0.02 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.0053 & -0.5287 & 1 \end{bmatrix}$$

$$G_n = \begin{bmatrix} 0.0015 & 3.64e-05 & 0 & 0 & 0 & 0 \\ 3.64e-05 & 0.0034 & 0 & 0 & 0 & 0 \\ 0.0017 & 0.1689 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0014 & 3.61e-05 & 0 & 0 \\ 0 & 0 & 3.61e-05 & 0.0034 & 0 & 0 \\ 0 & 0 & 0.0017 & 0.1689 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0015 & 3.62e-05 \\ 0 & 0 & 0 & 0 & 3.62e-05 & 0.0034 \\ 0 & 0 & 0 & 0 & 0.0017 & 0.1689 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} 0.0021 & 0.0023 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0021 & 0.0023 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0021 & 0.0023 \end{bmatrix}$$

A continuación, se muestra el algoritmo implementado de la simulación de la trayectoria:

Algoritmo 5.1: Simulación de trayectoria SMO1, SMO2 vs IMM

Entradas: Señal de entrada de control $\boldsymbol{\mu_k}$**,** perturbaciones $\boldsymbol{\xi_k}$ y modelo mixto gaussiano de ruido

Salidas: VE reales del modelo $\boldsymbol{x_k}$**,** VE medidas del modelo $\boldsymbol{y_k}$**,** VE estimadas de posición "x", "y" y "z" $\boldsymbol{xhSMO1}$ de la solución número 1, VE estimadas $\boldsymbol{xhSMO2}$ de la solución número 2, y VE estimadas $\boldsymbol{xhIMM}$ del filtro de modelos múltiples interactivos, errores cartesianos, errores polares, error RMSE, tiempo de cómputo de los algoritmos en cada iteración TCPU y gráficos.

Inicio del programa:

1: Declara tiempo de muestreo $\boldsymbol{T}$**,** gravedad $\boldsymbol{g}$**,** constante de mach **M**, sobre aceleración máxima ***load_factor*** tiempo inicial de simulación $\boldsymbol{t_i}$**,** tiempo final de simulación t_{ff} y número total de muestras $\boldsymbol{nt}$**.**

2: Declara condiciones iniciales de traqueo al blanco: distancia $\boldsymbol{Ad}$**,** acimut verdadero $\boldsymbol{Bdn}$**,** elevación $\boldsymbol{Ed}$**,** velocidad $\boldsymbol{Vm}$ y rumbo verdadero $\boldsymbol{Rv}$**.**

3: Convierte posición de coordenadas polares a cartesianas usando $\boldsymbol{Ad, Bdn\ y\ Ed}$

4: Declara el vector de estados inicial $\boldsymbol{x_k(0)}$

5: Declara matrices del espacio de estados $\boldsymbol{F_k}$, $\boldsymbol{G_k}$, $\boldsymbol{D_k}$ y $\boldsymbol{C_k}$.

6: Obtén $\boldsymbol{u_k}$ (invoca a función ***gen_jerk_sea_skimming3.m*** **(programa del apéndice 2.5))**

7: Obtén $\boldsymbol{\xi_k}$ **(invoca** a función ***perturb_1.m*** **(programa del apéndice 2.6))**

8: Cargar ruido angular ***glint_puntos.mat*** **(datos guardados del programa del apéndice 2.7)**

9: Generar datos intra pulso de posición **(programa del apéndice 2.16*) batch_ykx batch_yky batch_ykz***

10: Generar datos intra pulso de velocidad doppler **(programa del apéndice 5.2*) batch_ykvx batch_ykvy batch_ykvz***

11: Inicializar del algoritmo IMM.

12: Genera una detección inicial: detection = objectDetection(1,[0.7*Adx_init 0.7*Ady_init 1.5*Adz_init]','MeasurementNoise',10, 'SensorIndex',1,'ObjectAttributes',{'Example object',5});

13: Inicializa los filtros a partir de la detección filter = {initcaukf(detection);initcapf(detection); initcvkf(detection);initctckf(detection)};

14: Función de conversion de los modelos: modelConv = @switchimm;

15: Define la matriz de transición de probabilidades transProb = [0.99 0.01 0 0;0.01 0.98 0.01 0.00; 0.00 0.01 0.98 0.01; 0.00 0.00 0.01 0.99]

16: Declara el vector de estados inicial initialState = 0.5*xk(0);

17: Inicializa el filtro IMM = trackingIMM('State',initialState,'StateCovariance', eye(9), 'TransitionProbabilities', transProb,'TrackingFilters',filter,'ModelConversionFcn', modelConv);

18: Ejecuta ***WZSMO_sintesis_SMO2.m*** para obtener las ganancias del algoritmo.

19: disp('Observador de modos deslizantes combinado vs Modelo Múltiple Interactivo (IMM)')

20: prompt('Simular con filtrado intrapulso [1], sin filtrado intrapulso [0]')

21: intra = input('prompt')

22: Para tt = ti:T:tff **Hacer**

23: Obtén $\boldsymbol{x_k}$ futuro.

24: Invoca función ***c2p.mat*** para convertir $\boldsymbol{x_k}$ en coordenadas polares.

25: Invoca función ***yk_noise.m*** para obtener y_k

26: Tic. Ejecuta UREDF**.m (programa del apéndice 2.15)** para obtener $\mathbf{xhURED\_F}$.

27: Ejecuta **ARED.m** para obtener xhSMO1 y uhSMO1. TCPU_SMO1 = toc;

28: Tic. Ejecuta **UREDFV.m** (igual a UREDF pero para filtrado de velocidad)

29: Obtén yk_SMO2 = [xhUREDF(1,1) xhUREDFV(1,1) xhUREDF(3,1) xhUREDFV(3,1) xhUREDF(5,1) xhUREDF(5,1)]'

30: uk_SMO2(:,k) = uhsMO1;

31: Ejecuta ***WZSMO.m*** para obtener xhSMO2 y fiSMO2.

32: TCPU_SMO2 = toc;

33: Tic. Ejecuta [xhIMM,phIMM] = predict(IMM,T) para predecir estimados futuros.

34: [xcIMM,pcIMM] = correct(IMM,[yk(1,1),yk(3,1),yk(5,1)]') para corregir con la medición disponible. TCPU_xh_IMM = toc.

35: Extrae errores polares, error RMSE, errores cartesianos, tiempo de cómputo de los algoritmos en cada iteración TCPU y almacena vectores para gráficas.

Algoritmo 5.1: Simulación de trayectoria SMO1, SMO2 vs IMM (continuación)

36: Fin Para
37: Conversión de unidades del vector de estados a Mn, Mach y g's.
38: Graficar
39: Fin de programa

Al efectuar la simulación del programa del apéndice 5.1 se visualiza en la figura 5.1 que todos los observadores aparentemente logran efectuar una buena estimación de la trayectoria del blanco aéreo de alta maniobrabilidad a partir de mediciones de posición y velocidad con alto ruido angular; mientras el algoritmo de modelo múltiple interactivo (IMM) efectúa el filtrado del ruido mediante el uso de los filtros Kalman y filtro de partículas, el SMO1 y SMO2 hace lo mismo con el diferenciador robusto exacto y uniforme filtrado (UREDF).

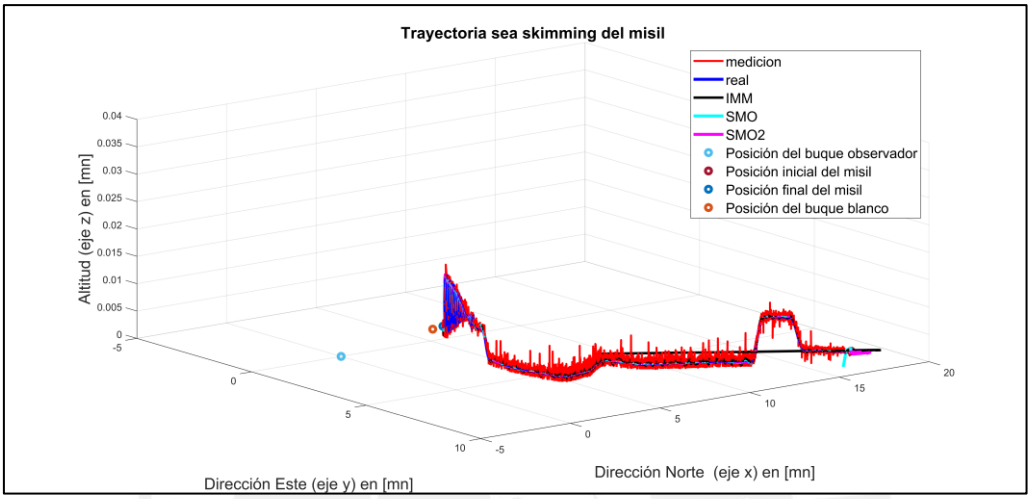

Figura 5.1.: Trayectoria tri dimensional del misil (SMO1, SMO2 vs IMM).
Fuente: Elaboración propia.

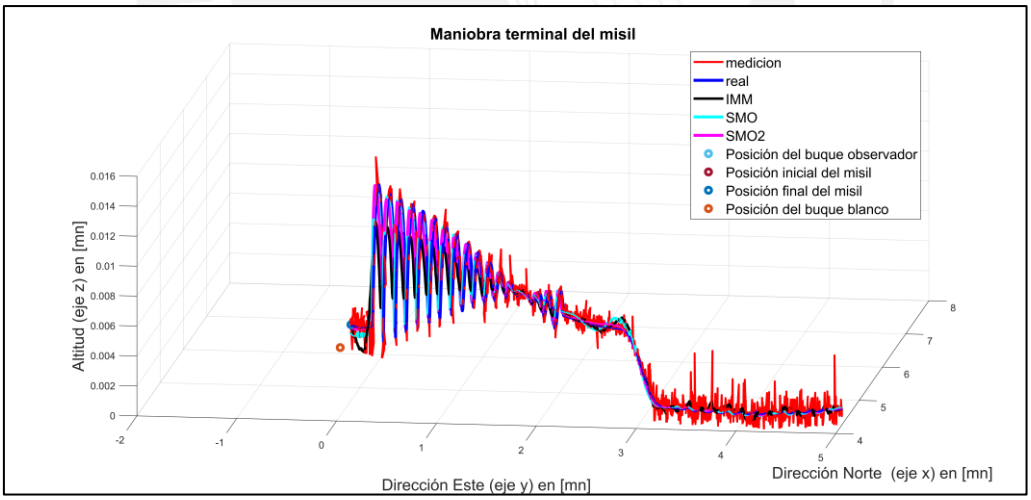

Figura 5.2.: Maniobra terminal del misil (SMO1, SMO2 vs IMM).
Fuente: Elaboración propia.

En lo que respecta a la maniobra terminal del misil, se observa que el algoritmo de modelos múltiples interactivo pierde robustez ante las no linealidades presentadas durante los momentos finales de la trayectoria. Si bien es cierto, en las simulaciones finales del capítulo III se pudo observar que los filtros Unscented Kalman Filter y Cubature Kalman Filter ofrecieron buena robustez ante este tipo de perturbaciones durante la maniobra terminal, se observa que cuando se utiliza estos filtros conjuntamente en el algoritmo de modelo múltiple interactivo (IMM) no se logra el mismo desempeño. Por tanto, se asume que la razón por la cual el IMM no obtiene una buena robustez es debido a que este detecta probabilísticamente cambios en la dinámica del modelo y efectúa la transición de un modelo a otro, pero la velocidad de detección y transición entre filtros no es lo suficientemente rápida para lograr una estimación óptima durante la maniobra terminal.

En la figura 5.3 se observa que el estimador que posee más rapidez de convergencia y exactitud durante los cambios de rumbo y maniobra terminal es el SMO2 (solución número 2). Sin embargo, se observa una sobre estimación de la velocidad durante las trayectorias senoidales y cosenoidales del SMO (solución número 1) y IMM. Asimismo, se puede observar que la filtración de ruido del SMO1 y SMO2 por medio del diferenciador robusto exacto y uniforme filtrado (UREDF) es superior a la del IMM, a pesar de que este cuente con el filtro de partículas (PF) para el filtrado durante los tramos de dinámica del blanco a velocidad constante.

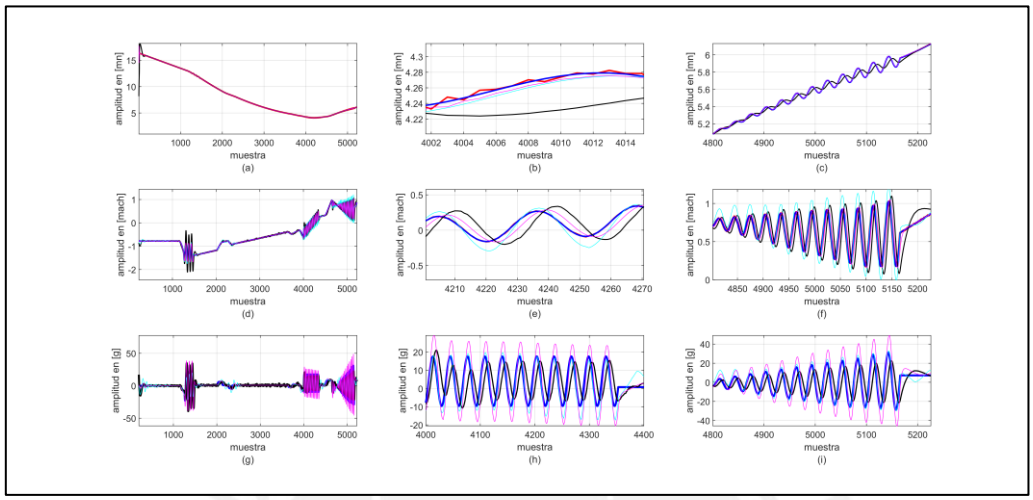

Figura 5.3.: Variable de estado de posición del misil en la coordenada "x" (SMO1 vs SMO2 vs IMM). Variable de estado real (línea azul). Variable de estado medida (línea roja). Variable de estado estimada IMM (línea negra). Variable de estado estimada SMO1 (línea cyan). Variable de estado estimada SMO2 (línea magenta). (a) Posición (k = [0 5226]). (b) Posición (k = [4002 4015]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Posición (k = [4208 4270]) (f) Posición (k = [4800 5226]). (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [4000 4270]) (f) Aceleración (k = [4000 4400])
Fuente: Elaboración propia.

En la figura 5.4 se observa que en la coordenada "y" el SMO1 posee menor rapidez de convergencia que los otros dos observadores, sin embargo, el IMM posee menor exactitud que el SMO1 y SMO2 durante las trayectorias senoidales y cosenoidales. Asimismo, las derivadas de velocidad y aceleración del SMO2 poseen menor amplificación del ruido que el SMO1 e IMM debido a que el SMO2 utiliza las mediciones de posición y velocidad Doppler filtrada, lo que permite diferenciar variables de estado con un menor nivel de ruido angular.

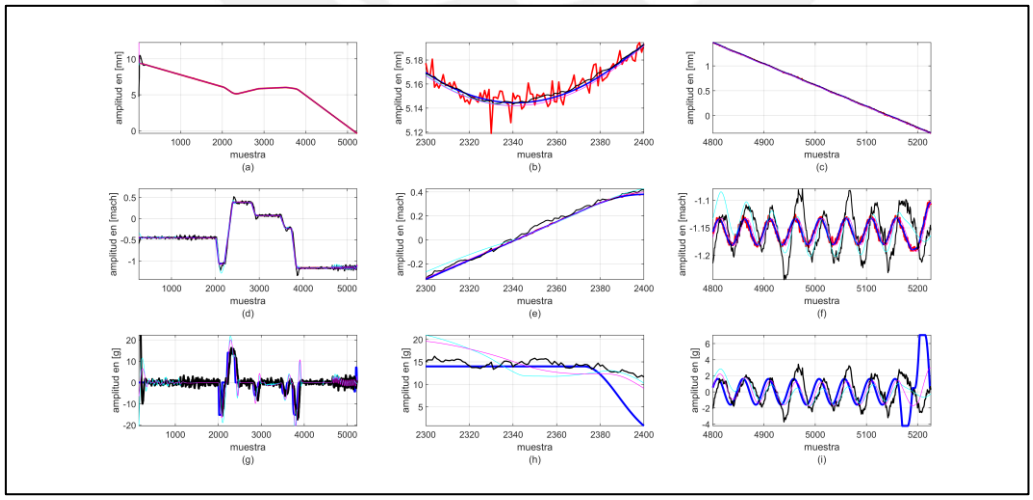

Figura 5.4.: Variable de estado de posición del misil en la coordenada "y" (SMO1 vs SMO2 vs IMM). Variable de estado real (línea azul). Variable de estado medida (línea roja). Variable de estado estimada IMM (línea negra). Variable de estado estimada SMO1 (línea cyan). Variable de estado estimada SMO2 (línea magenta). (a) Posición (k = [0 5226]). (b) Posición (k = [2300 2400]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Posición (k = [2300 2400]) (f) Posición (k = [4800 5226]). (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [2300 2400]) (f) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

En la figura 5.5 se aprecia que los observadores SMO1 y SMO2 poseen un mejor filtrado de ruido angular que el IMM. Por otro lado, la mejor exactitud durante la maniobra terminal la obtuvo el SMO2.

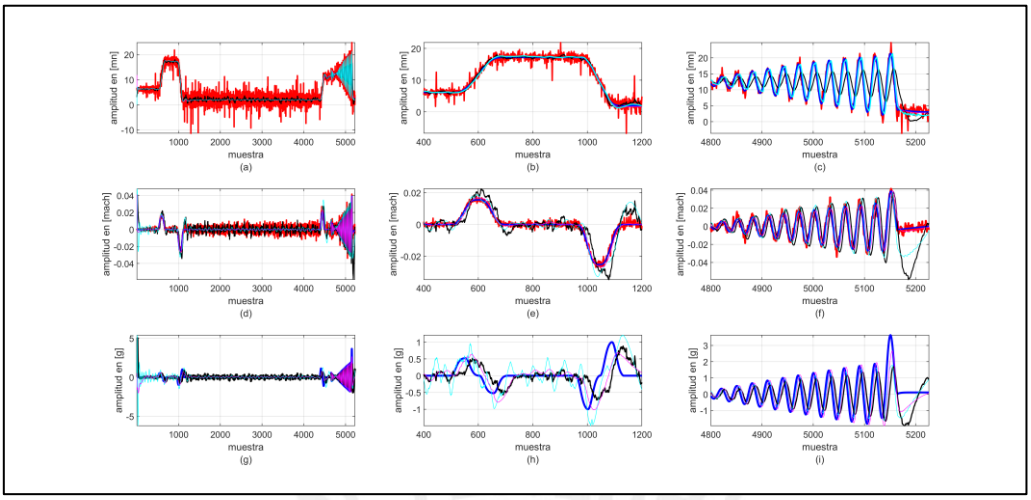

Figura 5.5.: Variable de estado de posición del misil en la coordenada “z” (SMO1 vs SMO2 vs IMM). Variable de estado real (línea azul). Variable de estado medida (línea roja). Variable de estado estimada IMM (línea negra). Variable de estado estimada SMO1 (línea cyan). Variable de estado estimada SMO2 (línea magenta). (a) Posición (k = [0 5226]). (b) Posición (k = [400 1200]) (c) Posición (k = [4800 5226]) (d) Velocidad (k = [0 5226]). (e) Posición (k = [400 1200]) (f) Posición (k = [4800 5226]). (g) Aceleración (k = [0 5226]). (h) Aceleración (k = [400 1200]) (f) Aceleración (k = [4800 5226])
Fuente: Elaboración propia.

En la figura 5.6 se comprueba que los errores cartesianos en las coordenadas “x”, “y” y “z” de los observadores SMO1 y SMO2 son menores que los errores cartesianos correspondientes al IMM. Sin embargo, en la coordenada “y” se observa un mejor traqueo del IMM respecto a los otros dos observadores, dado que estos presentan un sesgo durante la maniobra terminal. Por otro lado, en la figura 5.7 se observa que el SMO1 y SMO2 mantienen los errores polares dentro de los límites permisibles, a diferencia del IMM que sus errores polares de distancia y acimut salen de los límites permisibles durante algunos instantes de la trayectoria del misil.

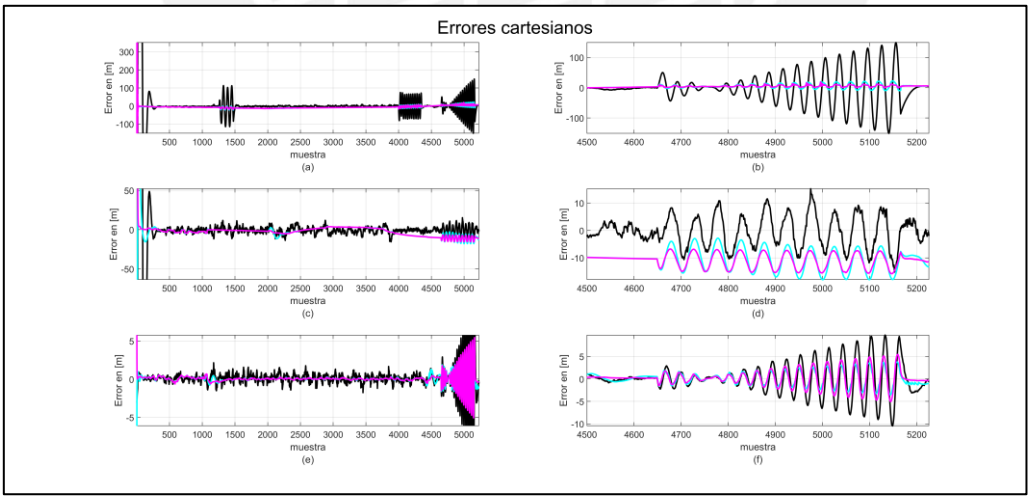

Figura 5.6.: Errores cartesianos (IMM vs SMO1 vs SMO2). IMM (línea negra). SMO1 (línea magenta). SMO2 (línea cyan). (a) Error de posición coordenada “x” (k = [0 5226]). (b) Error de posición coordenada “x” (k = [4800 5226]).(c) Error de posición coordenada “y” (k = [0 5226]). (d) Error de posición coordenada “y” (k = [4800 5226]).(e) Error de posición coordenada “z” (k = [0 5226]). (f) Error de posición coordenada “z” (k = [4800 5226]).
Fuente: Elaboración propia.

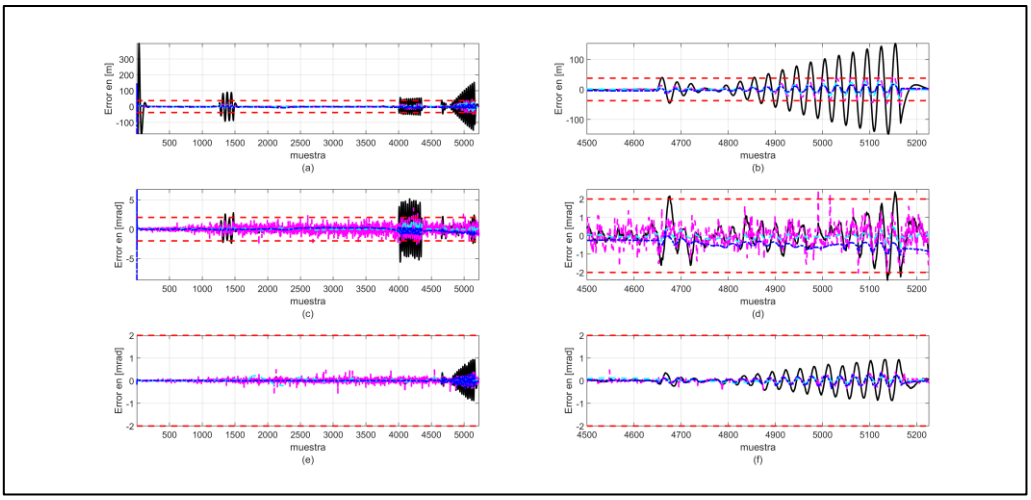

Figura 5.7.: Errores polares (IMM vs SMO1 vs SMO2). IMM (línea negra). SMO1 (línea magenta). SMO2 (línea cyan). (a) Error en alcance Ad (k = [0 5226]). (b) Error en alcance Ad (k = [4500 5226]). (c) Error en azimuth verdadero Bdn (k = [0 5226]). (d) Error en azimuth verdadero Bdn (k = [4500 5226]). (e) Error en elevación Ed (k = [0 5226]). (f) Error en elevación Ed (k = [4500 5226]).
Fuente: Elaboración propia.

De acuerdo a los cálculos efectuados de tiempo de cómputo de los algoritmos de observación, el IMM obtuvo un tiempo de cómputo promedio de 0.0199 segundos, lo cual muestra que en muchas ocasiones el algoritmo no se ejecutó en un tiempo menor que el tiempo de muestreo de 0.02 segundos. Por otro lado, los observadores SMO1 y SMO2 obtuvieron un tiempo de cómputo promedio de 0.00026310 segundos y 0.00041670 segundos, respetivamente, siendo evidente que el SMO2 posee un mayor tiempo de cómputo debido a que ejecuta secuencialmente el algoritmo de filtración de ruido de posición, el algoritmo de filtración de ruido de velocidad, el algoritmo de estimación del vector de control y el algoritmo de estimación de variables de estado.

5.3 Conclusiones del capítulo

Al finalizar el capítulo se llegó a las siguientes conclusiones:

1) Las soluciones de modos deslizantes número 1 (SMO1) y número 2 (SMO2) permiten la estimación de variables de estado de posición, velocidad y aceleración de manera efectiva, contando con buena filtración del ruido angular mientras preserva buena robustez ante perturbaciones o no linealidades no contempladas en el modelamiento.
2) Los observadores SMO1 y SMO2 poseen mejor filtración del ruido angular o glint que el observador IMM.
3) Los observadores SMO1 y SMO2 poseen mejor robustez que el observador IMM ante perturbaciones o no linealidades no contempladas en el modelo. Sin embargo, existe lugar para mejora de los algoritmos.
4) Se observó que la transición probabilística entre filtros IMM es lenta ante la perturbación no lineal inyectada al modelo durante la maniobra terminal, generando que el IMM no converja rápidamente a la trayectoria real del misil; se deberá probar con otras estructuras de IMM posteriormente.
5) Se ha comprobado que la solución SMO2 posee mayor rapidez de convergencia y robustez que el SMO1 durante los cambios de rumbo y trayectorias senoidales y cosenoidales, debido a que el SMO2 posee dentro de su diseño a un observador de modos deslizantes “clásico” (Walcott-Zak), a diferencia del SMO1 que efectúa la estimación de variables de estado por medio del diferenciador robusto exacto adaptativo (ARED) a base del algoritmo super twisting; se observó en el capítulo III que los observadores y diferenciadores a base del algoritmo super twisting son más lentos que los algoritmos de modos deslizantes “clásicos” debido a que efectúan super torsiones en la etapa final de su trayectoria hacia el origen del plano de estados. Por tanto, el observador de Walcott-Zak, habiendo filtrado las variables medidas de posición y velocidad previamente por medio del diferenciador robusto exacto y uniforme (UREDF), se muestra como la solución más robusta para el problema planteado. Sin embargo, es necesario contar con la medición de velocidad Doppler y la estimación del vector de

entrada de control para poder implementarlo, siendo esto una limitante para los casos en que el radar no cuente con capacidad Doppler.

6) Se comprobó mediante simulaciones que el IMM es computacionalmente más costoso que los observadores propuestos de modos deslizantes SMO1 y SMO2.
7) Todavía se encuentra pendiente desarrollar leyes de adaptación para el diferenciador robusto exacto filtrado y uniforme, a fin de brindar derivadas exactas de la velocidad y aceleración.

Capítulo VI

Propuesta de implementación

6.1 Introducción

En el presente capítulo se presenta una propuesta de implementación de los algoritmos de observación de modos deslizantes para efectuar la estimación de variables de estado de posición, velocidad y aceleración de un blanco aéreo de alta maniobrabilidad mediante el uso de un sistema PXI (PCI eXtension for Instrumentation) de la empresa National Instruments. El uso de este sismeta permitiría obtener un alto rendimiento y precisión en la adquisición de datos de posición y velocidad provenientes del radar de control de tiro y ejecutar los algoritmos de observación y diferenciación propuestos en tiempo real.

6.2 Conceptos generales del sistema PXI

El sistema PXI de la empresa National Instruments (NI) es un hardware especializado para aplicaciones de instrumentación y control relacionadas a pruebas de producción y validación, permitiendo obtener un alto rendimiento y excelente temporización y sincronización de tareas. El PXI obtiene un alto rendimiento debido a que incorpora en sus hardware procesadores multinúcleo, FPGAs (Field-Programmable Gate Arrays), entre otros, para tareas de concurrencia en tiempo real, y posee una alta precisión en la adquisición de datos. Asimismo, es un sistema que posee una arquitectura escalable mediante el uso de distintos módulos de acuerdo a los requisitos de instrumentación y control.

Figura 6.1.: Sistema PXI.
Fuente: National Instruments.

El sistema PXI está compuesto de tres componentes principales: el chasis PXI, los módulos y el controlador. El chasis PXI proporciona la alimentación, refrigeración y bus de comunicación del sistema, permitiendo contar, dependiendo del modelo, con 4 a 18 alojamientos para los módulos del sistema. Cabe resaltar que el chasis implementa tecnología comercial de comunicación de tipo PCI (Peripheral Component Interconnect) y PCI express basadas en PC. Por otro lado, el controlador PXI es un componente que permite ejecutar la aplicación del sistema PXI de forma embebida o ejecutarla desde una PC de forma remota. Los controladores PXI ofrecidos por National Instruments pueden ser:

- Controlador PXI, el cual proporciona una solución embebida, compacta y de alto rendimiento para sistemas PXI, compactPCI y PCI express.

- Módulo de control remoto PXI, el cual permite el control del sistema PXI y PXI express desde una PC o laptop.
- Dispositivo controlador para PXI, el cual permite el control de sistemas PXI y PXI express a través de una interfaz MXI para un formato de montaje en rack.

Finalmente, los módulos del sistema PXI permiten ejecutar una infinidad de tareas, dependiendo de la aplicación, tales como adquisición de datos, activación y sincronización de dispositivos, medición de señales, función de osciloscopio, multímetros digitales, entre otros.

En lo que respecta al software utilizado por el sistema PXI, National Instruments proporciona entornos de programación, software de aplicación, complementos de software y software suites para adquisición de datos y control, pruebas electrónicas e instrumentación y diseño y pruebas de dispositivos inalámbricos. Cabe resaltar que el entorno de programación mas utilizado es el LabView y Windows CVI.

6.3 Implementación del hardware

En este apartado se detalla el diagrama general de la propuesta de implementación y la configuración del sistema PXI que permitirá la adquisición de los datos de posición y velocidad (si es que se cuenta con un radar pulse-doppler) y la implementación del algoritmo de observación de variables de estado.

6.3.1 Sistema de control de tiro a intervenir

Para la presente tesis se ha considerado al sistema MK-92 mod. 2 como el sistema de control de tiro a intervenir, el cual es un sistema ampliamente utilizado por distintas marinas para el traqueo de blancos de superficie y aéreos. Este radar cuenta con la versión americana, denominada MK-92, el cual se encuentra a bordo de unidades de superficie tales como las fragatas Oliver-Hazard Perry de los Estados Unidos de Norteamérica y fragatas clase Adelaide de Australia. Asimismo, cuenta con una versión de exportación no americana denominada WM-25, la cual estuvo embarcada en el crucero ligero misilero “B.A.P. Almirante Grau” de la Marina de Guerra del Perú. Asimismo, en la actualidad se cuenta la Marina de Guerra del Perú cuenta con la versión reducida del sistema WM-25 denominado WM-28, a bordo de la corbeta misilera clase “Pohang” B.A.P. “Ferré”.

El sistema de control de tiro MK-92 mod. 2 provee la capacidad de búsqueda, adquisición, traqueo y designación de blancos aéreos y de superficie, así como la asignación de armas tales como cañones y lanzadores de misiles [119]. El sistema puede contar con diferentes tipos de configuraciones dependiendo de la clase de buque en el que se encuentra embarcado. Por ejemplo, el sistema WM-28, a bordo de la corbeta misilera B.A.P. “Ferré” de la Marina de Guerra del Perú, es considerado como un sistema reducido de exportación europea del sistema de control de tiro MK-92 mod. 2 debido a que no cuenta con el radar de iluminación de onda continua (CW) denominado STIR. Por otro lado, el sistema de control de tiro MK-92 mod. 2 cuenta con la configuración que se detalla en la figura 6.2, siendo sus partes componentes principales las siguientes [119]:

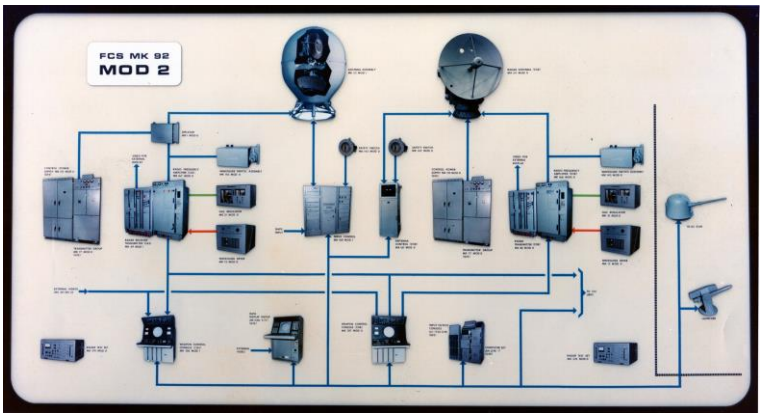

Figura 6.2.: Diagrama general del sistema de control de tiro MK-92 mod. 2.
Fuente: Thales group.

- Sistema combinado de antena (CAS): Consiste en una antena de búsqueda y una antena de traqueo de blancos aéreos mono pulsada, utilizando solo un transmisor de radar en banda “X” para la búsqueda de blancos aéreos y de superficie, traqueo-mientras-busca (TWS) de blancos de superficie y modo de navegación. Asimismo, el mismo transmisor de radar permite el traqueo de blancos aéreos por medio de la antena mono pulsada del sistema combinado. Cabe resaltar que cuenta adicionalmente con un transmisor de onda continua (CW) que permite la iluminación del blanco aéreo para el guiado de misiles antiaéreos.
- Cuenta con un segundo radar de control de tiro denominado STIR para la detección y traqueo de blancos aéreos y de superficie a largo alcance. Asimismo, posee la capacidad de iluminación del blanco para el guiado de misiles antiaéreos.
- Dos consolas de armas y una consola de display de datos que permite el control integrado y visualización para la vigilancia, detección, designación, traqueo, asignación y evaluación de daños del blanco.
- Computador de tiro digital que efectúa cálculos balísticos para los cañones y lanzadores de misiles antiaéreos STANDARD, provee de órdenes pre lanzamiento del misil STANDARD, entre otros.
- Varios gabinetes correspondientes a transmisores, receptores, fuentes de poder, control de servomecanismos, entre otros, necesarios para la operación del sistema.

Asimismo, el sistema de control de tiro MK-92 mod. 2 posee las siguientes especificaciones principales:

- Frecuencia de transmisión: 8-10 GHz
- Ancho de pulso (PW): 0.22 y 0.45 useg
- Frecuencia de repetición de pulsos (PRF): 1800 y 3600 pulsos/seg
- Período de rotación de antena: 60 rpm

En el gráfico 6.1 se puede observar que los datos de posición y velocidad del blanco en coordenadas polares deben ser extraídos del bus de datos entre la unidad MK-39 mod 1 (transmisor-receptor del sistema combinado de antenas) y la unidad MK-102 mod 1 (Consola de control de armas). Posteriormente, estos datos deben ser adquiridos por el sistema PXI para su conversión digital-analógica.

6.3.2 Sistema PXI propuesto

El sistema PXI deberá contar con una computadora embebida PXIe-8880-RT (con sistema operativo real time), de acuerdo a lo descrito en el diagrama de bloques de la figura 6.3, la cual deberá ser programada en el software LabWindows/CVI (programación C) o LabView (Programación gráfica) para ejecutar los algoritmos de observación de modos deslizantes en tiempo real.

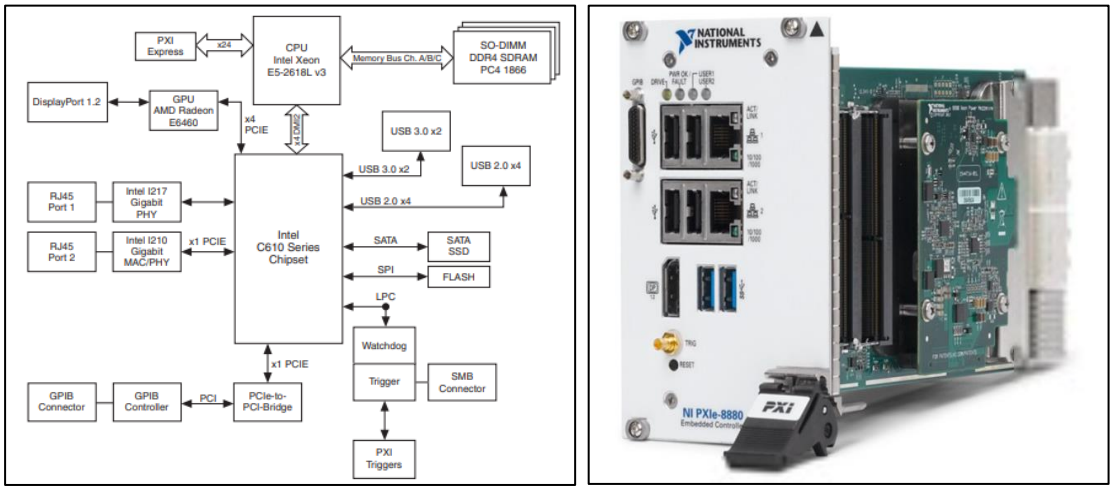

Figura 6.3.: Computadora embebida PXIe-8880-RT.
Fuente: National Instruments [120].

La computadora embebida de la figura 6.3 posee las siguientes partes componentes:

- Tres 72-bit DDR4 SDRAM SODIMM sockets, con una capacidad de 8GB de ram por socket, hasta un total de 24 GB.
- Trigger SMB a PXI express permite contar con una conexión ruteable de los triggers PXI express de y hacia el SMB en el panel frontal del sistema.
- Puerto de video DisplayPort 1.2 integrado a una tarjeta de video GPU embebida AMD Radeon E6460
- WatchDog timer que permite resetear el controlador o generar un trigger.
- Chipset Intel serie C610 que permite la conexión con los buses SPI, USB, serial ATA, PCI express y LPC.
- Disco duro de estado sólido serial ATA de 240 GB a más, permitiendo velocidades de transferencia de hasta 600 MB/s.
- Conector PXI express que conecta a la computadora embebida NI PXIe-8880 al backplane PXI express/CompactPCI express.
- Puertos ethernet gigabit I210 y I217 que se conectan a interfaces ethernet de 10Mbit, 100Mbit o 1000 Mbit
- Interfaz GPIB.

Por otro lado, el sistema PXI deberá contar con un módulo E/S multifunción PXIe-6375 para la adquisición de las mediciones de distancia, acimut, elevación y la velocidad doppler (***Ad, Bdn, Ed y Vm***) del blanco en coordenadas polares, extraídas del bus de datos del sistema de control de tiro MK-92 mod. 2. El módulo PXIe-6375 provee entradas y salidas analógicas y digitales con conversión de 16 bits y cuatro contadores/temporizadores de 32 bits con tecnología NI-STC3 para PWM, codificador, frecuencia, conteo de eventos, entre otros.

Figura 6.4.: Módulo PXIe-6375
Fuente: National Instruments [120].

Las características técnicas del módulo PXIe-6375 se detallan a continuación:

Entradas analógicas	
Número de canales	104 (Diferenciales), 208 (Single-Ended)
Resolución ADC	16 bits
Ratio de muestreo	Canal simple (máximo): 3.846 MS/s
	Multi canal (máximo): 1 MS/s
Resolución de tiempo	10 ns
Exactitud de tiempo	50 ppm del ratio de muestreo
Acoplamiento de entrada	DC
Rango de entrada	$\pm 0.1, \pm 0.2, \pm 0.5, \pm 1, \pm 2, \pm 5\ y\ \pm 10V$
CMRR (DC a 60Hz)	100 dB
Triggers Analógicos	
Número de triggers	1

Funciones	Trigger de inicio, trigger de referencia, trigger de pausa, reloj de muestreo, reloj de conversión, base de tiempo del reloj de muestreo
Modos	Triggering de flanco analógico, Triggering de flanco analógico con hysteresis y triggering de ventana analógica
Salidas analógicas	
Número de canales	2
Resolución DAC	16 bits
DNL	$\pm 1\ LSB$
Ratio de actualización máximo	1 canal – 2.86 MS/s 2 canales – 2.00 MS/s
Exactitud en tiempo	50 ppm del tiempo de muestreo
Resolución en tiempo	10 ns
Rango de salida	$\pm 10, \pm 5V, \pm referencia\ externa\ en\ APFI\ 0$
Contadores de propósito general	
Numero de contadores/timers	4
Resolución	32 bits
Relojes base internos	100 MHz, 20 MHz, 100 KHz
Frecuencia de reloj base externa	0 a 25 MHz; 0 a 100 MHz en PXIe-DSTAR<A,B>
Exactitud reloj base	50 ppm
Generador de frecuencia	
Número de canales	1
Relojes base	20 MHz, 10 MHz y 100 KHz
Divisores	1 a 16
Exactitud del reloj base	50 ppm
PLL (Phase-Locked Loop)	
Número de PLLs	1
Señal de referencia PLL	PXIe-DSTAR <A,B> - 10,20,100 MHz PXIe-STAR - 10,20 MHz PXIe-CLK100 - 100 MHz PXI-TRIG<0..7> - 10,20 MHz PFI<0..15> - 10,20 MHz
Triggers digitales externos	
Fuente	Cualquier PFI, PXIe-DSTAR<A,B>,PXI_TRIG,PXI-STAR
Bus de trigger dispositivo-dispositivo	
Fuente de entrada	PXI_TRIG<0..7>,PXI_STAR,PXIe_DSTAR<A,B>

Como componente final del sistema PXI se eligió un chasis express 3U PXIe-1071, de cuatro ranuras, para alojar a la computadora embebida PXIe-8880-RT y los dos módulos de adquisición de datos PXIe-6375. Este chasis posee la capacidad de aceptar módulos 3U PXI Express, Compact PCI Express y slots compatibles híbridos PXI-1/CompactPCI. En la figura 6.4, sub figura izquierda se muestra el panel frontal del chasis, observándose que cuenta un alojamiento para la computadora embebida o controlador del sistema y hasta tres alojamientos para módulos adicionales, que para la aplicación en estudio serán cubiertos por dos módulos de adquisición de datos.

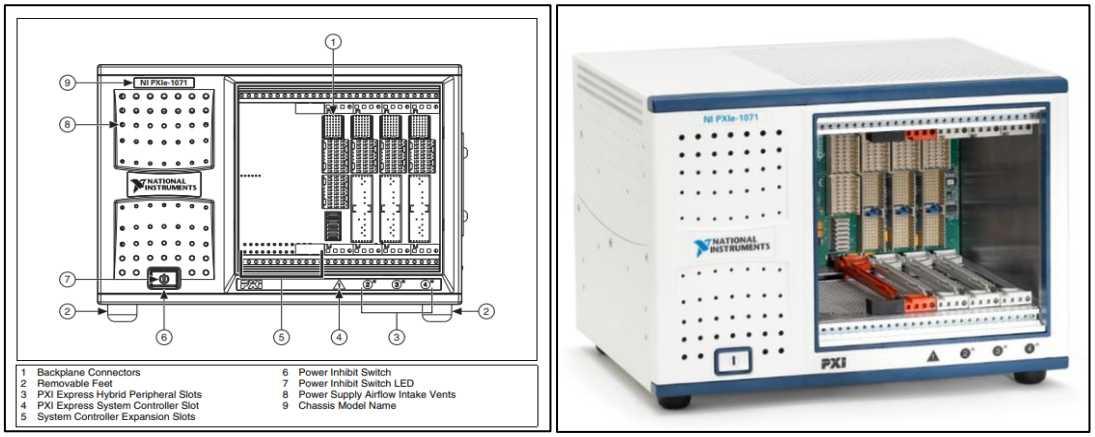

Figura 6.4.: Módulo PXIe-1071
Fuente: National Instruments [121].

Finalmente, como software de programación se puede optar por el entorno de programación gráfico LabView para Windows o con el entorno de programación C LabWindows/CVI Full Development System. Se recomendaría utilizar el entorno de programación gráfico de LabView para implementar el prototipo de las soluciones de modos deslizantes planteadas en el capítulo IV debido a que este entorno permite contar con un entorno visual en una PC o laptop con un sistema operativo Windows, siendo esto una ayuda para verificar el comportamiento de los observadores. Sin embargo, si después del prototipo se requiere desplegar el sistema PXI se recomendaría efectuar la programación en C mediante el uso de LabWindows/CVI, lo que obligaría a optar por el diseño de un software adicional que permita visualizar los parámetros de observación del sistema PXI en la PC o laptop con un sistema operativo a elección del usuario.

6.3.3 Diagrama general de la propuesta de implementación

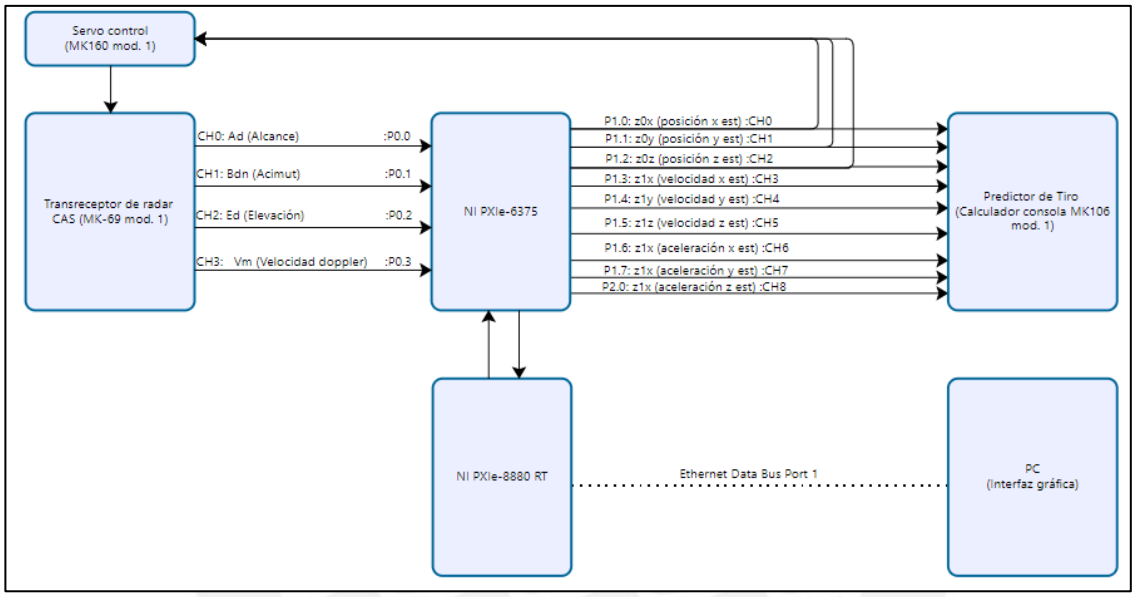

Figura 6.5.: Diagrama general de la propuesta de implementación.
Fuente: Elaboración propia.

En la figura 6.5 se visualiza el diagrama general del sistema, el cual cuenta con cuatro componentes fundamentales: transreceptor de radar CAS MK-69 mod.1, módulo E/S multifunción PXIe-6375, computadora embebida real time PXIe-8880-RT, unidad de servo control MK160 mod.1, predictor de tiro de la consola MK106 mod. 1 y una PC o laptop remota para la interfaz gráfica. Las mediciones digitales de radar de distancia, acimut, elevación y velocidad Doppler son extraídas del bus de datos del transreceptor de radar CAS MK-69 mod.1 y son adquiridas por el módulo E/S multifunción PXIe-6375 del sistema PXI de acuerdo al siguiente detalle:

- Ad (Distancia): Medición de distancia al blanco. Extraída del radar de control de tiro por el canal CH0 e ingresa al módulo PXIe-6375 por el puerto P0.0 (Port 0 Digital I/O Channel 0). El puerto P0.0 deberá ser configurado como una entrada digital.
- Bdn (Acimut): Medición de acimut al blanco. Extraída del radar de control de tiro por el canal CH1 e ingresa al módulo PXIe-6375 por el puerto P0.1 (Port 0 Digital I/O Channel 1). El puerto P0.1 deberá ser configurado como una entrada digital.
- Ed (Elevación): Medición de elevación al blanco. Extraída del radar de control de tiro por el canal CH2 e ingresa al módulo PXIe-6375 por el puerto P0.2 (Port 0 Digital I/O Channel 2). El puerto P0.2 deberá ser configurado como una entrada digital.
- Vm (Velocidad Doppler. Solo si se encuentra disponible): Medición de velocidad Doppler del blanco. Extraída del radar de control de tiro por el canal CH3 e ingresa al módulo PXIe-6375 por el puerto P0.3 (Port 0 Digital I/O Channel 3). El puerto P0.3 deberá ser configurado como una entrada digital.

Posteriormente, la computadora embebida real time PXIe-8880-RT utiliza las mediciones adquiridas para ejecutar el algoritmo de observación de modos deslizantes, enviando por el

puerto número uno ethernet los datos de posición y velocidad medida del blanco en coordenadas cartesianas y polares, la posición, velocidad y aceleración estimadas del blanco en coordenadas cartesianas, los errores cartesianos de posición y los errores polares de posición a la PC remota que contará con la interfaz gráfica para visualizar todos estos datos. Asimismo, las variables de estado estimadas por el observador son enviadas al predictor de tiro de la consola MK106 mod. 1 del sistema de control de tiro MK92 mod.2 por medio del módulo PXIe-6375, de acuerdo al siguiente detalle:

- z0x: Posición estimada del blanco en la coordenada “x”. Sale del módulo PXIe-6375 por el puerto P1.0 (Port 1 Digital I/O Channel 0). El puerto P1.0 deberá ser configurado como una salida digital.
- z0y: Posición estimada del blanco en la coordenada “y”. Sale del módulo PXIe-6375 por el puerto P1.1 (Port 1 Digital I/O Channel 1). El puerto P1.1 deberá ser configurado como una salida digital.
- z0z: Posición estimada del blanco en la coordenada “z”. Sale del módulo PXIe-6375 por el puerto P1.2 (Port 1 Digital I/O Channel 2). El puerto P1.2 deberá ser configurado como una salida digital.
- z1x: Velocidad estimada del blanco en la coordenada “x”. Sale del módulo PXIe-6375 por el puerto P1.3 (Port 1 Digital I/O Channel 3). El puerto P1.3 deberá ser configurado como una salida digital.
- z1y: Velocidad estimada del blanco en la coordenada “y”. Sale del módulo PXIe-6375 por el puerto P1.4 (Port 1 Digital I/O Channel 4). El puerto P1.4 deberá ser configurado como una salida digital.
- z1z: Velocidad estimada del blanco en la coordenada “z”. Sale del módulo PXIe-6375 por el puerto P1.5 (Port 1 Digital I/O Channel 5). El puerto P1.5 deberá ser configurado como una salida digital.
- z2x: Aceleración estimada del blanco en la coordenada “x”. Sale del módulo PXIe-6375 por el puerto P1.6 (Port 1 Digital I/O Channel 6). El puerto P1.6 deberá ser configurado como una salida digital.
- z2y: Aceleración estimada del blanco en la coordenada “y”. Sale del módulo PXIe-6375 por el puerto P1.7 (Port 1 Digital I/O Channel 7). El puerto P1.7 deberá ser configurado como una salida digital.
- z2z: Aceleración estimada del blanco en la coordenada “z”. Sale del módulo PXIe-6375 por el puerto P2.0 (Port 2 Digital I/O Channel 0). El puerto P2.0 deberá ser configurado como una salida digital.

Las salidas P1.0, P1.1 y P1.2, correspondientes a la posición estimada del blanco en coordenadas cartesianas, son enviadas al módulo de servo control MK-160 mod. 1, a fin de reposicionar la línea de mira o *Line-of-sight* del radar hacia la posición estimada del blanco.

Por último, el prototipo de la interfaz gráfica de la PC remota al sistema PXI podría ser desarrollado en el entorno de programación gráfico Lab View o en el software SCADA Lookout (debido a que es un software propietario de National Instruments y es compatible con los entornos de programación de LabView y LabWindows CVI). Al respecto, deberá contar como mínimo con las siguientes características:

- Pantalla gráfica para visualizar la posición del blanco en coordenadas esféricas de distancia, acimut y elevación.
- Pantalla gráfica para visualizar las variables de estado reales del blanco versus las variables de estado estimadas del blanco en coordenadas cartesianas “x”, “y” y “z”.
- Pantalla gráfica para visualizar los errores cartesianos y polares de estimación.
- Pantalla alfa numérica para configuración del observador de modos deslizantes.
- Pantalla alfa numérica con los datos de posición y velocidad en coordenadas esféricas y cartesianas del blanco, así como datos de posición, velocidad y aceleración en coordenadas esféricas y cartesianas estimadas.
- Pantalla alfa numérica para el ingreso de parámetros para la simulación de la trayectoria del blanco aéreo a modo de prueba.

Conclusiones

Para la presente tesis se llegaron a las siguientes conclusiones:

1) En el capítulo II se comprobó que el modelo lineal incierto propuesto describe adecuadamente la dinámica lineal de un blanco aéreo común durante trayectos a rumbo constante y la dinámica no lineal de un blanco aéreo altamente maniobrable durante cambios de rumbo y maniobra terminal. Esto permitió trabajar con un solo modelo del blanco, a diferencia del algoritmo de modelo múltiple interactivo (IMM) que utiliza varios modelos del blanco en paralelo.
2) El vector de entrada control $\boldsymbol{u_k}$ al modelo, en unidades de sobre aceleración, es una consecuencia directa del movimiento de los mandos del piloto de la aeronave o las superficies de control del misil. Si bien es cierto el vector de entrada de control no puede ser medido, se demostró que este puede ser estimado mediante el uso de un diferenciador robusto exacto.
3) Las incertidumbres estructuradas $\boldsymbol{\xi_k}$ o perturbaciones al modelo, en unidades sobre aceleración, permitieron capturar la dinámica no lineal incierta del blanco durante los cambios de rumbo y maniobra terminal helicoidal.
4) El ruido angular fue modelado adecuadamente mediante una distribución gaussiana mixta, permitiendo evaluar la sensibilidad a este de los observadores diseñados y la capacidad de filtrado del diferenciador robusto exacto filtrado en relación a los filtros Kalman y filtro de partículas.
5) En el capítulo II se pudo constatar que los observadores de modos deslizantes clásico poseen una excelente exactitud y robustez ante las no linealidades del blanco, siendo bastante sensibles al ruido angular. Sin embargo, efectuando una síntesis del observador de Walcott-Zak por medio del algoritmo LQG se pudo atenuar de cierta forma el ruido angular y obtener muy buena robustez.
6) Los observadores basados en el algoritmo Super-Twisting de orden arbitrario permiten efectuar la estimación de las variables de estado de velocidad y aceleración solo mediante el uso de la posición, siendo esta una ventaja sobre los observadores clásicos si es que no se cuenta con un radar a pulsos Doppler. Por otro lado, estos observadores minimizan el castañeo o "chattering" pero reducen la velocidad de convergencia al origen del plano de estados debido a la super torsión que implementan. Por ello, para aplicaciones de radar es fundamental aplicar el algoritmo Super-Twisting con términos de orden alto, los cuales elevan considerablemente la velocidad de convergencia a un conjunto compacto indistintamente del error inicial entre la variable de estimación y la real.
7) Los diferenciadores mostraron ser de gran ayuda para poder obtener las derivadas de forma robusta y exacta de la posición hasta la cuarta derivada en la ausencia de ruido, permitiendo estimar el vector de entrada de control $\boldsymbol{\hat{u}_k}$. Sin embargo, es fundamental efectuar la adaptación de la constante de Lipschitz e introducir términos de super-twisting de orden alto para efectuar la estimación exacta y robusta de señales variantes en el tiempo, tales como las incertidumbres estructuradas propuestas del modelo. Por ello, en esta tesis se propuso un diferenciador robusto exacto adaptativo (ARED) en base a una función débil de Lyapunov y una estructura pseudo lineal del sistema de error que permite obtener un mejor desempeño.
8) El diferenciador robusto exacto de filtrado estándar muestra muy buena capacidad de filtrado del ruido angular pero pierde robustez a valores bajos de la constante Lipschitz. Asimismo, se observó que al aumentar los valores de la constante de Lipschitz, con la finalidad de aumentar la robustez y exactitud, se perdió capacidad de filtrado. Por ello, se

propuso un nuevo diferenciador robusto exacto y uniforme, el cual combina mediante una función de conmutación la capacidad de filtrado del diferenciador de filtrado estándar de Levant con la robustez de los términos de orden alto super-Twisting. Asimismo, para lograr mantener la filtración del ruido durante la conmutación a los términos de orden alto, se incluyó en el algoritmo un filtro de mediana de secuencia intra pulso, logrando obtener una muy buena relación de filtrado de ruido angular y robustez durante los cambios de rumbo y maniobra terminal del misil; mediante simulaciones se observó que la robustez y capacidad de filtrado del algoritmo propuesto es superior a la del filtro Kalman lineal, Kalman Cubature, Filtro de Partículas y Robust Student-t based Kalman Filter.

9) Finalizado el capítulo III se efectuó la comparación del desempeño de todos los observadores y diferenciadores diseñados, llegando a la conclusión de que en la ausencia de ruido el diferenciador robusto exacto adaptativo (ARED), el diferenciador robusto exacto y uniforme filtrado (UREDF) y los observadores clásicos muestran los mejores desempeños. Sin embargo, en un ambiente de ruido angular es necesario efectuar en primera instancia la filtración del ruido por medio del algoritmo UREDF, con la finalidad de poder utilizar el ARED y los observadores clásicos para la estimación de variables de estado.
10) En el capítulo IV se propusieron dos soluciones de modos deslizantes, comparando su desempeño respecto al algoritmo de modelo múltiple interactivo del toolbox de Sensor Fusion and Tracking en el capítulo V, obteniéndose resultados satisfactorios. Al respecto, se observó que el IMM no posee la velocidad de transición entre filtros suficiente para traquear al blanco de manera exacta y robusta durante la maniobra terminal del misil, siendo su desempeño inferior al de las soluciones de modos deslizantes propuestas. Asimismo, se pudo constatar por medio de simulaciones que el costo computacional de los algoritmos de modos deslizantes planteados es inferior al del IMM, siendo esto una ventaja para la implementación de las soluciones propuestas en tiempo real.
11) En el capítulo VI se pudo constatar que es factible la implementación de las soluciones propuestas en un sistema PXI para efectuar la estimación de variables de estado del blanco para un sistema de control de tiro MK.92 mod.2, el cual es similar a los sistemas de control de tiro que posee la Marina de Guerra del Perú. Sin embargo, queda pendiente contar con información más detalla del sistema de control de tiro a intervenir previamente a la ejecución de la propuesta de implementación del citado capítulo.

Recomendaciones

Para la presente tesis, a partir de las conclusiones, se deducen las siguientes recomendaciones:

1) Continuar con la investigación de la factibilidad de utilizar diferenciadores robustos exactos filtrados y uniformes con capacidades adaptativas para la estimación del vector de entrada de control al sistema, con la finalidad de utilizar un modelo lineal incierto del blanco.
2) Efectuar el modelamiento de las incertidumbres estructuradas $\boldsymbol{\xi_k}$ o perturbaciones al modelo con otro tipo de señales que no sean senoidales o cosenoidales, mas bien, que sean perturbaciones mas impulsivas, a fin de verificar el comportamiento de los observadores propuestos.
3) Efectuar la implementación del diferenciador robusto exacto filtrado y uniforme propuesto en un sistema PXI, a fin de efectuar la estimación de variables de estado en un sistema de control de tiro MK.92 mod.2, el cual es idéntico a los sistemas de control de tiro WM-28 que posee la Marina de Guerra del Perú.

Bibliografía

[1] M. Skolnik, *Radar handbook*. New York: McGraw-Hill, 2009.

[2] S. Sherman and D. Barton, *Monopulse principles and techniques*. Boston: Artech House, 2011.

[3] E. Mazor, A. Averbuch, Y. Bar-Shalom and J. Dayan, “Interacting multiple model methods in target tracking: a survey,” in IEEE Transactions on Aerospace and Electronic Systems, vol. 35, no. 1, pp. 103-123, Jan. 1998, doi: 10.1109/7.640267.

[4] R. Hampton and J. Cooke, "Unsupervised Tracking of Maneuvering Vehicles", *IEEE Transactions on Aerospace and Electronic Systems*, vol. -9, no. 2, pp. 197-207, 1973. Available: 10.1109/taes.1973.309767.

[5] W. Chang and S. Lin, "Incremental maneuver estimation model for target tracking", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 28, no. 2, pp. 439-452, 1992. Available: 10.1109/7.144570.

[6] R. Singer, R. Sea and K. Housewright, "Derivation and evaluation of improved tracking filter for use in dense multitarget environments", *IEEE Transactions on Information Theory*, vol. 20, no. 4, pp. 423-432, 1974. Available: 10.1109/tit.1974.1055256.

[7] R. Singer, "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets", *IEEE Transactions on Aerospace and Electronic Systems*, vol. -6, no. 4, pp. 473-483, 1970. Available: 10.1109/taes.1970.310128.

[8] J. Thorp, "Optimal Tracking of Maneuvering Targets", *IEEE Transactions on Aerospace and Electronic Systems*, vol. -9, no. 4, pp. 512-519, 1973. Available: 10.1109/taes.1973.309633 [Accessed 19 September 2020].

[9] H. SUN and S. CHIANG, "Manoeuvring target tracking algorithm for a radar system", *International Journal of Systems Science*, vol. 20, no. 10, pp. 1801-1811, 1989. Available: 10.1080/00207728908910266 [Accessed 19 September 2020].

[10] S. Lim and M. Farooq, "Maneuvering target tracking using jump processes", *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*. Available: 10.1109/cdc.1991.261779 [Accessed 19 September 2020].

[11] D. Sworder, M. Kent, R. Vojak and R. Hutchins, "Renewal models for maneuvering targets", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 1, pp. 138-150, 1995. Available: 10.1109/7.366300.

[12] Y. Chan, A. Hu and J. Plant, "A Kalman Filter Based Tracking Scheme with Input Estimation", *IEEE Transactions on Aerospace and Electronic Systems*, vol. -15, no. 2, pp. 237-244, 1979. Available: 10.1109/taes.1979.308710 [Accessed 19 September 2020].

[13] Y. Chan and F. Couture, "Manoeuvre detection and track correction by input estimation", *IEE Proceedings F Radar and Signal Processing*, vol. 140, no. 1, p. 21, 1993. Available: 10.1049/ip-f-2.1993.0003 [Accessed 19 September 2020].

[14] P. Bogler, "Tracking a Maneuvering Target Using Input Estimation", *IEEE Transactions on Aerospace and Electronic Systems*, vol. -23, no. 3, pp. 298-310, 1987. Available: 10.1109/taes.1987.310826.

[15] Kun Zhou, Xiqin Wang, M. Tomizuka, Wei-Bin Zhang and Ching-Yao Chan, "A new maneuvering target tracking algorithm with input estimation", *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*. Available: 10.1109/acc.2002.1024798 [Accessed 19 September 2020].

[16] Y. Bar-Shalom and K. Birmiwal, "Variable Dimension Filter for Maneuvering Target Tracking", *IEEE Transactions on Aerospace and Electronic Systems*, vol. -18, no. 5, pp. 621-629, 1982. Available: 10.1109/taes.1982.309274 [Accessed 19 September 2020].

[17] H. Khaloozadeh and A. Karsaz, "Modified input estimation technique for tracking manoeuvring targets", *IET Radar, Sonar & Navigation*, vol. 3, no. 1, p. 30, 2009. Available: 10.1049/iet-rsn:20080028.

[18] P. West and A. Haddsd, "Switched-Markov Filtering for Tracking Maneuvering Targets", *1991 American Control Conference*, 1991. Available: 10.23919/acc.1991.4791813 [Accessed 19 September 2020].

[19] A. Jaffer and S. Gupta, "Recursive Bayesian estimation with uncertain observation (Corresp.)", *IEEE Transactions on Information Theory*, vol. 17, no. 5, pp. 614-616, 1971. Available: 10.1109/tit.1971.1054684 [Accessed 19 September 2020].

[20] S. Zhang, J. Li, L. Wu and C. Shi, "A multiple maneuvering targets tracking algorithm based on a generalized pseudo-Bayesian estimator of first order", *Journal of Zhejiang University SCIENCE C*, vol. 14, no. 6, pp. 417-424, 2013. Available: 10.1631/jzus.c1200310.
[21] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients", *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780-783, 1988. Available: 10.1109/9.1299 [Accessed 19 September 2020].
[22] R. Liu and X. Chen, "Maneuvering target tracking algorithm based on fuzzy interacting multiple model", *JOURNAL OF ELECTRONIC MEASUREMENT AND INSTRUMENT*, vol. 26, no. 10, pp. 846-850, 2013. Available: 10.3724/sp.j.1187.2012.00846.
[23] C. Gao et al., "Maneuvering Target Tracking with Recurrent Neural Networks for Radar Application", *2018 International Conference on Radar (RADAR)*, 2018. Available: 10.1109/radar.2018.8557284 [Accessed 19 September 2020].
[24] C. Gao, J. Yan, S. Zhou, P. Varshney and H. Liu, "Long short-term memory-based deep recurrent neural networks for target tracking", *Information Sciences*, vol. 502, pp. 279-296, 2019. Available: 10.1016/j.ins.2019.06.039.
[25] Y. Yu and Q. Cheng, "Particle filters for maneuvering target tracking problem", *Signal Processing*, vol. 86, no. 1, pp. 195-203, 2006. Available: 10.1016/j.sigpro.2005.03.021.
[26] H. Tsaknakis and M. Athans, "Tracking maneuvering targets using H/sub ∞/ filters", *Proceedings of 1994 American Control Conference - ACC '94*. Available: 10.1109/acc.1994.752382 [Accessed 19 September 2020].
[27] K. Harikumar, T. Bera, R. Bardhan and S. Sundaram, "Discrete-time sliding mode observer for the state estimation of a manoeuvring target", *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 233, no. 7, pp. 847-854, 2019. Available: 10.1177/0959651819826488.
[28] Y. Bar-Shalom, K. Chang and H. Blom, "Tracking a maneuvering target using input estimation versus the interacting multiple model algorithm", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, no. 2, pp. 296-300, 1989. Available: 10.1109/7.18693.
[29] Zuo Dongguang, Han Chongzhao, Bian Shutan Zhenglin and Zhu Hongyan, "Tracking of maneuvering target in glint noise environment", Sixth International Conference of Information Fusion, 2003. Proceedings of the, 2003. Available: 10.1109/icif.2003.177401 [Accessed 27 July 2020].
[30] D. Patterson and R. Ashley, "Glint Tracking Errors in Radar", *Dynamic Modeling and Econometrics in Economics and Finance*, pp. 121-136, 2000. Available: 10.1007/978-1-4419-8688-7_7 [Accessed 16 March 2021].
[31] J. Kim, M. Tandale, P. Menon and E. Ohlmeyer, "Particle Filter for Ballistic Target Tracking with Glint Noise", *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 6, pp. 1918-1921, 2010. Available: 10.2514/1.51000.
[32] E. Daeipour and Y. Bar-Shalom, "An interacting multiple model approach for target tracking with glint noise", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 2, pp. 706-715, 1995. Available: 10.1109/7.381918.
[33] A. F. Genovese, "The interacting multiple model algorithm for accurate state estimation of maneuvering targets," Johns Hopkins APL technical digest, vol. 22, no. 4, pp. 614–623, 2001.
[34] X. Rong Li and V. Jilkov, "Survey of maneuvering target tracking. part I: dynamic models", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333-1364, 2003. Available: 10.1109/taes.2003.1261132.
[35] R. A. Singer. Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets. IEEE Trans. Aerospace and Electronic Systems, AES-6:473–483, July 1970.
[36] H. Zhou and K. S. P. Kumar. A "Current" Statistical Model and Adaptive Algorithm for Estimating Maneuvering Targets. AIAA Journal of Guidance, 7(5):596–602, Sept.-Oct. 1984.
[37] K. Mehrotra and P. R. Mahapatra. A Jerk Model to Tracking Highly Maneuvering Targets. IEEE Trans. Aerospace and Electronic Systems, AES-33(4):1094–1105, 1997.
[38] K. Ramachandra, *Kalman Filtering techniques for radar tracking. New York: Marcel Dekker, Inc., 2000.*
[39] X. Nie and F. Zhang, "Adaptive tracking algorithm based on 3D variable turn model", *Journal of Systems Engineering and Electronics,* vol.28, no.5, pp.281-860, 2017. Available: 10.21629/jsee.2017.05.04
[40] R. Brulle, *Engineering the space age*. [Place of publication not identified]: Bibliogov, 2012.
[41] G. Siouris, *Missile guidance and control systems*. New York: Springer, 2011.
[42] I. Shkolnikov, Y. Shtessel and D. Lianos, "Effect of sliding mode observers in the homing guidance loop", *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 219, no. 2, pp. 103-111, 2005. Available: 10.1243/095441005x30199.
[43] K. Ogata, Modern control engineering. [Delhi]: Pearson, 2016.
[44] D. Gu, P. Petkov and M. Konstantinov, *Robust Control Design with MATLAB®®*. London: Springer, 2013.
[45] X. Yan, S. Spurgeon and C. Edwards, Variable Structure Control of Complex Systems, 1st ed. Cham, Switzerland: Springer International Publishing, 2017, pp. 3-5.
[46] Y. Shtessel, C. Edwards, L. Fridman and A. Levant, *Sliding Mode Control and Observation*. New York: Springer, 2014.

[47] V. Utkin, A. Poznyak, Y. Orlov and A. Polyakov, "Conventional and high order sliding mode control", *Journal of the Franklin Institute*, vol. 357, no. 15, pp. 10244-10261, 2020. Available: 10.1016/j.jfranklin.2020.06.018 [Accessed 2 February 2021].
[48] J. Zhang, A. Swain and S. Nguang, *Robust Observer-Based Fault Diagnosis for Nonlinear Systems Using MATLAB®®, Advances in Industrial Control*. Springer International Publishing Switzerland, 2016.
[49] A. Andronov, S. Khaĭkin, S. Lefschetz, N. Goldowskaja and A. Vitt, *Theory of oscillations*. Princeton, NJ [etc.]: Princeton University Press, 1949.
[50] V. Utkin, *Sliding modes and their applications in variable structure systems*. Moscow: MIR, 1978.
[51] A. Sabanovic, L. Fridman and S. Spurgeon, *Variable Structure Systems*. Stevenage: IET, 2004.
[52] A. Filippov, *Differencial'nye uravnenija s razryvnoj pravoj čast'ju*. Moskva: Nauka, 1985.
[53] B.N. Pshenichny, *Convex Analysis and Extremal Problems (en Ruso).* Nauka, Moscú, 1980.
[54] J. Reger, Class Lecture, Topic: "Brief review of Lyapunov Theory" ICA632, Escuela de Posgrado - Maestría en Ingeniería de Control y Automatización, Pontifica Universidad Católica del Perú, Lima, Perú., Apr., 2020.
[55] J. P. LaSalle, An Invariance Principle in the Theory of Stability. New York: Academic, 1967.
[56] C. Byrnes and C. Martin, "An integral-invariance principle for nonlinear systems," IEEE Trans. Autom. Control, vol. 40, no. 6, pp. 983–994, Jun. 1995
[57] E. Ryan, "An integral invariance principle for differential inclusions with applications in adaptive control," SIAM J. Control Optim., vol. 36, no. 3, pp. 960–980, 1998.
[58] N. Fischer, R. Kamalapurkar and W. Dixon, "LaSalle-Yoshizawa Corollaries for Nonsmooth Systems", *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2333-2338, 2013. Available: 10.1109/tac.2013.2246900.
[59] V. V. Velichenko, *Invariancia de sistemas discontinuos*, Avtomat. i Telemekh., 1973, Issue 8, 155–160
[60] L. I. Rozonoer, "A variational approach to an invariance problem. I", *Avtomat. i Telemekh.*, **24**:6 (1963), 744–756
[61] C. Edwards and S. Spurgeon, "On the development of discontinuous observers", *International Journal of Control*, vol. 59, no. 5, pp. 1211-1229, 1994. Available: 10.1080/00207179408923128.
[62] S. Spurgeon, "Sliding mode observers: a survey", *International Journal of Systems Science*, vol. 39, no. 8, pp. 751-764, 2008. Available: 10.1080/00207720701847638.
[63] S. Spurgeon, Class Lecture, Topic: "Lecture 3: Introduction to Sliding Mode Observers – MATLAB® Design (SKS)" GIAN Course on Advanced Sliding Mode Control and Estimation for Real Complex Systems of the 21st Century, Indian Institute of Technology, Rorkee, India., Oct., 2017.
[64] J. Xiang, H. Su and J. Chu, "On the design of Walcott-Zak sliding mode observer", *Proceedings of the 2005, American Control Conference, 2005.*. Available: 10.1109/acc.2005.1470334 [Accessed 28 September 2020].
[65] E. Boukas, *Control of Singular Systems with Random Abrupt Changes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.
[66] Barbashin, E.A., Geraschenko, E.I.: On speeding up sliding modes in automatic control systems. Differentzialniye Uravneniya 1, 25–32 (1965).
[67] A. S. Huaman Loayza and C. G. Pérez Zuñiga, "Design of a fuzzy sliding mode controller for the autonomous path-following of a quadrotor, IEEE Latin America Transactions, vol. 17, no. 6, pp. 962, 971, Jun. 2019. DOI:10.1109/TLA. 2019.8896819.
[68] Y. Shtessel, C. Edwards, L. Fridman and A. Levant, *Sliding Mode Control and Observation*. New York: Springer, 2014.
[69] S. Emelyanov, *Binary control systems*. Moscow: International Research Institute for Management Sciences, 1985.
[70] Y. Shtessel, J. Moreno, F. Plestan, L. Fridman and A. Poznyak, "Super-twisting adaptive sliding mode control: A Lyapunov design", *49th IEEE Conference on Decision and Control (CDC)*, 2010. Available: 10.1109/cdc.2010.5717908 [Accessed 26 January 2021].
[71] A. Barth, M. Reichhartinger, J. Reger, M. Horn and K. Wulff, "Lyapunov-design for a super-twisting sliding-mode controller using the certainty-equivalence principle", *IFAC-PapersOnLine*, vol. 48, no. 11, pp. 860-865, 2015. Available: 10.1016/j.ifacol.2015.09.298.
[72] K. Young and U. Özgüner, *Variable structure systems, sliding mode and nonlinear control*, 1st ed. Springer, London.
[73] X. Yan, "Development of robust control based on sliding mode for nonlinear uncertain systems.", Ph. D., Ecole Centrale de Nantes (ECN). Automatic Control Engineering., 2016.
[74] LV. Levantovsky. Second order sliding algorithms: their realization. Dynamics of Heterogeneous Systems, pages 32–43, 1985
[75] A. Levant, "Robust exact differentiation via sliding mode technique", *Automatica*, vol. 34, no. 3, pp. 379-384, 1998. Available: 10.1016/s0005-1098(97)00209-4.
[76] A. Levant, "Higher-order sliding modes, differentiation and output-feedback control", *International Journal of Control*, vol. 76, no. 9-10, pp. 924-941, 2003. Available: 10.1080/0020717031000099029.
[77] S. Kamal, A. Chalanga, J. Moreno, L. Fridman and B. Bandyopadhyay, "Higher order super-twisting algorithm", *2014 13th International Workshop on Variable Structure Systems (VSS)*, 2014. Available: 10.1109/vss.2014.6881129 [Accessed 14 October 2020].

[78] A. Levant, "Sliding order and sliding accuracy in sliding mode control", *International Journal of Control*, vol. 58, no. 6, pp. 1247-1263, 1993. Available: 10.1080/00207179308923053.
[79] A. Chalanga, S. Kamal, L. Fridman, B. Bandyopadhyay and J. Moreno, "Implementation of Super-Twisting Control: Super-Twisting and Higher Order Sliding-Mode Observer-Based Approaches", *IEEE Transactions on Industrial Electronics*, vol. 63, no. 6, pp. 3677-3685, 2016. Available: 10.1109/tie.2016.2523913 [Accessed 26 January 2021].
[80] A. Levant and M. Livne, "Robust exact filtering differentiators", *European Journal of Control*, vol. 55, pp. 33-44, 2020. Available: 10.1016/j.ejcon.2019.08.006.
[81] M. Angulo, J. Moreno and L. Fridman, "Robust exact uniformly convergent arbitrary order differentiator", *Automatica*, vol. 49, no. 8, pp. 2489-2495, 2013. Available: 10.1016/j.automatica.2013.04.034.A. Bacciotti, L. Rosier, Liapunov Functions and Stability in Control Theory, Springer Verlag, London, 2005.
[82] A. Levant, "Homogeneity approach to high-order sliding mode design", *Automatica*, vol. 41, no. 5, pp. 823-830, 2005. Available: 10.1016/j.automatica.2004.11.029.
[83] Cruz–Zavala, E. and Moreno, J. A. and Fridman, L. M., 2010. Diferenciador Robusto Exacto y Uniforme. Proceedings of AMCA 2010, 1–6.
[84] E. Cruz-Zavala, J. Moreno and L. Fridman, "Uniform Robust Exact Differentiator", *IEEE Transactions on Automatic Control*, vol. 56, no. 11, pp. 2727-2733, 2011. Available: 10.1109/tac.2011.2160030.
[85] M. Mojallizadeh, B. Brogliato and V. Acary, "Discrete-time differentiators: design and comparative analysis", Posgraduate, University of Grenoble Alpes, 2021.
[86] M. Livne and A. Levant, "Proper discretization of homogeneous differentiators", *Automatica*, vol. 50, no. 8, pp. 2007-2014, 2014. Available: 10.1016/j.automatica.2014.05.028.
[87] F. Plestan, Y. Shtessel, V. Brégeault and A. Poznyak, "New methodologies for adaptive sliding mode control", International Journal of Control, vol. 83, no. 9, pp. 1907-1919, 2010. Available: 10.1080/00207179.2010.501385 [Accessed 19 February 2021].
[88] M. Reichhartinger and S. Spurgeon, "An arbitrary-order differentiator design paradigm with adaptive gains", *International Journal of Control*, vol. 91, no. 9, pp. 2028-2042, 2018. Available: 10.1080/00207179.2018.1429671.
[89] B. Kuo, Automatic control systems. Englewood Cliffs, NJ: Prentice-Hall, 1987.
[90] J. Moreno, in *51st IEEE Conference on Decision and Control*, Maui, Hawaii, USA, 2012.
[91] J. Moreno, "Levant's Arbitrary Order Differentiator with Varying Gain", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1705-1710, 2017. Available: 10.1016/j.ifacol.2017.08.496.
[92] E. Cruz-Zavala and J. Moreno, "Levant's Arbitrary-Order Exact Differentiator: A Lyapunov Approach", *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 3034-3039, 2019. Available: 10.1109/tac.2018.2874721 [Accessed 10 July 2021].
[93] Y. Shtessel, M. Taleb and F. Plestan, "A novel adaptive-gain supertwisting sliding mode controller: Methodology and application", *Automatica*, vol. 48, no. 5, pp. 759-769, 2012. Available: 10.1016/j.automatica.2012.02.024.
[94] L. Xie and Y. Soh, "Robust Kalman filtering for uncertain systems", *Systems & Control Letters*, vol. 22, no. 2, pp. 123-129, 1994. Available: 10.1016/0167-6911(94)90106-6.
[95] G. Agamennoni, J. Nieto and E. Nebot, "An outlier-robust Kalman filter", *2011 IEEE International Conference on Robotics and Automation*, 2011. Available: 10.1109/icra.2011.5979605 [Accessed 15 March 2021].
[96] J. Luo and P. Bosch, "Performance Robustness of Kalman Filters for Uncertain Linear Discrete-Time Systems", *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 3870-3875, 1996. Available: 10.1016/s1474-6670(17)58283-x.
[97] M. Zorzi, "Robust Kalman Filtering Under Model Perturbations", *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2902-2907, 2017. Available: 10.1109/tac.2016.2601879.
[98] C. Masreliez, "Approximate non-Gaussian filtering with linear state and observation relations", *IEEE Transactions on Automatic Control*, vol. 20, no. 1, pp. 107-110, 1975. Available: 10.1109/tac.1975.1100882.
[99] C. Masreliez and R. Martin, "Robust bayesian estimation for the linear model and robustifying the Kalman filter", *IEEE Transactions on Automatic Control*, vol. 22, no. 3, pp. 361-371, 1977. Available: 10.1109/tac.1977.1101538.
[100] W. Wu and A. Kunda, "Kalman filtering in non-Gaussian environment using efficient score function approximation", *IEEE International Symposium on Circuits and Systems*. Available: 10.1109/iscas.1989.100378.
[101] Weng-Rong Wu, "Target racking with glint noise", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 1, pp. 174-185, 1993. Available: 10.1109/7.249123.
[102] P. Pichlik and J. Zdenke, "Locomotive Wheel Slip Control Method Based on an Unscented Kalman Filter", *IEEE Transactions on Vehicular Technology*, pp. 1-1, 2018. Available: 10.1109/tvt.2018.2808379.
[103] E. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation", *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. Available: 10.1109/asspcc.2000.882463.
[104] S. Julier and J. Uhlmann, "New extension of the Kalman filter to nonlinear systems", *Signal Processing, Sensor Fusion, and Target Recognition VI*, 1997. Available: 10.1117/12.280797.

[105] I. Arasaratnam and S. Haykin, "Cubature Kalman Filters", *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254-1269, 2009. Available: 10.1109/tac.2009.2019800.
[106] P. Del Moral, "Nonlinear filtering: Interacting particle resolution", *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, vol. 325, no. 6, pp. 653-658, 1997. Available: 10.1016/s0764-4442(97)84778-7.
[107] J. Holland, *Adaptation in natural and artificial systems*. Cambridge, Mass: University of Michigan Press Ann Arbor, 1975.
[108] R. Cerf, *Une théorie asymptotique des algorithmes génétiques*. Montpellier, France: Université Montpellier II, Sciences et techniques du Langhedoc, 1994.
[109] H. Kunita, "Asymptotic behavior of the nonlinear filtering errors of Markov processes", *Journal of Multivariate Analysis*, vol. 1, no. 4, pp. 365-393, 1971. Available: 10.1016/0047-259x(71)90015-7.
[110] Ł. Stettner, "Invariant measures of the pair: state, approximate filtering process", *Colloquium Mathematicum*, vol. 62, no. 2, pp. 347-351, 1991. Available: 10.4064/cm-62-2-347-351.
[111] M. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking", *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174-188, 2002. Available: 10.1109/78.978374.
[112] F. Janabi-Sharifi, V. Hayward and C. Chen, "Discrete-time adaptive windowing for velocity estimation", *IEEE Transactions on Control Systems Technology*, vol. 8, no. 6, pp. 1003-1009, 2000. Available: 10.1109/87.880606.
[113] S. Jin, Y. Jin, X. Wang and X. Xiong, "Discrete-Time Sliding Mode Filter with Adaptive Gain", *Applied Sciences*, vol. 6, no. 12, p. 400, 2016. Available: 10.3390/app6120400.
[114] R. Kikuuwe, R. Pasaribu and G. Byun, "A First-Order Differentiator with First-Order Sliding Mode Filtering", *IFAC-PapersOnLine*, vol. 52, no. 16, pp. 771-776, 2019. Available: 10.1016/j.ifacol.2019.12.056.
[115] Wen-Rong Wu and Dah-Chung Chang, "Feedback median filter for robust preprocessing of glint noise", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 4, pp. 1026-1035, 2000. Available: 10.1109/7.892655.
[116] A. Bovik, T. Huang and D. Munson, "A generalization of median filtering using linear combinations of order statistics", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 31, no. 6, pp. 1342-1350, 1983. Available: 10.1109/tassp.1983.1164247.
[117] Aranda, I. and Pérez-Zuñiga, G., 2021. Highly maneuverable target tracking under glint noise via Uniform Robust Exact Filtering Differentiator with Intra Pulse Median Filter. *IEEE Transactions on Aerospace and Electronic Systems*, pp.1-1.
[118] Y. Huang, Y. Zhang, N. Li, Z. Wu and J. A. Chambers, “A Novel Robust Student's t-Based Kalman Filter,” in IEEE Transactions on Aerospace and Electronic Systems, vol. 53, no. 3, pp. 1545-1554, June 2017.
[119] K. Shade and M. Lucas, The MK-92 Fire control system: John's Hopkins Applied Physics Laboratory, 1981, pp. 69-73.
[120] National Instruments, *PXI Express NI PXIe-8880 User Manual*. Austin, Texas: National Instruments, 2015.
[121] National Instruments, *Device Specifications NI6375 X series Data Acquisition: 208 AI, 3.8 MS/s, 1 MS/s*, 16-bit resolution, $\pm 10V$, 24 DIO, 2 AO, Austin, Texas: National Instruments, 2015.
[122] National Instruments, *PXI Express NI PXIe-1071 User Manual*, Austin, Texas: National Instruments, 2013.

Apéndice I

Programas de Matlab del Capítulo II

Apéndice 1.1: Programa para la generación de vector de entrada de control u_k

```
function [uk] = gen_jerk_sea_skimming3(nt,load_factor,g,T)
%=========================================================================
%
%       Generación de entrada al modelo (jerk)
%       Modelo: Modelo lineal incierto
%       Entradas: constante gravitatoria "g"
%                 tiempo de muestreo T
%                 número de muestras nt
%                 factor de saturación de sobre aceleración load_factor
%       Salidas: Sobre aceleraciones en x,y,z
%
%
%       Italo Aranda Cetraro
%       Ingeniería de Control y Automatización
%       Pontificia Universidad Católica del Perú
%       Email: a20194480@pucp.edu.pe
%       14/06/2020
%
%=========================================================================
%-------------------------------------------------------------------------
% Función monopulso
%-------------------------------------------------------------------------
fc = 0.5;
fs = 50;
tc = gmonopuls('cutoff',fc);
sg = 1/(2*pi*fc);
%-------------------------------------------------------------------------
% Jerk coordenada x
%-------------------------------------------------------------------------
% Los vectores t0,t1,t2,... representan la duración del pulso de sobre
% aceleración
% Los vectores x0,x1,x2,... representan la amplitud del pulso de sobre
% aceleración
t0  = 0*T:1/fs:1999*T*tc/0.6366;              % 2000
t1  = 0*T*tc:1/fs:49*T*tc/0.6366;             % 50
t2  = 0*T:1/fs:49*T*tc/0.6366;                % 50
t3  = 0*T:1/fs:49*T*tc/0.6366;                % 50
t4  = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t5  = 0*T:1/fs:99*T*tc/0.6366;                % 100
t6  = 0*T:1/fs:59*T*tc/0.6366;                % 60
t7  = 0*T:1/fs:9*T*tc/0.6366;                 % 10
t8  = 0*T:1/fs:59*T*tc/0.6366;                % 60
t9  = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t10 = 0*T:1/fs:300*T*tc/0.6366;               % 300
t11 = 0*T:1/fs:3*T*tc/0.6366;                 % 4
t12 = 0*T:1/fs:49*T*tc/0.6366;                % 50
t13 = 0*T:1/fs:3*T*tc/0.6366;                 % 4
t14 = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t15 = 0*T:1/fs:883*T*tc/0.6366;               % 884
t16 = 0*T:1/fs:31*T*tc/0.6366;                % 32
t17 = 0*T:1/fs:49*T*tc/0.6366;                % 50
```

```
t18 = 0*T:1/fs:47*T*tc/0.6366;                % 48
t19 = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t20 = 0*T:1/fs:729*T*tc/0.6366;               % 730
t21 = 0*T:1/fs:59*T*tc/0.6366;                % 60
t22 = 0*T:1/fs:524*T*tc/0.6366;               % 525
t23 = 0*T:1/fs:54*T*tc/0.6366;                % 55

x0  = 0*exp(1/2)*t0/sg.*exp(-(t0/sg).^2/2);
x1  = 10*exp(1/2)*t1/sg.*exp(-(t1/sg).^2/2);
x2  = 0*exp(1/2)*t2/sg.*exp(-(t2/sg).^2/2);
x3  =-10*exp(1/2)*t3/sg.*exp(-(t3/sg).^2/2);
x4  = 0*exp(1/2)*t4/sg.*exp(-(t4/sg).^2/2);
x5  = 0*exp(1/2)*t5/sg.*exp(-(t5/sg).^2/2);
x6  = -9.2*exp(1/2)*t6/sg.*exp(-(t6/sg).^2/2);
x7  = 0*exp(1/2)*t7/sg.*exp(-(t7/sg).^2/2);
x8  = 9.2*exp(1/2)*t8/sg.*exp(-(t8/sg).^2/2);
x9  = 0*exp(1/2)*t9/sg.*exp(-(t9/sg).^2/2);
x10 = 0*exp(1/2)*t10/sg.*exp(-(t10/sg).^2/2);
x11 = -15.17*exp(1/2)*t11/sg.*exp(-(t11/sg).^2/2);
x12 = 0*exp(1/2)*t12/sg.*exp(-(t12/sg).^2/2);
x13 = 15.17*exp(1/2)*t13/sg.*exp(-(t13/sg).^2/2);
x14 = 0*exp(1/2)*t14/sg.*exp(-(t14/sg).^2/2);
x15 = 0*exp(1/2)*t15/sg.*exp(-(t15/sg).^2/2);
x16 = -6.5*exp(1/2)*t16/sg.*exp(-(t16/sg).^2/2);
x17 = 0*exp(1/2)*t17/sg.*exp(-(t17/sg).^2/2);
x18 = 7.83*exp(1/2)*t18/sg.*exp(-(t18/sg).^2/2);
x19 = 0*exp(1/2)*t19/sg.*exp(-(t19/sg).^2/2);
x20 = 0*exp(1/2)*t20/sg.*exp(-(t20/sg).^2/2);
x21 = 14.1*exp(1/2)*t21/sg.*exp(-(t21/sg).^2/2);
x22 = 0.0*exp(1/2)*t22/sg.*exp(-(t22/sg).^2/2);
x23 = 5.1*exp(1/2)*t23/sg.*exp(-(t23/sg).^2/2);

tx  = [x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x20 x21 x22
x23];
length(tx);
tx = [ tx zeros(1,nt-length(tx))];
%-------------------------------------------------------------------------
% Jerk coordenada y
%-------------------------------------------------------------------------
t0  = 0*T:1/fs:1999*T*tc/0.6366;              % 2000
t1  = 0*T:1/fs:19*T*tc/0.6366;                % 20
t2  = 0*T:1/fs:49*T*tc/0.6366;                % 50
t3  = 0*T:1/fs:19*T*tc/0.6366;                % 20
t4  = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t5  = 0*T:1/fs:99*T*tc/0.6366;                % 100
t6  = 0*T:1/fs:29*T*tc/0.6366;                % 30
t7  = 0*T:1/fs:149*T*tc/0.6366;               % 150
t8  = 0*T:1/fs:29*T*tc/0.6366;                % 30
t9  = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t10 = 0*T:1/fs:400*T*tc/0.6366;               % 400
t11 = 0*T:1/fs:39*T*tc/0.6366;                % 40
t12 = 0*T:1/fs:49*T*tc/0.6366;                % 50
t13 = 0*T:1/fs:39*T*tc/0.6366;                % 40
t14 = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t15 = 0*T:1/fs:527*T*tc/0.6366;               % 528
t16 = 0*T:1/fs:49*T*tc/0.6366;                 % 50
t17 = 0*T:1/fs:49*T*tc/0.6366;                % 50
t18 = 0*T:1/fs:49*T*tc/0.6366;                 % 50
t19 = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t20 = 0*T:1/fs:99*T*tc/0.6366;                % 100
t21 = 0*T:1/fs:57*T*tc/0.6366;                % 58
t22 = 0*T:1/fs:49*T*tc/0.6366;                % 50
t23 = 0*T:1/fs:57*T*tc/0.6366;                % 58
t24 = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t25 = 0*T:1/fs:770*T*tc/0.6366;               % 771
t26 = 0*T:1/fs:3*T*tc/0.6366;                 % 4
t27 = 0*T:1/fs:499*T*tc/0.6366;               % 500
t28 = 0*T:1/fs:9*T*tc/0.6366;                 % 10
t29 = 0*T:1/fs:9*T*tc/0.6366;                 % 10
t30 = 0*T:1/fs:9*T*tc/0.6366;                 % 10
t31 = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t32 = 0*T:1/fs:1*T*tc/0.6366;                 % 1
t33 = 0*T:1/fs:9*T*tc/0.6366;                 % 10
t34 = 0*T:1/fs:9*T*tc/0.6366;                 % 10
t35 = 0*T:1/fs:9*T*tc/0.6366;                 % 10
t36 = 0*T:1/fs:1*T*tc/0.6366;                 % 1
```

```
y0  =   0*exp(1/2)*t0/sg.*exp(-(t0/sg).^2/2);
y1  = -55*exp(1/2)*t1/sg.*exp(-(t1/sg).^2/2);
y2  =   0*exp(1/2)*t2/sg.*exp(-(t2/sg).^2/2);
y3  =  55*exp(1/2)*t3/sg.*exp(-(t3/sg).^2/2);
y4  =   0*exp(1/2)*t4/sg.*exp(-(t4/sg).^2/2);
y5  =   0*exp(1/2)*t5/sg.*exp(-(t5/sg).^2/2);
y6  =  32.5*exp(1/2)*t6/sg.*exp(-(t6/sg).^2/2);
y7  =   0*exp(1/2)*t7/sg.*exp(-(t7/sg).^2/2);
y8  = -32.5*exp(1/2)*t8/sg.*exp(-(t8/sg).^2/2);
y9  =   0*exp(1/2)*t9/sg.*exp(-(t9/sg).^2/2);
y10 =   0*exp(1/2)*t10/sg.*exp(-(t10/sg).^2/2);
y11 =-12*exp(1/2)*t11/sg.*exp(-(t11/sg).^2/2);
y12 =   0*exp(1/2)*t12/sg.*exp(-(t12/sg).^2/2);
y13 = 12*exp(1/2)*t13/sg.*exp(-(t13/sg).^2/2);
y14 =   0*exp(1/2)*t14/sg.*exp(-(t14/sg).^2/2);
y15 =   0*exp(1/2)*t15/sg.*exp(-(t15/sg).^2/2);
y16 = -10*exp(1/2)*t16/sg.*exp(-(t16/sg).^2/2);
y17 =   0*exp(1/2)*t17/sg.*exp(-(t17/sg).^2/2);
y18 =  10*exp(1/2)*t18/sg.*exp(-(t18/sg).^2/2);
y19 =   0*exp(1/2)*t19/sg.*exp(-(t19/sg).^2/2);
y20 =   0*exp(1/2)*t20/sg.*exp(-(t20/sg).^2/2);
y21 = -29*exp(1/2)*t21/sg.*exp(-(t21/sg).^2/2);
y22 =   0*exp(1/2)*t22/sg.*exp(-(t22/sg).^2/2);
y23 =  29*exp(1/2)*t23/sg.*exp(-(t23/sg).^2/2);
y24 =   0*exp(1/2)*t24/sg.*exp(-(t24/sg).^2/2);
y25 =   0*exp(1/2)*t25/sg.*exp(-(t25/sg).^2/2);
y26 = -34*exp(1/2)*t26/sg.*exp(-(t26/sg).^2/2);
y27 =   0*exp(1/2)*t27/sg.*exp(-(t27/sg).^2/2);
y28 = -55*exp(1/2)*t28/sg.*exp(-(t28/sg).^2/2);
y29 =   0*exp(1/2)*t29/sg.*exp(-(t29/sg).^2/2);
y30 =  55*exp(1/2)*t30/sg.*exp(-(t30/sg).^2/2);
y31 =   0*exp(1/2)*t31/sg.*exp(-(t31/sg).^2/2);
y32 =   0*exp(1/2)*t32/sg.*exp(-(t32/sg).^2/2);
y33 =  77*exp(1/2)*t33/sg.*exp(-(t33/sg).^2/2);
y34 =   0*exp(1/2)*t34/sg.*exp(-(t34/sg).^2/2);
y35 = -77*exp(1/2)*t35/sg.*exp(-(t35/sg).^2/2);
y36 =   0*exp(1/2)*t36/sg.*exp(-(t36/sg).^2/2);

ty  = [y0 y1 y2 y3 y4 y5 y6 y7 y8 y9 y10 y11 y12 y13 y14 y15 y16 y17 y18 y19 y20 y21 y22
y23...
       y24 y25 y26 y27 y28 y29 y30 y31 y32 y33 y34 y35 y36];
length(ty);
ty = [ ty zeros(1,nt-length(ty))];
%-------------------------------------------------------------------------
% Jerk coordenada z
%-------------------------------------------------------------------------
t0  = 0*T:1/fs:499*T*tc/0.6366;                 % 500
t1  = 0*T:1/fs:49*T*tc/0.6366;                  % 50
t2  = 0*T:1/fs:1*T*tc/0.6366;                   % 1
t3  = 0*T:1/fs:49*T*tc/0.6366;                  % 50
t4  = 0*T:1/fs:1*T*tc/0.6366;                   % 1
t5  = 0*T:1/fs:49*T*tc/0.6366;                  % 50
t6  = 0*T:1/fs:1*T*tc/0.6366;                   % 1
t7  = 0*T:1/fs:49*T*tc/0.6366;                  % 50
t8  = 0*T:1/fs:1*T*tc/0.6366;                   % 1
t9  = 0*T:1/fs:249*T*tc/0.6366;                 % 250
t10  = 0*T:1/fs:41*T*tc/0.6366;                 % 42
t11  = 0*T:1/fs:1*T*tc/0.6366;                  % 1
t12  = 0*T:1/fs:41*T*tc/0.6366;                 % 42
t13  = 0*T:1/fs:1*T*tc/0.6366;                  % 1
t14  = 0*T:1/fs:41*T*tc/0.6366;                 % 42
t15  = 0*T:1/fs:1*T*tc/0.6366;                  % 1
t16  = 0*T:1/fs:41*T*tc/0.6366;                 % 42
t17  = 0*T:1/fs:1*T*tc/0.6366;                  % 1
t18  = 0*T:1/fs:3245*T*tc/0.6366;               % 3246
t19  = 0*T:1/fs:29*T*tc/0.6366;                 % 30
t20  = 0*T:1/fs:1*T*tc/0.6366;                  % 1
t21  = 0*T:1/fs:29*T*tc/0.6366;                 % 30
t22  = 0*T:1/fs:1*T*tc/0.6366;                  % 1
t23  = 0*T:1/fs:29*T*tc/0.6366;                 % 30
t24  = 0*T:1/fs:1*T*tc/0.6366;                  % 1
t25  = 0*T:1/fs:29*T*tc/0.6366;                 % 30
t26  = 0*T:1/fs:1*T*tc/0.6366;                  % 1
t27  = 0*T:1/fs:614*T*tc/0.6366;                % 613
t28  = 0*T:1/fs:94*T*tc/0.6366;                 % 48

z0  =   0*exp(1/2)*t0/sg.*exp(-(t0/sg).^2/2);
```

```
z1  =   1*exp(1/2)*t1/sg.*exp(-(t1/sg).^2/2);
z2  =   0*exp(1/2)*t2/sg.*exp(-(t2/sg).^2/2);
z3  =  -1*exp(1/2)*t3/sg.*exp(-(t3/sg).^2/2);
z4  =   0*exp(1/2)*t4/sg.*exp(-(t4/sg).^2/2);
z5  =  -1*exp(1/2)*t5/sg.*exp(-(t5/sg).^2/2);
z6  =   0*exp(1/2)*t6/sg.*exp(-(t6/sg).^2/2);
z7  =   1*exp(1/2)*t7/sg.*exp(-(t7/sg).^2/2);
z8  =   0*exp(1/2)*t8/sg.*exp(-(t8/sg).^2/2);
z9  =   0*exp(1/2)*t9/sg.*exp(-(t9/sg).^2/2);

z10 =   -1.97*exp(1/2)*t10/sg.*exp(-(t10/sg).^2/2);
z11 =       0*exp(1/2)*t11/sg.*exp(-(t11/sg).^2/2);
z12 =    1.97*exp(1/2)*t12/sg.*exp(-(t12/sg).^2/2);
z13 =       0*exp(1/2)*t13/sg.*exp(-(t13/sg).^2/2);
z14 =    1.97*exp(1/2)*t14/sg.*exp(-(t14/sg).^2/2);
z15 =       0*exp(1/2)*t15/sg.*exp(-(t15/sg).^2/2);
z16 =   -1.97*exp(1/2)*t16/sg.*exp(-(t16/sg).^2/2);
z17 =       0*exp(1/2)*t17/sg.*exp(-(t17/sg).^2/2);

z18 =   0*exp(1/2)*t18/sg.*exp(-(t18/sg).^2/2);
z19 =   2.6*exp(1/2)*t19/sg.*exp(-(t19/sg).^2/2);
z20 =   0*exp(1/2)*t20/sg.*exp(-(t20/sg).^2/2);
z21 =  -2.6*exp(1/2)*t21/sg.*exp(-(t21/sg).^2/2);
z22 =   0*exp(1/2)*t22/sg.*exp(-(t22/sg).^2/2);
z23 =  -2.6*exp(1/2)*t23/sg.*exp(-(t23/sg).^2/2);
z24 =   0*exp(1/2)*t24/sg.*exp(-(t24/sg).^2/2);
z25 =   2.6*exp(1/2)*t25/sg.*exp(-(t25/sg).^2/2);
z26 =   0*exp(1/2)*t26/sg.*exp(-(t26/sg).^2/2);
z27 =   0.002*exp(1/2)*t27/sg.*exp(-(t27/sg).^2/2);
z28 =   4.1*exp(1/2)*t28/sg.*exp(-(t28/sg).^2/2);

tz  = [z0 z1 z2 z3 z4 z5 z6 z7 z8 z9 z10 z11 z12 z13 z14 z15 z16 z17 z18 z19 z20 z21 z22
z23...
       z24 z25 z26 z27 z28];
length(tz);
tz = [ tz zeros(1,nt-length(tz))];

jkx  = tx'.*g;  % m/s^3
jky  = ty'.*g;  % m/s^3
jkz  = tz'.*g;  % m/s^3
uk   = [jkx jky jkz]';
%-------------------------------------------------------------------------
% saturación de sobre aceleraciones
%-------------------------------------------------------------------------
for k = 1:nt
    for i = 1:3
        if (uk(i,k) > load_factor)
            uk(i,k) = load_factor;
        end
        if (uk(i,k) < -load_factor)
            uk(i,k) = -load_factor;
        end
    end
end
end
```

Apéndice 1.2: Programa para la Generación del vector de perturbaciones ξ_k

```
function [pk] = perturb_1(g,ti,T,tff)
%=====================================================================
%       Generación de perturbaciones del modelo
%       Modelo: Modelo lineal incierto
%       Entradas: Vector de tiempo y tiempo de muestreo de simulación
%       Salidas: Perturbaciones en x,y,z
%
%
%       Italo Aranda Cetraro
%       Ingeniería de Control y Automatización
%       Pontificia Universidad Católica del Perú
%       Email: a20194480@pucp.edu.pe
%       14/06/2020
%
%=====================================================================
% Perturbación del sistema (No linealidades)
fb   = 1.67;           % frecuencia base (Hz)
a    = 0.3*g;          % factor de amplitud de perturbación en [g]
```

```
b    = 4.5*g;          % factor de amplitud de perturbación en [g]
c    = 5.4*g;          % factor de amplitud de perturbación en [g]
k    = 1;
for tt = ti:T:tff
        if ((k >= 4650) && (k < 5167))
            perturb = [  b*sin(2*pi*fb*tt/(2*c)).*cos(2*pi*fb*tt);
                         cos(2*pi*16*fb*tt/(2*c));
                         a*sin(2*pi*fb*tt/(2*c)).*sin(2*pi*fb*tt)];%
        end
        if ((k >= 3900) && (k < 4000))
            perturb = [  -0.012*b;
                         0;
                         0];%
        end
        if ((k >= 4000) && (k < 4354))
            perturb = [  0.3*b*sin(2*pi*100*fb*tt/(2*c));
                         0;
                         0];%
        end
        if (((k < 1160) || ((k >= 1477) && (k < 3900)) || ((k >= 4354) && (k < 4650)) ||
(k >= 5167)))
            perturb = [0;
                       0;
                       0];
        end
        if ((k >= 1160) && (k < 1260))
            perturb = [  -0.0135*b;
                         0;
                         0];%
        end
        if ((k >= 1260) && (k < 1477))
            perturb = [  0.25*b*sin(2*pi*50*fb*tt/(2*c));
                         0;
                         0];%
        end
        pk(:,k) = perturb;
        k = k + 1;
end
end
```

Apéndice 1.3: Programa para la generación de modelo mixto gaussiano (GMM)

```
%=========================================================================
%
% Título               : glint_noise.m
% Propósito          : Generar ruido de tipo glint para radares
%
% Descripción      : Este programa permite la generación de ruido de medición de tipo
glint.
%
% Fecha de creación: 05/06/2020
% Autor                : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución        : Pontificia Universidad Católica del Perú
% Programa         : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
clear all
close all
clc
%-------------------------------------------------------------------------------------------
% Constantes
%-------------------------------------------------------------------------------------------
nt = 5226;        % número de puntos
nt_set  = 52;    % secuencia con 52 puntos
nt_set2 = 26;   % secuencia con 26 puntos
seg     = 101;    % cantidad de secuencias
%-------------------------------------------------------------------------------------------
% Modelo mixto Gaussiano de posición: alturas mayores a 5 mts
%-------------------------------------------------------------------------------------------
% Variables:    distancia, ruido angular acimut, ruido angular elevación
% #Filas:          Número de distribuciones a mezclar (2 distribuciones)
% #Columnas: Mean value de cada variable
%
% Matriz de valores promedio (Mean values)
```

```
%
%        mts mrad mrad
mu12 = [0.10 0.01 0.01;  % probabilidad 1
              0.05 0.02 0.03]; % probabilidad 2
%
% Proporción de la mezcla
%
p12     = [0.95 0.05];
%-------------------------------------------------------------------------------------------
% Modelo mixto Gaussiano de posición: alturas menores a 5 mts
%-------------------------------------------------------------------------------------------
%
%       mts mrad mrad
mu34 = [0.20 0.02 0.02;  % probabilidad 3
              0.10 0.04 0.06]; % probabilidad 4
% Proporción de la mezcla
p34     = [0.95 0.05];
%-------------------------------------------------------------------------------------------
% Modelo mixto Gaussiano de velocidad: alturas mayores a 5 mts
%-------------------------------------------------------------------------------------------
% Variables: velocidad doppler
% #Filas:      Número de distribuciones a mezclar (2 distribuciones)
% #Columnas: Mean value de cada variable
%
% Matriz de valores promedio (Mean values)
%
%          mts/s
mu56 = [0.02;  % probabilidad 1
             0.10]; % probabilidad 2
%
% Proporción de la mezcla
%
p56     = [0.95 0.05];
%-------------------------------------------------------------------------------------------
% Modelo mixto Gaussiano de velocidad: alturas menores a 5 mts
%-------------------------------------------------------------------------------------------
% Variables: velocidad doppler
% #Filas:       Número de distribuciones a mezclar (2 distribuciones)
% #Columnas: Mean value de cada variable
%
% Matriz de valores promedio (Mean values)
%
%            mts/s
mu78 = [0.03;  % probabilidad 1
              0.11]; % probabilidad 2
%
% Proporción de la mezcla
%
p78     = [0.9 0.1];

for k = 1:seg
%
% Matriz de covarianzas del ruido del modelo mixto gaussiano: alturas
% mayores a 5mts
%
%                            covarianza prob.1                    covarianza prob. 2
%                            mts2 mrad2 mrad2                  mts2 mrad2 mrad2
sigma12 = cat(3,[   0.02*k      0.00       0.00;
                              0.00  4.9505e-03*k     0.00;
                              0.00        0.00 9.9010e-03*k],[   0.297*k  0.00
0.00;
                                                                                    0.00
0.035*k   0.00;
                                                                                    0.00
0.00   0.045*k]);
%
% Crear modelo
%
gm12    = gmdistribution(mu12,sigma12,p12);
%
% Crear datos
%
    if k==seg
       Xgm12  = random(gm12,nt_set2);
    else
       Xgm12  = random(gm12,nt_set);
    end
```

```
% Matriz de covarianzas del ruido del modelo mixto gaussiano: alturas
% menores a 5mts
%               covarianza prob.1                   covarianza prob. 2
%               mts2 mrad2 mrad2                 mts2 mrad2 mrad2
sigma34 = cat(3,[  0.05*k  0.00 0.00;
                            0.00    0.010*k 0.00;
                            0.00    0.00 0.020*k],[   0.694*k   0.00    0.00;
                                                      0.00    0.044*k   0.00;
                                                      0.00    0.00
0.049*k]);
% Crear modelo
gm34    = gmdistribution(mu34,sigma34,p34);
% Crear datos
    if k==seg
       Xgm34  = random(gm34,nt_set2);
    else
       Xgm34  = random(gm34,nt_set);
    end
% Matriz de covarianzas del ruido del modelo mixto gaussiano (velocidad):
% alturas menores a 5mts
%
%                          varianza prob.1      varianza prob. 2
%                               mts2/s2              mts2/s2
sigma56 = cat(3,       0.010*k ,                0.035*k);
%
% Crear modelo
%
gm56    = gmdistribution(mu56,sigma56,p56);
%
% Crear datos
%
    if k==seg
       Xgm56  = random(gm56,nt_set2);
    else
       Xgm56  = random(gm56,nt_set);
    end
%
% Matriz de covarianzas del ruido
%
%                        varianza prob.1      varianza prob. 2
%                             mts2/s2              mts2/s2
sigma78 = cat(3,       0.015*k ,                0.045*k);
%
% Crear modelo
%
gm78    = gmdistribution(mu78,sigma78,p78);
%
% Crear datos
%
    if k==seg
       Xgm78  = random(gm78,nt_set2);
    else
       Xgm78  = random(gm78,nt_set);
    end
%
% Concatenar números de distribución mixta anteriores con nuevos
%
if k > 1
   Xgm12t = [Xgm12old Xgm12']; Xgm34t = [Xgm34old Xgm34'];
   Xgm56t = [Xgm56old Xgm56']; Xgm78t = [Xgm78old Xgm78'];
else
   Xgm12t = Xgm12'; Xgm34t = Xgm34'; Xgm56t = Xgm56'; Xgm78t = Xgm78';
end
% Guardar última distribución
Xgm12old = Xgm12t; Xgm34old = Xgm34t; Xgm56old = Xgm56t; Xgm78old = Xgm78t;
end
% Gráficos de modelos de posición
figure(1);qqplot(Xgm12t(3,:));
grid on
xlabel('Cuantiles normales estándar','FontSize',22)
ylabel('Datos ordenados','FontSize',22)
title('QQ-plot Distribución Mixta Gaussiana: Caso 1','FontSize',22)
legend({'Ad'},'Location','north','NumColumns',2,'FontSize',22)
figure(2);qqplot(Xgm34t(3,:));
grid on
xlabel('Cuantiles normales estándar','FontSize',22)
ylabel('Datos ordenados','FontSize',22)
```

```
title('QQ-plot Distribución Mixta Gaussiana: Caso 2','FontSize',22)
legend({'Ad'},'Location','north','NumColumns',2,'FontSize',22)
figure(3);qqplot(Xgm12t(2,:));
grid on
xlabel('Cuantiles normales estándar','FontSize',22)
ylabel('Datos ordenados','FontSize',22)
title('QQ-plot Distribución Mixta Gaussiana: Caso 1','FontSize',22)
legend({'Bdn'},'Location','north','NumColumns',2,'FontSize',22)
figure(4);qqplot(Xgm34t(2,:));
grid on
xlabel('Cuantiles normales estándar','FontSize',22)
ylabel('Datos ordenados','FontSize',22)
title('QQ-plot Distribución Mixta Gaussiana: Caso 2','FontSize',22)
legend({'Bdn'},'Location','north','NumColumns',2,'FontSize',22)
figure(5);qqplot(Xgm12t(1,:));
grid on
xlabel('Cuantiles normales estándar','FontSize',22)
ylabel('Datos ordenados','FontSize',22)
title('QQ-plot Distribución Mixta Gaussiana: Caso 1','FontSize',22)
legend({'Ed'},'Location','north','NumColumns',2,'FontSize',22)
figure(6);qqplot(Xgm34t(1,:));
grid on
xlabel('Cuantiles normales estándar','FontSize',22)
ylabel('Datos ordenados','FontSize',22)
title('QQ-plot Distribución Mixta Gaussiana: Caso 2','FontSize',22)
legend({'Ed'},'Location','north','NumColumns',2,'FontSize',22)
figure(7);qqplot(Xgm56t(1,:));
grid on
xlabel('Cuantiles normales estándar','FontSize',22)
ylabel('Datos ordenados','FontSize',22)
title('QQ-plot Distribución Mixta Gaussiana: Caso 1','FontSize',22)
legend({'Vm'},'Location','north','NumColumns',2,'FontSize',22)
figure(8);qqplot(Xgm78t(1,:));
grid on
xlabel('Cuantiles normales estándar','FontSize',22)
ylabel('Datos ordenados','FontSize',22)
title('QQ-plot Distribución Mixta Gaussiana: Caso 2','FontSize',22)
legend({'Vm'},'Location','north','NumColumns',2,'FontSize',22)
figure(9);plot(Xgm12t');hold on;
grid on
legend({'Ad','Bdn','Ed'},'Location','north','NumColumns',3,'FontSize',22)
axis([1 nt -20 20])
xlabel('muestras','FontSize',22)
ylabel('Ruido en [m],[mrad],[mrad]','FontSize',22)
title('Ruido vs muestras de Distribución Mixta Gaussiana: Caso 1','FontSize',22)
figure(10);plot(Xgm34t');hold on;
grid on
legend({'Ad','Bdn','Ed'},'Location','north','NumColumns',3,'FontSize',22)
axis([1 nt -20 20])
xlabel('muestras','FontSize',22)
ylabel('Ruido en [m],[mrad],[mrad]','FontSize',22)
title('Ruido vs muestras de Distribución Mixta Gaussiana: Caso 2','FontSize',22)
figure(12);plot(Xgm56t');hold on;
grid on
legend({'Vm'},'Location','north','NumColumns',1,'FontSize',22)
axis([1 nt -20 20])
xlabel('muestras','FontSize',22)
ylabel('Ruido en [m/s]','FontSize',22)
title('Ruido vs muestras de Distribución Mixta Gaussiana: Caso 1','FontSize',22)
figure(13);plot(Xgm78t');hold on;
grid on
legend({'Vm'},'Location','north','NumColumns',1,'FontSize',22)
axis([1 nt -20 20])
xlabel('muestras','FontSize',22)
ylabel('Ruido en [m/s]','FontSize',22)
title('Ruido vs muestras de Distribución Mixta Gaussiana: Caso 2','FontSize',22)
save glint_puntos.mat
```

Apéndice 1.4: Programa para la simulación de la trayectoria del blanco

```
%=========================================================================
%
% Título            : sim_modelo.m
% Propósito         : Simulación del modelo lineal incierto de un blanco
%                     aéreo de alta maniobrabilidad
%
```

```
% Descripción       : Este programa permite la generación de una
%                     trayectoria tridimensional de un blanco aéreo de alta
%                     maniobrabilidad (misil).
%
% Fecha de creación : 01/07/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
clear all
close all
clc
%-------------------------------------------------------------------------
% Parámetros iniciales de radar (sensor)
%-------------------------------------------------------------------------
T     = 1/50;                       % Período de muestreo (seg)
%-------------------------------------------------------------------------
% Definición de constantes físicas
%-------------------------------------------------------------------------
g     = 9.81;                       % gravedad (m/s2)
M     = 343;                        % velocidad del sonido (m/s)
load_factor = 100*g;                % máxima sobre aceleración (g/s)
%-------------------------------------------------------------------------
% Definición de vectores de tiempo
%-------------------------------------------------------------------------
ti    = 0;                          % Tiempo inicial de simulación
tff   = 104.5;                      % Tiempo final de simulación
tt    = ti:T:tff;                   % Vector de tiempo
tt    = tt';                        % Transpuesta del vector de tiempo
nt    = length(tt);                 % Número de muestras
%-------------------------------------------------------------------------
% Definición de posición y velocidad inicial del blanco en
% coordenadas esféricas
%-------------------------------------------------------------------------
Ad_init  = 35000;                         % Distancia al blanco (m)
Ed_init  = 0.01;                          % Elevación al blanco (°)
Bdn_init = 30;                            % Marcación al blanco (°)
Vm_init  = 0.9*M;                         % Velocidad del blanco (1 Mach = 343 m/s)
Rv_init  = 210;                           % Rumbo inicial del blanco (°)
%-------------------------------------------------------------------------
% Conversión de coordenadas esféricas a cartesianas (North-Sky-East) de la
% posición y velocidad inicial del blanco
%-------------------------------------------------------------------------
[Adx_init, Ady_init, Adz_init] = p2c(Ad_init,Ed_init,Bdn_init);
[Vmx_init, Vmy_init, Vmz_init] = p2c(Vm_init,0,Rv_init);
%-------------------------------------------------------------------------
% Inicialización de vector de estados
%-------------------------------------------------------------------------
xk = [Adx_init Vmx_init 0*g Ady_init Vmy_init 0*g Adz_init Vmz_init 0*g]';
%-------------------------------------------------------------------------
% Definición de matrices del modelo de espacio de estados del sistema
%-------------------------------------------------------------------------
% Matriz de medición (Pulse-Doppler radar)
Ck     = [1 0 0 0 0 0 0 0 0;        % pos x
          0 1 0 0 0 0 0 0 0;        % vel x
          0 0 0 1 0 0 0 0 0;        % pos y
          0 0 0 0 1 0 0 0 0;        % vel y
          0 0 0 0 0 0 1 0 0;        % pos z
          0 0 0 0 0 0 0 1 0];       % vel z

% Cálculo de matriz de transición Fk
Fk = [1 T (T^2)/2 0 0       0 0 0       0;
      0 1       T 0 0       0 0 0       0;
      0 0       1 0 0       0 0 0       0;
      0 0       0 1 T (T^2)/2 0 0       0;
      0 0       0 0 1       T 0 0       0;
      0 0       0 0 0       1 0 0       0;
      0 0       0 0 0       0 1 T (T^2)/2;
      0 0       0 0 0       0 0 1       T;
      0 0       0 0 0       0 0 0       1];

% Cálculo de la matriz de entrada Gk
Gk = [(T^3)/6       0        0;
      (T^2)/2       0        0;
```

```
        T            0        0;
        0       (T^3)/6       0;
        0       (T^2)/2       0;
        0            T        0;
        0            0   (T^3)/6;
        0            0   (T^2)/2;
        0            0        T];

% Matriz de distribución de perturbaciones (aceleraciones)
Dk = [75e4*(T^3)/6       0                0;
      5e3*(T^2)/2        0                0;
       1e2*T             0                0;
       0       75e4*(T^3)/6                0;
       0        5e3*(T^2)/2                0;
       0             1e2*T                 0;
       0                 0     75e4*(T^3)/6;
       0                 0      5e3*(T^2)/2;
       0                 0           1e2*T];
%-------------------------------------------------------------------------
% Generación del vector de sobre aceleraciones de entrada uk
%-------------------------------------------------------------------------
uk = gen_jerk_sea_skimming3(nt,load_factor,g,T);
%-------------------------------------------------------------------------
% Generación del vector de perturbaciones ?k
%-------------------------------------------------------------------------
pk = perturb_1(g,ti,T,tff);
%-------------------------------------------------------------------------
% Cargar ruido glint
%-------------------------------------------------------------------------
load glint_puntos.mat
%-------------------------------------------------------------------------
% Seleccionar simulación con ruido o sin ruido
%-------------------------------------------------------------------------
disp('Simulación de la trayectoria del blanco aéreo de alta maniobrabilidad')
disp('Modelo lineal incierto')
prompt = 'Para simular con ruido presione "1". Para simular sin ruido presione "0": ';
gl     = input(prompt);
%-------------------------------------------------------------------------
% Simulación del modelo
%-------------------------------------------------------------------------
for k = 1:nt
    % Modelo dinámico o de transición de estados
    xk       = Fk*xk + Gk*uk(:,k) + Dk*pk(:,k);
    % Simulación de adquisición de datos por el radar
    % Datos de posición adquiridos en coordenadas esféricas
    [Ad,Bdn,Ed] = c2p(xk);
    % Dato de velocidad adquirido por doppler (si cuenta con capacidad)
    Vm = sqrt(xk(2,1)^2 + xk(5,1)^2 + xk(8,1)^2);
    % Modelo de medición polar a cartesiano
    yk = yk_noise(Ad,Bdn,Ed,Vm,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,nt,gl,k);
    % Guardar señales
    bin_log(:,k)      = k;
    xk_log(:,k)       = xk;
    yk_log(:,k)       = yk;
end
%-------------------------------------------------------------------------
% Conversión de vector de estados reales a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
% 0.10197 - de mts/seg2 a g's
conv_xk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197]';
xk_log = xk_log.*conv_xk;
%-------------------------------------------------------------------------
% Conversión de vector de estados medidos a Mn, Mach
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
conv_yk   = [ 0.00054 0.00292 0.00054 0.00292 0.00054 0.00292 ]';
yk_log    = yk_log.*conv_yk;
%-------------------------------------------------------------------------
% Conversión de vector de entrada de control de m/s3 a g/s
%-------------------------------------------------------------------------
uk_log    = uk./g;
%-------------------------------------------------------------------------
% Conversión de vector de perturbaciones de m/s3 a g/s
%-------------------------------------------------------------------------
```

```
pk_log    = pk./g;
%--------------------------------------------------------------------------
% Plots
%--------------------------------------------------------------------------
%% 3D trayectoria completa
figure(1);p1 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-g',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-r',...
                                 0,                0,                0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p1(2),'LineWidth',1.5); set(p1(3),'LineWidth',1.5);
set(p1(4),'LineWidth',1.5); set(p1(5),'LineWidth',1.5);
set(p1(6),'LineWidth',1.5);
axis([-5 20 -5 10 0 0.04]);          % Toda la trayectoria
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Trayectoria del misil','FontSize',16)
legend({'mediciones','real','Posición del buque observador','Posición inicial del
misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
%% 3D trayectoria completa
figure(2);p2 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-g',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-r',...
                                 0,                0,                0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p2(2),'LineWidth',1.5); set(p2(3),'LineWidth',1.5);
set(p2(4),'LineWidth',1.5); set(p2(5),'LineWidth',1.5);
set(p2(6),'LineWidth',1.5);
axis([4 8 -2 5 0 0.016]);
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Maniobra terminal del misil','FontSize',16)
legend({'mediciones','real','Posición del buque observador','Posición inicial del
misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
%% Coordenada x
% Posición x: Trayectoria completa
figure(3);subplot(1,2,1);
p3 = plot(bin_log(1,:),xk_log(1,:),'-g',bin_log(1,:),yk_log(1,:),'-r');
title('Trayectoria completa del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición x en [mn]','FontSize',22)
legend({'real','medición'},'Location','north','NumColumns',2,'FontSize',18)
set(p3(1),'LineWidth', 2.0);
grid on
axis([1 nt min([xk_log(1,:) yk_log(1,:)]) max([xk_log(1,:) yk_log(1,:)])])
% Posición x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p4 = plot(bin_log(1,lpx1:nt),xk_log(1,lpx1:nt),'-
g',bin_log(1,lpx1:nt),yk_log(1,lpx1:nt),'-r');
title('Maniobra terminal del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición x en [mn]','FontSize',22)
legend({'real','medición'},'Location','north','NumColumns',2,'FontSize',18)
set(p4(1),'LineWidth', 2.0);
grid on
axis([lpx1 nt min([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)]) max([xk_log(1,lpx1:nt)
yk_log(1,lpx1:nt)])])
% Velocidad x: Trayectoria completa
figure(4);subplot(1,2,1);
p5 = plot(bin_log(1,:),xk_log(2,:),'-g',bin_log(1,:),yk_log(2,:),'-r');
title('Trayectoria completa del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad x en [mach]','FontSize',22)
legend({'real','medición'},'Location','north','NumColumns',2,'FontSize',18)
```

```
set(p5(1),'LineWidth', 2.0);
grid on
axis([1 nt min([xk_log(2,:) yk_log(2,:)]) max([xk_log(2,:) yk_log(2,:)])])
% Velocidad x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p6 = plot(bin_log(1,lpx1:nt),xk_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),yk_log(2,lpx1:nt),'-r');
title('Maniobra terminal del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad x en [mach]','FontSize',22)
legend({'real','medición'},'Location','north','NumColumns',2,'FontSize',18)
set(p6(1),'LineWidth', 2.0);
grid on
axis([lpx1 nt min([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt)]) max([xk_log(2,lpx1:nt)
yk_log(2,lpx1:nt)])])
% Aceleración x: Trayectoria completa
figure(5);subplot(1,2,1)
p7 = plot(bin_log(1,:),xk_log(3,:),'-g');
title('Trayectoria completa del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración x en [g]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p7(1),'LineWidth', 2.0);
grid on
axis([1 nt min(xk_log(3,:)) max(xk_log(3,:))])
% Aceleración x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p8 = plot(bin_log(1,lpx1:nt),xk_log(3,lpx1:nt),'-g');
title('Maniobra terminal del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración x en [g]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p8(1),'LineWidth', 2.0);
grid on
axis([lpx1 nt min(xk_log(3,lpx1:nt)) max(xk_log(3,lpx1:nt))])
%% Coordenada y
% Posición y: Trayectoria completa
figure(6);
subplot(1,2,1);
p9 = plot(bin_log(1,:),xk_log(4,:),'-g',bin_log(1,:),yk_log(3,:),'-r');
title('Trayectoria completa del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición y en [mn]','FontSize',22)
legend({'real','medición'},'Location','north','NumColumns',2,'FontSize',18)
set(p9(1),'LineWidth', 2.0);
grid on
axis([1 nt min([xk_log(4,:) yk_log(3,:)]) max([xk_log(4,:) yk_log(3,:)])])
% Posición y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
p10 = plot(bin_log(1,lpx1:nt),xk_log(4,lpx1:nt),'-
g',bin_log(1,lpx1:nt),yk_log(3,lpx1:nt),'-r');
title('Maniobra terminal del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición y en [mn]','FontSize',22)
legend({'real','medición'},'Location','north','NumColumns',2,'FontSize',18)
set(p10(1),'LineWidth', 2.0);
grid on
axis([lpx1 nt min([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt)]) max([xk_log(4,lpx1:nt)
yk_log(3,lpx1:nt)])])
% Velocidad y: Trayectoria completa
figure(7);
subplot(1,2,1)
p11 = plot(bin_log(1,:),xk_log(5,:),'-g',bin_log(1,:),yk_log(4,:),'-r');
title('Trayectoria completa del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad y en [mach]','FontSize',22)
legend({'real','medición'},'Location','northeast','NumColumns',2,'FontSize',18)
set(p11(1),'LineWidth', 2.0);
grid on
axis([1 nt min([xk_log(5,:) yk_log(4,:)]) max([xk_log(5,:) yk_log(4,:)])])
% Velocidad y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
```

```
p12 = plot(bin_log(1,lpx1:nt),xk_log(5,lpx1:nt),'-
g',bin_log(1,lpx1:nt),yk_log(4,lpx1:nt),'-r');
title('Maniobra terminal del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad y en [mach]','FontSize',22)
legend({'real','medición'},'Location','north','NumColumns',2,'FontSize',18)
set(p12(1),'LineWidth', 2.0);
grid on
axis([lpx1 nt min([xk_log(5,lpx1:nt) yk_log(4,lpx1:nt)]) max([xk_log(5,lpx1:nt)
yk_log(4,lpx1:nt)])])
% Aceleración y: Trayectoria completa
figure(8);
subplot(1,2,1)
p13 = plot(bin_log(1,:),xk_log(6,:),'-g');
title('Trayectoria completa del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración y en [g]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p13(1),'LineWidth', 2.0);
grid on
axis([1 nt min(xk_log(6,:)) max(xk_log(6,:))])
% Aceleración y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
p14 = plot(bin_log(1,lpx1:nt),xk_log(6,lpx1:nt),'-g');
title('Maniobra terminal del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración y en [g]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p14(1),'LineWidth', 2.0);
grid on
axis([lpx1 nt min(xk_log(6,lpx1:nt)) max(xk_log(6,lpx1:nt))])
%% Coordenada z
% Posición z: Trayectoria completa
figure(9);
subplot(1,2,1)
p15 = plot(bin_log(1,:),yk_log(5,:)./0.00054,'-r',bin_log(1,:),xk_log(7,:)./0.00054,'-
g');
title('Trayectoria completa del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición z en [mts]','FontSize',22)
legend({'medición','real'},'Location','north','NumColumns',2,'FontSize',18)
set(p15(1),'LineWidth', 1.5);
set(p15(2),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(7,:)./0.00054 yk_log(5,:)./0.00054]) max([xk_log(7,:)./0.00054
yk_log(5,:)./0.00054])])
% Posición z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
p16 = plot(bin_log(1,lpx1:nt),yk_log(5,lpx1:nt)./0.00054,'-
r',bin_log(1,lpx1:nt),xk_log(7,lpx1:nt)./0.00054,'-g');
title('Maniobra terminal del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición z en [mts]','FontSize',22)
legend({'medición','real'},'Location','north','NumColumns',2,'FontSize',18)
set(p16(1),'LineWidth', 1.5);
set(p16(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054])
max([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054])])
% Velocidad z: Trayectoria completa
figure(10);
subplot(1,2,1)
p17 = plot(bin_log(1,:),yk_log(6,:),'-r',bin_log(1,:),xk_log(8,:),'-g');
title('Trayectoria completa del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad z en [mach]','FontSize',22)
legend({'real','medición'},'Location','northeast','NumColumns',2,'FontSize',18)
set(p17(1),'LineWidth', 1.5);
set(p17(2),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(8,:) yk_log(6,:)]) max([xk_log(8,:) yk_log(6,:)])])
% Velocidad z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
```

```
p18 = plot(bin_log(1,lpx1:nt),yk_log(6,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(8,lpx1:nt),'-g');
title('Maniobra terminal del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad z en [mach]','FontSize',22)
legend({'real','medición'},'Location','north','NumColumns',2,'FontSize',18)
set(p18(1),'LineWidth', 1.5);
set(p18(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(8,lpx1:nt) yk_log(6,lpx1:nt)]) max([xk_log(8,lpx1:nt)
yk_log(6,lpx1:nt)])])
% Aceleración z: Trayectoria completa
figure(11);
subplot(1,2,1)
p19 = plot(bin_log(1,:),xk_log(9,:),'-g');
title('Trayectoria completa del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración z en [g]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p19(1),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(9,:)) max(xk_log(9,:))])
% Aceleración z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
p20 = plot(bin_log(1,lpx1:nt),xk_log(9,lpx1:nt),'-g');
title('Maniobra terminal del misil','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración z en [g]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p20(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min(xk_log(9,lpx1:nt)) max(xk_log(9,lpx1:nt))])
%% Grafica de señal de entrada de control uk
% Coordenada x
figure(12);
p21 = plot(bin_log(1,:),uk_log(1,:),'-g');
title('Señal de entrada de control: coordenada x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('sobre aceleración en [g/s]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p21(1),'LineWidth', 1.5);
grid on
axis([1 nt min(uk_log(1,:)) max(uk_log(1,:))])
% Coordenada y
figure(13);
p22 = plot(bin_log(1,:),uk_log(2,:),'-g');
title('Señal de entrada de control: coordenada y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('sobre aceleración en [g/s]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p22(1),'LineWidth', 1.5);
grid on
axis([1 nt min(uk_log(2,:)) max(uk_log(2,:))])
% Coordenada z
figure(14);
p23 = plot(bin_log(1,:),uk_log(3,:),'-g');
title('Señal de entrada de control: coordenada z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('sobre aceleración en [g/s]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p23(1),'LineWidth', 1.5);
grid on
axis([1 nt min(uk_log(3,:)) max(uk_log(3,:))])
%% Grafica de perturbaciones pk
% Coordenada x
figure(15);
p24 = plot(bin_log(1,:),pk_log(1,:),'-g');
title('Perturbación: coordenada x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('sobre aceleración en [g/s]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p24(1),'LineWidth', 1.5);
grid on
axis([1 nt min(pk_log(1,:)) max(pk_log(1,:))])
% Coordenada y
figure(16);
```

```
p25 = plot(bin_log(1,:),pk_log(2,:),'-g');
title('Perturbación: coordenada y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('sobre aceleración en [g/s]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p25(1),'LineWidth', 1.5);
grid on
axis([1 nt min(pk_log(2,:)) max(pk_log(2,:))])
% Coordenada z
figure(17);
p26 = plot(bin_log(1,:),pk_log(3,:),'-g');
title('Perturbación: coordenada z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('sobre aceleración en [g/s]','FontSize',22)
legend({'real'},'Location','north','NumColumns',1,'FontSize',18)
set(p26(1),'LineWidth', 1.5);
grid on
axis([1 nt min(pk_log(3,:)) max(pk_log(3,:))])
save ('sim_modelo.mat')
```

Apéndice II

Programas de Matlab del Capítulo III

Apéndice 2.1: Síntesis del observador ESSMO

```
function [Gl,Gn,P2,D2] = ESSMO_sintesis(Fk,Gk,Ck,Dk)
%=========================================================================
%
% Título            : ESSMO_sintesis.m
% Propósito         : Función para sintetizar las ganancias lineal y no
%                     lineal del observador de Edwards-Spurgeon
%
% Descripción       : Este programa permite la síntesis de las ganancias
%                     lineal y no lineal del observador de Edwards-Spurgeon.
%                     El modelo dinámico y de medición de la planta o
%                     proceso es un modelo lineal incierto que representa
%                     la dinámica de un blanco aéreo de alta
%                     maniobrabilidad.
%                     Para tal efecto, se utilizan transformaciones
%                     matriciales canónicas.
%
%                     Estructura del modelo lineal incierto del blanco aéreo
%                     xk_1        = Fk*xk + Gk*uk + Dk*pk;
%
%                     donde,
%                     xk_1 es el vector de estados futuros del modelo
%                     xk es el vector de estados actuales del modelo
%                     Fk es la matriz de transición del modelo
%                     Gk es la matriz de entrada de control del modelo
%                     uk es el vector de entrada de control del modelo
%                     Dk es la matriz de distribución de perturbaciones
%                     pk es el vector de perturbaciones del modelo
%
%                     Estructura del Observador Discreto de Walcott-Zak:
%                     xhatk_1 = A*xhatk + B*uk - Gl*yk + Gn*v
%
%                     donde,
%                     xhatk_1 es el vector de estados estimados futuros
%                     xhatk es el vector de estados estimados actuales
%                     uk es el vector de entrada del sistema
%                     yk es el vector de estados medidos
%                     v es la función discontinua
%                     Gl es la ganancia lineal de retroalimentación
%                     Gn es la ganancia no lineal de retroalimentación
%
%
% Fecha de creación : 01/07/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
```

```
%--------------------------------------------------------------------------
% Matrices nominales del sistema
%--------------------------------------------------------------------------
A       = Fk;           % nxn
B       = Gk;           % nxm
C       = Ck;           % pxn
D       = Dk;           % nxq
%--------------------------------------------------------------------------
% Extraer dimensiones de las matrices
%--------------------------------------------------------------------------
[n,q]   = size(D);
[p,n]   = size(C);
%--------------------------------------------------------------------------
% Chequear consistencia de matrices
%--------------------------------------------------------------------------
rk = abcdchk(A,D,C);
if ~isempty(rk)
  printf('Las matrices C y D no cuenta con la consistencia adecuada')
end
%--------------------------------------------------------------------------
% Primera transformación
%--------------------------------------------------------------------------
nc      = null(C);
Tc      = [nc'; C];
Ac      = Tc*A*inv(Tc);
Dc      = Tc*D;
Cc      = C*inv(Tc);
%--------------------------------------------------------------------------
% Particionar la matriz de distribución de entradas a fin de canalizar las
% perturbaciones hacia los canales de salida
%--------------------------------------------------------------------------
Dc1     = Dc(1:n-p,:);
Dc2     = Dc(n-p+1:n,:);
%--------------------------------------------------------------------------
% Segunda transformación (QR)
%--------------------------------------------------------------------------
[Q,R]  = qr(Dc2);
QT         = (flipud(Q'))';
Tb        = [ eye(n-p) -Dc1*inv(Dc2'*Dc2)*Dc2';
              zeros(p,n-p)                QT'];
Aa       = Tb*Ac*inv(Tb);
Da       = Tb*Dc;
Ca       = Cc*inv(Tb);
cerosTx  = tzero(Aa,Da,Ca,zeros(p,q));
%fprintf('El sistema cuenta con %.0f ceros de transmisión \n',cerosTx);
%--------------------------------------------------------------------------
% Colocar A11 y A211 en forma canónica observable
%--------------------------------------------------------------------------
A11     = Aa(1:n-p,1:n-p);
A211    = Aa(n-p+1:n-q,1:n-p);
[Ab,Db,Cb,Tobs,kk] = obsvf(A11,zeros(n-p,1),A211,1000*eps);
%--------------------------------------------------------------------------
% Cálculo de r, el cual representa la dimensión del subespacio no
% observable, así como el número de ceros invariantes de la tripleta A,B,C
%--------------------------------------------------------------------------
r = n - p -sum(kk);
%--------------------------------------------------------------------------
% Tercera transformación Tf
%--------------------------------------------------------------------------
Tf = [Tobs zeros(n-p,p);zeros(p,n-p) eye(p)];
Af = Tf*Aa*inv(Tf);
Df = Tf*Da;
Cf = Ca*inv(Tf);
%--------------------------------------------------------------------------
% Colocar ceros invariantes en el plano izquierdo del locus (LHP)
%--------------------------------------------------------------------------
if r>0
   A11o = Af(1:r,1:r);
   if any(real(eig(A11o))>-100*eps)
      fprintf('Existen ceros invariantes inestables en el sistema\n');
      Af = [];
      Df = [];
      Cf = [];
      To = [];
      r  = 0;
      return
   else
```

```
      fprintf('\n')
      fprintf('El sistema tiene %d ceros invariantes estables \n',r);
   end
end
To=Tf*Tb*Tc;
%--------------------------------------------------------------------------
% Verificar si la forma canónica es posible
%--------------------------------------------------------------------------
if isempty(To)
   fprintf('La forma canónica del observador no es posible\n')
   Ac = [];
   Dc = [];
   C  = [];
   Tc = [];
   return
end
%--------------------------------------------------------------------------
% Ubicación de los polos del modo deslizante
%--------------------------------------------------------------------------
if n-p-r>0
    A22o    = Af(r+1:n-p,r+1:n-p);
    A21o    = Af(n-p+1:n-q,r+1:n-p);
    % Configuración de polos
    p1       = -0.92;
    p2       = -0.99;
    p3       = -0.97;
    pLt      = [p1 p2 p3];
    Lt       = place(A22o',A21o', pLt)';
    if r >0
       L    = -[zeros(r,p-q);Lt];
    else
       L    = -Lt;
    end
else
    L    = zeros(n-p,p-q);
end
Lbar     = [L zeros(n-p,q)];
%--------------------------------------------------------------------------
% Cuarta transformación TL C=[0 I] D=[0 D2']
% Esto logra que la matriz A tenga una submatriz A11 con autovalores
% estables
%--------------------------------------------------------------------------
T  = Cf(:,n-p+1:n);
TL = [eye(n-p) Lbar ; zeros(p,n-p) T];
Ac = TL*Af*inv(TL);
Dc = TL*Df;
Cc = Cf*inv(TL);
Tc = TL*To;
%--------------------------------------------------------------------------
% Hallar matriz D2
%--------------------------------------------------------------------------
D2 = Dc(n-p+1:n,:);
%--------------------------------------------------------------------------
% Calcular matrices Luenberger en el eje de coordenadas modificado
%--------------------------------------------------------------------------
A12 = Ac(1:n-p,n-p+1:n);
A22 = Ac(n-p+1:n,n-p+1:n);
%--------------------------------------------------------------------------
% Fijar polos de estimación de error y hallar matriz P2 por medio de la
% función de Lyapunov
%--------------------------------------------------------------------------
% Configuración de polos sin ruido
p4      = -0.35;
p5      = -0.35;
p6      = -0.35;
p7      = -0.35;
p8      = -0.35;
p9      = -0.35;
% Configuración de polos con ruido
%p4      = -0.005;
%p5      = -0.005;
%p6      = -0.005;
%p7      = -0.005;
%p8      = -0.005;
%p9      = -0.005;

plyap = [p4 p5 p6 p7 p8 p9];
```

```
A22s  = diag(plyap);
Q2    = diag([1e-4;1e-4;1e-4;1e-4;1e-4;1e-4]);
P2    = lyap(A22s',Q2);
%-------------------------------------------------------------------------
% Calcular ganancias Gl y Gn en las coordenadas originales
%-------------------------------------------------------------------------
Gl = inv(Tc)*[A12;(A22-A22s)];
Gn = norm(D2)*inv(Tc)*[zeros(n-p,p);eye(p)];
save pLt plyap
end
```

Apéndice 2.2: Observador ESSMO

```
function [xhk,fi] = ESSMO(Gl,Gn,P2,D2,Fk,Gk,yk,uk,k,g)
%=========================================================================
%
% Título            : ESSMO.m
% Propósito         : Observador de Modo Deslizante de Edward-Spurgeon
%
% Descripción       : Este programa permite estimar variables de estado
%                     de posición, velocidad y aceleración de un blanco
%                     aéreo de alta maniobrabilidad al establecer un modo
%                     deslizante.
%       Observador:
%       xhatk = A*xhatk + B*uk - Gl*ey + Gn*v
%
%       donde,
%
%       xhatk es el vector de estados estimados
%       uk es el vector de entrada del sistema
%       ey es el vector de error de estados medidos
%       v es la función discontinua
%       Gl es la ganancia lineal de Luenberger
%       Gn es la ganancia no lineal de inyección discontinua
%
% Fecha de creación : 05/08/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
%-------------------------------------------------------------------------
% Persistencia de variables
%-------------------------------------------------------------------------
persistent yhatk xhatk

if isempty (xhatk)
   yhatk    = [0;0;0;0;0;0];%[35000;-200;25000;-160;20;20];
   xhatk    = [0;0;0;0;0;0;0;0;0];%[35000;-200;0*g;25000;-160;0*g;20;20;0*g];
end
%-------------------------------------------------------------------------
% Actualizar errores
%-------------------------------------------------------------------------
ey   = yhatk  - yk;
%-------------------------------------------------------------------------
% Función discontinua
%-------------------------------------------------------------------------
ro         = [0.005*g;0.005*g;0.005*g;0.005*g;0.005*g;0.005*g];
v          = -ro.*(norm(D2)*P2*ey./norm(P2*ey));
%-------------------------------------------------------------------------
% Observador de Edwards-Spurgeon
%-------------------------------------------------------------------------
xhatk_1    = Fk*xhatk + Gk*uk(:,k) - Gl*ey + Gn*v;
%-------------------------------------------------------------------------
% Reconstrucción de perturbación (Control Equivalente)
%-------------------------------------------------------------------------
delta = 0.01;
fi    = -[1.0*g; 0.85*g;
0.9*g].*(norm(D2)*inv(D2'*D2)*D2'*(P2*ey./(norm(P2*ey)+delta)));
%-------------------------------------------------------------------------
% Estados futuros medibles
%-------------------------------------------------------------------------
yhatk_1    = [xhatk_1(1);
```

```
              xhatk_1(2);
              xhatk_1(4);
              xhatk_1(5);
              xhatk_1(7);
              xhatk_1(8)];
%-------------------------------------------------------------------------
% Aging de variables
%-------------------------------------------------------------------------
yhatk      = yhatk_1;
xhatk      = xhatk_1;
xhk        = xhatk;
end
```

Apéndice 2.3: Simulación de la trayectoria del misil (ESSMO)

```
%=========================================================================
%
% Título            : sim_ESSMO.m
% Propósito         : Simulación de observador de modo deslizante de
%                     Edwards-Spurgeon
%
% Descripción       : Este programa permite la estimación de parámetros de
%                     posición, velocidad y aceleración de un blanco aéreo
%                     de alta maniobrabilidad a partir de las mediciones
%                     ruidosas de posición y velocidad en coordenadas
%                     cartesianas obtenidas por un radar de seguimiento
%                     automático. Para tal efecto, la ganancia lineal y no
%                     no lineal son sintetizadas por medio de
%                     transformaciones canónicas.
%                     Simulación con ruido = 1.
%                     Simulación sin ruido = 0.
%
% Fecha de creación : 20/08/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
clear all;
clc;
close all;
%-------------------------------------------------------------------------
% Inicialización de variables
%-------------------------------------------------------------------------
RMSEcold = 0;
RMSEpold = 0;
%-------------------------------------------------------------------------
% Parámetros iniciales de radar (sensor)
%-------------------------------------------------------------------------
T     = 1/50;                       % Período de muestreo (seg)
%-------------------------------------------------------------------------
% Definición de constantes físicas
%-------------------------------------------------------------------------
g     = 9.81;                       % gravedad (m/s2)
M     = 343;                        % velocidad del sonido (m/s)
load_factor = 100*g;                % máxima sobre aceleración (g/s)
%-------------------------------------------------------------------------
% Definición de vectores de tiempo
%-------------------------------------------------------------------------
ti    = 0;                          % Tiempo inicial de simulación
tff   = 104.5;                      % Tiempo final de simulación
tt    = ti:T:tff;                   % Vector de tiempo
tt    = tt';                        % Transpuesta del vector de tiempo
nt    = length(tt);                 % Número de muestras
%-------------------------------------------------------------------------
% Definición de posición y velocidad inicial del blanco en
% coordenadas esféricas
%-------------------------------------------------------------------------
Ad_init  = 35000;                       % Distancia al blanco (m)
Ed_init  = 0.01;                        % Elevación al blanco (°)
Bdn_init = 30;                          % Marcación al blanco (°)
Vm_init  = 0.9*M;                       % Velocidad del blanco (1 Mach = 343 m/s)
Rv_init  = 210;                         % Rumbo inicial del blanco (°)
%-------------------------------------------------------------------------
% Conversión de coordenadas esféricas a cartesianas (North-Sky-East) de la
```

```
% posición y velocidad inicial del blanco
%-------------------------------------------------------------------------
[Adx_init, Ady_init, Adz_init] = p2c(Ad_init,Ed_init,Bdn_init);
[Vmx_init, Vmy_init, Vmz_init] = p2c(Vm_init,0,Rv_init);
%-------------------------------------------------------------------------
% Inicialización de vector de estados
%-------------------------------------------------------------------------
xk = [Adx_init Vmx_init 0*g Ady_init Vmy_init 0*g Adz_init Vmz_init 0*g]';
%-------------------------------------------------------------------------
% Definición de matrices del modelo de espacio de estados del sistema
%-------------------------------------------------------------------------
% Matriz de medición (Pulse-Doppler radar)
Ck     = [1 0 0 0 0 0 0 0 0;        % pos x
          0 1 0 0 0 0 0 0 0;        % vel x
          0 0 0 1 0 0 0 0 0;        % pos y
          0 0 0 0 1 0 0 0 0;        % vel y
          0 0 0 0 0 0 1 0 0;        % pos z
          0 0 0 0 0 0 0 1 0];       % vel z

% Cálculo de matriz de transición Fk
Fk = [1 T (T^2)/2 0 0       0 0 0       0;
      0 1       T 0 0       0 0 0       0;
      0 0       1 0 0       0 0 0       0;
      0 0       0 1 T (T^2)/2 0 0       0;
      0 0       0 0 1       T 0 0       0;
      0 0       0 0 0       1 0 0       0;
      0 0       0 0 0       0 1 T (T^2)/2;
      0 0       0 0 0       0 0 1       T;
      0 0       0 0 0       0 0 0       1];

% Cálculo de la matriz de entrada Gk
Gk = [(T^3)/6       0         0;
      (T^2)/2       0         0;
       T            0         0;
       0       (T^3)/6        0;
       0       (T^2)/2        0;
       0            T         0;
       0            0   (T^3)/6;
       0            0   (T^2)/2;
       0            0         T];

% Matriz de distribución de perturbaciones (aceleraciones)
Dk = [75e4*(T^3)/6       0                0;
      5e3*(T^2)/2        0                0;
       1e2*T             0                0;
       0       75e4*(T^3)/6                0;
       0        5e3*(T^2)/2                0;
       0              1e2*T                0;
       0                 0     75e4*(T^3)/6;
       0                 0      5e3*(T^2)/2;
       0                 0            1e2*T];
%-------------------------------------------------------------------------
% Generación del vector de sobre aceleraciones de entrada uk
%-------------------------------------------------------------------------
uk = gen_jerk_sea_skimming3(nt,load_factor,g,T);
%-------------------------------------------------------------------------
% Generación del vector de perturbaciones ?k
%-------------------------------------------------------------------------
pk = perturb_1(g,ti,T,tff);
%-------------------------------------------------------------------------
% Cargar ruido glint
%-------------------------------------------------------------------------
load glint_puntos.mat
%-------------------------------------------------------------------------
% Síntesis de ganancias Gl y Gn
%-------------------------------------------------------------------------
[Gl,Gn,P2,D2] = ESSMO_sintesis(Fk,Gk,Ck,Dk);
%-------------------------------------------------------------------------
% Seleccionar simulación con ruido o sin ruido
%-------------------------------------------------------------------------
disp('Simulación de la trayectoria del blanco aéreo de alta maniobrabilidad')
disp('Modelo lineal incierto')
prompt = 'Para simular con ruido presione "1". Para simular sin ruido presione "0": ';
gl     = input(prompt);
%-------------------------------------------------------------------------
% Simulación del modelo
%-------------------------------------------------------------------------
```

```
for k = 1:nt

    % Modelo dinámico o de transición de estados
    xk       = Fk*xk + Gk*uk(:,k) + Dk*pk(:,k);
    % Simulación de adquisición de datos por el radar
    % Datos de posición adquiridos en coordenadas esféricas
    [Ad,Bdn,Ed] = c2p(xk);
    % Modelo de medición polar a cartesiano
    yk = yk_noise(Ad,Bdn,Ed,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,gl,k);
    % Observador de modo deslizante Edwards-Spurgeon
    tic
    [xhk,fi] = ESSMO(Gl,Gn,P2,D2,Fk,Gk,yk,uk,k,g);
    TCPU = toc;

    % Conversión de coordenada estimadas cartesianas a coordenadas polares
    [Adh,Bdnh,Edh] = c2p(xhk);

    % Cálculo de errores en coordenadas cartesianas
    errorc       = xhk - xk;

    % Cálculo de errores en coordenadas esféricas
    errorAd      = Adh - Ad;                                    %mts
    errorBdn     = Bdnh*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEd      = Edh*(pi*1000/180) - Ed*(pi*1000/180);
    errorp       = [errorAd; errorBdn; errorEd];

    % Cálculo de RMSE
    if ( k >= 4650)
        RMSEc         = RMSEcold + ((errorc).^2);
        RMSEp         = RMSEpold + ((errorp).^2);
    else
        RMSEc         = 0;
        RMSEp         = 0;
    end

    % Guardar señales
    bin_log(:,k)      = k;
    xk_log(:,k)       = xk;
    yk_log(:,k)       = yk;
    xhk_log(:,k)      = xhk;
    errorp_log(:,k)   = errorp;
    errorc_log(:,k)   = errorc;
    polar_log(:,k)    = [Ad;Bdn;Ed];
    polarhat_log(:,k) = [Adh;Bdnh;Edh];
    fi_log(:,k)       = fi;
    TCPU_log(:,k)     = TCPU;
    % Age variables
    RMSEcold          = RMSEc;
    RMSEpold          = RMSEp;
end
RMSEctot = sqrt(RMSEc)/(nt-4650);
RMSEptot = sqrt(RMSEp)/(nt-4650);
%-------------------------------------------------------------------------
% Conversión de vector de estados reales a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
% 0.10197 - de mts/seg2 a g's
conv_xk   = [0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197]';
xk_log = xk_log.*conv_xk;
%-------------------------------------------------------------------------
% Conversión de vector de estados medidos a Mn, Mach
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
conv_yk   = [ 0.00054 0.00292 0.00054 0.00292 0.00054 0.00292 ]';
yk_log    = yk_log.*conv_yk;
%-------------------------------------------------------------------------
% Conversión de vector de estados estimados a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - mn
% 0.00292 - Mach
% 0.10197 - g's
conv_xhk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292
0.10197]';
xhk_log = xhk_log.*conv_xhk;
%-------------------------------------------------------------------------
```

```
% Conversión de vector de entrada real y estimado a g/s
%--------------------------------------------------------------------------
uk_log    = uk./g;
%--------------------------------------------------------------------------
% Conversión de perturbaciones a g/s
%--------------------------------------------------------------------------
pk_log    = pk./g;
%--------------------------------------------------------------------------
% Plots
%--------------------------------------------------------------------------
%% 3D trayectoria completa

figure(1);p1 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                  0,                  0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                   0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p1(1),'LineWidth',2.0);
set(p1(2),'LineWidth',1.8);
set(p1(3),'LineWidth',1.5);
set(p1(4),'LineWidth',2.0);
set(p1(5),'LineWidth',2.0);
set(p1(6),'LineWidth',2.0);
set(p1(7),'LineWidth',2.0);

axis([-5 20 -5 10 0 0.04]);          % Toda la trayectoria
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Trayectoria sea skimming del misil','FontSize',16)
legend({'mediciones','real','ESSMO','Posición del buque observador','Posición inicial
del misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
%% 3D trayectoria completa

figure(2);p2 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                  0,                  0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                   0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p2(1),'LineWidth',2.0);
set(p2(2),'LineWidth',1.8);
set(p2(3),'LineWidth',1.5);
set(p2(4),'LineWidth',2.0);
set(p2(5),'LineWidth',2.0);
set(p2(6),'LineWidth',2.0);
set(p2(7),'LineWidth',2.0);
axis([4 8 -2 5 0 0.016]);
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Maniobra terminal del misil','FontSize',16)
legend({'mediciones','real','ESSMO','Posición del buque observador','Posición inicial
del misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14);

%% Coordenada x
% Posición x: Trayectoria completa
figure(3);
subplot(1,2,1);
p3 = plot(bin_log(1,:),yk_log(1,:),'-r',bin_log(1,:),xk_log(1,:),'-
g',bin_log(1,:),xhk_log(1,:),'-k');
title('Trayectoria completa del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p3(1),'LineWidth', 2.0);
```

```
set(p3(2),'LineWidth', 1.8);
set(p3(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(1,:) yk_log(1,:)]) max([xk_log(1,:) yk_log(1,:)])])

% Posición x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p4 = plot(bin_log(1,lpx1:nt),yk_log(1,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(1,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(1,lpx1:nt),'-
k');
title('Maniobra terminal del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p4(1),'LineWidth', 2.0);
set(p4(2),'LineWidth', 1.8);
set(p4(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)]) max([xk_log(1,lpx1:nt)
yk_log(1,lpx1:nt)])])
%%
% Velocidad x: Trayectoria completa
figure(4);
subplot(1,2,1);
p5 = plot(bin_log(1,:),yk_log(2,:),'-r',bin_log(1,:),xk_log(2,:),'-
g',bin_log(1,:),xhk_log(2,:),'-k');
title('Trayectoria completa del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p5(1),'LineWidth', 2.0);
set(p5(2),'LineWidth', 1.8);
set(p5(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(2,:) yk_log(2,:)]) max([xk_log(2,:) yk_log(2,:)])])
%axis([1 nt -10 10])
% Velocidad x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p6 = plot(bin_log(1,lpx1:nt),yk_log(2,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(2,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(2,lpx1:nt),'-
k');
title('Maniobra terminal del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p6(1),'LineWidth', 2.0);
set(p6(2),'LineWidth', 1.8);
set(p6(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt -1 2]) % con ruido
%axis([lpx1 nt min([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])
max([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])])
%%
% Aceleración x: Trayectoria completa
figure(5);
subplot(1,2,1);
p7 = plot(bin_log(1,:),xk_log(3,:),'-g',bin_log(1,:),xhk_log(3,:),'-k');
title('Trayectoria completa del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ESSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p7(1),'LineWidth', 1.8);
set(p7(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(3,:)) max(xk_log(3,:))])
%axis([1 nt min([xk_log(3,:) xhk_log(3,:)]) max([xk_log(3,:) xhk_log(3,:)])])
% Aceleración x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
p8 = plot(bin_log(1,lpx1:nt),xk_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(3,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ESSMO'},'Location','north','NumColumns',2,'FontSize',18)
```

```
set(p8(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(3,lpx1:nt) xhk_log(3,lpx1:nt)]) max([xk_log(3,lpx1:nt)
xhk_log(3,lpx1:nt)])])
%axis([lpx1 nt -20 20])
%% Coordenada y
% Posición y: Trayectoria completa
figure(6);
subplot(1,2,1);
p9 = plot(bin_log(1,:),yk_log(3,:),'-r',bin_log(1,:),xk_log(4,:),'-
g',bin_log(1,:),xhk_log(4,:),'-k');
title('Trayectoria completa del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p9(1),'LineWidth', 2.0);
set(p9(2),'LineWidth', 1.8);
set(p9(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(4,:) yk_log(3,:)]) max([xk_log(4,:) yk_log(3,:)])])

% Posición y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p10 = plot(bin_log(1,lpx1:nt),yk_log(3,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(4,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(4,lpx1:nt),'-
K');
title('Maniobra terminal del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p10(1),'LineWidth', 2.0);
set(p10(2),'LineWidth', 1.8);
set(p10(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])
max([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])])
%%
% Velocidad y: Trayectoria completa
figure(7);
subplot(1,2,1);
p11 = plot(bin_log(1,:),yk_log(4,:),'-r',bin_log(1,:),xk_log(5,:),'-
g',bin_log(1,:),xhk_log(5,:),'-k');
title('Trayectoria completa del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p11(1),'LineWidth', 2.0);
set(p11(2),'LineWidth', 1.8);
set(p11(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(5,:) yk_log(4,:)]) max([xk_log(5,:) yk_log(4,:)])])
%axis([1 nt -10 10])
% Velocidad y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p12 = plot(bin_log(1,lpx1:nt),yk_log(4,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(5,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(5,lpx1:nt),'-
k');
title('Maniobra terminal del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p12(1),'LineWidth', 2.0);
set(p12(2),'LineWidth', 1.8);
set(p12(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(5,lpx1:nt) yk_log(4,lpx1:nt)]) max([xk_log(5,lpx1:nt)
yk_log(4,lpx1:nt)])])
%%
figure(8);
% Aceleración y: Trayectoria
subplot(1,2,1);
p14 = plot(bin_log(1,:),xk_log(6,:),'-g',bin_log(1,:),xhk_log(6,:),'-k');
title('Maniobra terminal del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
```

```
legend({'real','ESSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p14(1),'LineWidth', 1.8);
set(p14(2),'LineWidth', 1.5);
grid on
%axis([1 nt min([xk_log(6,:) xhk_log(6,:)]) max([xk_log(6,:) xhk_log(6,:)])])
axis([1 nt min(xk_log(6,:)) max(xk_log(6,:))])
% Aceleración y: Maniobra terminal
subplot(1,2,2);
p13 = plot(bin_log(1,lpx1:nt),xk_log(6,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(6,lpx1:nt),'-k');
title('Trayectoria completa del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ESSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p13(1),'LineWidth', 1.8);
set(p13(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(6,lpx1:nt) xhk_log(6,lpx1:nt)]) max([xk_log(6,lpx1:nt)
xhk_log(6,lpx1:nt)])])

%% Coordenada z
% Posición z: Trayectoria completa
figure(9);
subplot(1,2,1);
p15 = plot(bin_log(1,:),yk_log(5,:)./0.00054,'-r',bin_log(1,:),xk_log(7,:)./0.00054,'-
g',bin_log(1,:),xhk_log(7,:)./0.00054,'-k');
title('Trayectoria completa del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p15(1),'LineWidth', 2.0);
set(p15(2),'LineWidth', 1.8);
set(p15(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(7,:)./0.00054 yk_log(5,:)./0.00054]) max([xk_log(7,:)./0.00054
yk_log(5,:)./0.00054])])

% Posición z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p16 = plot(bin_log(1,lpx1:nt),yk_log(5,lpx1:nt)./0.00054,'-
r',bin_log(1,lpx1:nt),xk_log(7,lpx1:nt)./0.00054,'-
g',bin_log(1,lpx1:nt),xhk_log(7,lpx1:nt)./0.00054,'-k');
title('Maniobra terminal del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mts]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p16(1),'LineWidth', 2.0);
set(p16(2),'LineWidth', 1.8);
set(p16(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054]) max([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054])])
%%
% Velocidad z: Trayectoria completa
figure(10);
subplot(1,2,1);
p17 = plot(bin_log(1,:),yk_log(6,:),'-r',bin_log(1,:),xk_log(8,:),'-
g',bin_log(1,:),xhk_log(8,:),'-k');
title('Trayectoria completa del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p17(1),'LineWidth', 2.0);
set(p17(2),'LineWidth', 1.8);
set(p17(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(8,:) yk_log(6,:)]) max([xk_log(8,:) yk_log(6,:)])])
%axis([1 nt -4 4])
% Velocidad z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p18 = plot(bin_log(1,lpx1:nt),yk_log(6,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(8,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(8,lpx1:nt),'-
k');
title('Maniobra terminal del misil: velocidad z','FontSize',22);
```

```
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','ESSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p18(1),'LineWidth', 1.0);
set(p18(2),'LineWidth', 1.8);
set(p18(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(8,lpx1:nt) yk_log(6,lpx1:nt)]) max([xk_log(8,lpx1:nt)
yk_log(6,lpx1:nt)])])
%%
% Aceleración z: Trayectoria completa
figure(11);
subplot(1,2,1);
p19 = plot(bin_log(1,:),xk_log(9,:),'-g',bin_log(1,:),xhk_log(9,:),'-k');
title('Trayectoria completa del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ESSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p19(1),'LineWidth', 1.8);
set(p19(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(9,:)) max(xk_log(9,:))])
%axis([1 nt min([xk_log(9,:) xhk_log(9,:)]) max([xk_log(9,:) xhk_log(9,:)])])

% Aceleración z: Maniobra terminal
subplot(1,2,2);
p20 = plot(bin_log(1,lpx1:nt),xk_log(9,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(9,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ESSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p20(1),'LineWidth', 1.8);
set(p20(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(9,lpx1:nt) xhk_log(9,lpx1:nt)]) max([xk_log(9,lpx1:nt)
xhk_log(9,lpx1:nt)])])

%% Plots señales de errores cartesianos posición x,y,z
figure(12);
sgtitle('Errores cartesianos','FontSize',18)
lpx1 = 4500;
subplot(3,2,1);
p21 = plot(bin_log(1,:),errorc_log(1,:),'-g',bin_log(1,:),errorc_log(4,:),'-
c',bin_log(1,:),errorc_log(7,:),'-m');
title('Trayectoria completa: posición','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m]','FontSize',16)
legend({'pos x','pos y','pos z'},'Location','north','NumColumns',3,'FontSize',14)
set(p21,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(1,:) errorc_log(4,:) errorc_log(7,:)]) max([errorc_log(1,:)
errorc_log(4,:) errorc_log(7,:)])])

subplot(3,2,2);
p22 = plot(bin_log(1,lpx1:nt),errorc_log(1,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(4,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(7,lpx1:nt),'-m');
title('Maniobra terminal: posición','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p22,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])
max([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])])
% Plots señales de errores cartesianos velocidad x,y,z
subplot(3,2,3);
p23 = plot(bin_log(1,:),errorc_log(2,:),'-g',bin_log(1,:),errorc_log(5,:),'-
c',bin_log(1,:),errorc_log(8,:),'-m');
title('Trayectoria completa: velocidad','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s]','FontSize',16)
legend({'vel x','vel y','vel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p23,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(2,:) errorc_log(5,:) errorc_log(8,:)]) max([errorc_log(2,:)
errorc_log(5,:) errorc_log(8,:)])])
```

```
subplot(3,2,4);
p24 = plot(bin_log(1,lpx1:nt),errorc_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(5,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(8,lpx1:nt),'-m');
title('Maniobra terminal: velocidad','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s]','FontSize',16);
set(p24,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])
max([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])])
% Plots señales de errores cartesianos aceleración x,y,z
subplot(3,2,5);
p25 = plot(bin_log(1,:),errorc_log(3,:),'-g',bin_log(1,:),errorc_log(6,:),'-
c',bin_log(1,:),errorc_log(9,:),'-m');
title('Trayectoria completa: aceleración','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s2]','FontSize',16)
legend({'acel x','acel y','acel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p25,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(3,:) errorc_log(6,:) errorc_log(9,:)]) max([errorc_log(3,:)
errorc_log(6,:) errorc_log(9,:)])])

subplot(3,2,6);
p26 = plot(bin_log(1,lpx1:nt),errorc_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(6,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(9,lpx1:nt),'-m');
title('Maniobra terminal: aceleración','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s2]','FontSize',16);
set(p26,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])
max([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])])

%% Plots señales de error polares: Distancia Ad
figure(13);
sgtitle('Errores polares','FontSize',18)
maxAd =  37.5*ones(1,nt);
minAd = -37.5*ones(1,nt);

subplot(3,2,1);
laz1 = 4500;
p27 = plot(bin_log(1,:),errorp_log(1,:),'-g',bin_log(1,:),maxAd(1,:),'--
r',bin_log(1,:),minAd(1,:),'--r');
title('Trayectoria completa: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
legend({'Ad','Límite +/- 37.5 mts'},'NumColumns',2,'Location','North','FontSize',12);
set(p27(1),'LineWidth',1.5);
set(p27(2),'LineWidth',1.0);
set(p27(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(1,:) minAd(1,:)]) max([errorp_log(1,:) maxAd(1,:)])])

subplot(3,2,2);
p28 = plot(bin_log(1,laz1:nt),errorp_log(1,laz1:nt),'-
g',bin_log(1,laz1:nt),maxAd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minAd(1,laz1:nt),'--r');
title('Maniobra terminal: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p28(1),'LineWidth',1.5);
set(p28(2),'LineWidth',1.0);
set(p28(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(1,laz1:nt) minAd(1,laz1:nt)]) max([errorp_log(1,laz1:nt)
maxAd(1,laz1:nt)])])
% Error polar en acimut
maxBdn = 2*ones(1,nt);
minBdn = -2*ones(1,nt);

subplot(3,2,3);
laz1 = 4500;
p29 = plot(bin_log(1,:),errorp_log(2,:),'-g',bin_log(1,:),maxBdn(1,:),'--
r',bin_log(1,:),minBdn(1,:),'--r');
```

```
title('Trayectoria completa: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Bdn','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p29(1),'LineWidth',1.5);
set(p29(2),'LineWidth',1.0);
set(p29(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(2,:) minBdn(1,:)]) max([errorp_log(2,:) maxBdn(1,:)])])

subplot(3,2,4);
p30 = plot(bin_log(1,laz1:nt),errorp_log(2,laz1:nt),'-
g',bin_log(1,laz1:nt),maxBdn(1,laz1:nt),'--r',bin_log(1,laz1:nt),minBdn(1,laz1:nt),'--
r');
title('Maniobra terminal: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p30(1),'LineWidth',1.5);
set(p30(2),'LineWidth',1.0);
set(p30(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(2,laz1:nt) minBdn(1,laz1:nt)]) max([errorp_log(2,laz1:nt)
maxBdn(1,laz1:nt)])])

maxEd = 2*ones(1,nt);
minEd = -2*ones(1,nt);

subplot(3,2,5);
laz1 = 4500;
p31 = plot(bin_log(1,:),errorp_log(3,:),'-g',bin_log(1,:),maxEd(1,:),'--
r',bin_log(1,:),minEd(1,:),'--r');
title('Trayectoria completa: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Ed','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p31(1),'LineWidth',1.5);
set(p31(2),'LineWidth',1.0);
set(p31(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(3,:) minEd(1,:)]) max([errorp_log(3,:) maxEd(1,:)])])

subplot(3,2,6);
p32 = plot(bin_log(1,laz1:nt),errorp_log(3,laz1:nt),'-
g',bin_log(1,laz1:nt),maxEd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minEd(1,laz1:nt),'--r');
title('Maniobra terminal: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p32(1),'LineWidth',1.5);
set(p32(2),'LineWidth',1.0);
set(p32(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(3,laz1:nt) minEd(1,laz1:nt)]) max([errorp_log(3,laz1:nt)
maxEd(1,laz1:nt)])])
%% Perturbaciones

figure(14);
sgtitle('Perturbaciones','FontSize',18)
lpx1 = 4500;
subplot(3,2,1);
p33 = plot(bin_log(1,:),pk_log(1,:),'-g',bin_log(1,:),fi_log(1,:),'-k');
title('Trayectoria completa: perturbación x','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('[g/s]','FontSize',16)
legend({'real','ESSMO'},'Location','north','NumColumns',2,'FontSize',14)
set(p33,'LineWidth',1.5);
grid on
axis([1 nt min([pk_log(1,:) fi_log(1,:)]) max([pk_log(1,:) fi_log(1,:)])])

subplot(3,2,2);
p34 = plot(bin_log(1,lpx1:nt),pk_log(1,lpx1:nt),'-
g',bin_log(1,lpx1:nt),fi_log(1,lpx1:nt),'-k');
title('Maniobra terminal: perturbación x','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p34,'LineWidth',1.5);
grid on
```

```
axis([lpx1 nt min([pk_log(1,lpx1:nt) fi_log(1,lpx1:nt)]) max([pk_log(1,lpx1:nt)
fi_log(1,lpx1:nt)])])

subplot(3,2,3);
p35 = plot(bin_log(1,:),pk_log(2,:),'-g',bin_log(1,:),fi_log(2,:),'-k');
title('Trayectoria completa: perturbación y','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('[g/s]','FontSize',16)
legend({'real','ESSMO'},'Location','north','NumColumns',2,'FontSize',14)
set(p35,'LineWidth',1.5);
grid on
axis([1 nt min([pk_log(2,:) fi_log(2,:)]) max([pk_log(2,:) fi_log(2,:)])])

subplot(3,2,4);
p36 = plot(bin_log(1,lpx1:nt),pk_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),fi_log(2,lpx1:nt),'-k');
title('Maniobra terminal: perturbación y','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p36,'LineWidth',1.5);
grid on
axis([lpx1 nt min([pk_log(2,lpx1:nt) fi_log(2,lpx1:nt)]) max([pk_log(2,lpx1:nt)
fi_log(2,lpx1:nt)])])

subplot(3,2,5);
p37 = plot(bin_log(1,:),pk_log(3,:),'-g',bin_log(1,:),fi_log(3,:),'-k');
title('Trayectoria completa: perturbación z','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('[g/s]','FontSize',16)
legend({'real','ESSMO'},'Location','north','NumColumns',2,'FontSize',14)
set(p37,'LineWidth',1.5);
grid on
axis([1 nt min([pk_log(3,:) fi_log(3,:)]) max([pk_log(3,:) fi_log(3,:)])])

subplot(3,2,6);
p38 = plot(bin_log(1,lpx1:nt),pk_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),fi_log(3,lpx1:nt),'-k');
title('Maniobra terminal: perturbación z','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p38,'LineWidth',1.5);
grid on
axis([lpx1 nt min([pk_log(3,lpx1:nt) fi_log(3,lpx1:nt)]) max([pk_log(3,lpx1:nt)
fi_log(3,lpx1:nt)])])

save ESSMO_datos_simulacion.mat
```

Apéndice 2.4: Síntesis del observador WZSMO

```
function [Gl,Gn,Z3,Ao,s0] = WZSMO_sintesis(Fk,Gk,Ck,Dk,gl)
%=========================================================================
%
% Título            : WZSMO_sintesis.m
% Propósito         : Función para sintetizar las ganancias lineal y no
%                     lineal del observador de Walcott-Zak
%
% Descripción       : Este programa permite la síntesis de las ganancias
%                     lineal y no lineal del observador de Walcott-Zak.
%                     El modelo dinámico y de medición de la planta o
%                     proceso es un modelo lineal incierto que representa
%                     la dinámica de un blanco aéreo de alta
%                     maniobrabilidad.
%                     Para tal efecto, la ganancia lineal es
%                     sintetizada mediante el algoritmo LQG (Linear Quadra-
%                     tic Gaussian) y la ganancia no lineal es sintetizada
%                     mediante desigualdades lineales matriciales (LMI).
%
%                     Estructura del modelo lineal incierto del blanco aéreo
%                     xk_1        = Fk*xk + Gk*uk + Dk*pk;
%
%                     donde,
%                     xk_1 es el vector de estados futuros del modelo
%                     xk es el vector de estados actuales del modelo
%                     Fk es la matriz de transición del modelo
%                     Gk es la matriz de entrada de control del modelo
%                     uk es el vector de entrada de control del modelo
```

```
%                       Dk es la matriz de distribución de perturbaciones
%                       pk es el vector de perturbaciones del modelo
%
%                       Estructura del Observador Discreto de Walcott-Zak:
%                       xhatk_1 = Ao*xhatk + B*uk + Gl*yk - Gn*v
%
%                       donde,
%                       xhatk_1 es el vector de estados estimados futuros
%                       xhatk es el vector de estados estimados actuales
%                       uk es el vector de entrada del sistema
%                       yk es el vector de estados medidos
%                       v es la función discontinua
%                       Gl es la ganancia lineal de retroalimentación
%                       Gn es la ganancia no lineal de retroalimentación
%                       Ao = A - Gl*C
%
% Fecha de creación : 04/08/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================

%
%       Observador de Modo Deslizante de Walcott-Zak
%       Sintetización de ganancias Gl y Gn por LMI
%
%
%       Italo Aranda Cetraro
%       Pontificia Universidad Católica del Perú
%       Email: a20194480@pucp.edu.pe
%       04/08/2020
%
%
%
%-------------------------------------------------------------------------
% Matrices del sistema de espacio de estados
%-------------------------------------------------------------------------
A = Fk;
B = Gk;
D = Dk;
C = Ck;
%-------------------------------------------------------------------------
% Matriz ortonormal de D
%-------------------------------------------------------------------------
DT = null(D');
%-------------------------------------------------------------------------
% Matriz de pesos de performance del LQG
%-------------------------------------------------------------------------
% Disminuir pesos para minimizar corrupción por ruido
% Simulación con ruido
if gl == 1
   W = diag([0.1 0.1 0.8 0.1 0.1 0.8 0.1 0.1 0.8]);
else
% Simulación sin ruido
   W = diag([0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8]);
end
%-------------------------------------------------------------------------
% Matriz de covarianza del ruido de medición
%-------------------------------------------------------------------------
Y   = diag([7.98e-05 4.56e-05 1.39e-04 4.56e-05 5.18e-06 4.56e-05]);
%Y   = diag(cov(nruido_log));
%Y   = diag(Y');
%-------------------------------------------------------------------------
% Síntesis de la ganancia lineal de retroalimentación Gl
%-------------------------------------------------------------------------
Qare      = dare(A', C',W,Y);
Gl        = Qare*C';%*(Y^-1);
Ao        = A - Gl*C;
eig(Ao)
%-------------------------------------------------------------------------
% Síntesis de la ganancia no lineal de retroalimentación Gn
%-------------------------------------------------------------------------
kk     = 10000;
s0     = 1000;
```

```
I9     = eye(9);
I3     = eye(3);
I6     = eye(6);
% Variables de las LMIs
setlmis([])
d      = lmivar(1,[1 1]);              % norm(Gn)
W3     = lmivar(1,[6 1]);
W4     = lmivar(1,[6 1]);
b      = lmivar(1,[1 1]);
Q      = lmivar(1,[9 1]);

% Primera LMI
%   [-k*I3 D'*C'*W4;
%     W4*C*D -k*I6 ]   <0
%
lmiterm([1 1 1 0],-kk*I3);              % -k*I3
lmiterm([1 1 2 W4],D'*C',1);            % D'*C'*W4
lmiterm([1 2 1 W4],1,C*D);              % W4*C*D
lmiterm([1 2 2 0],-kk*I6);              % -k*I6
% Segunda LMI
%   [-d*I     I;
%      I    -Q];   <0
%
lmiterm([2 1 1 d],-1,I9);               % -dI
lmiterm([2 1 2 0],I9);                  % I
lmiterm([2 2 1 0],I9);                  % I
lmiterm([2 2 2 Q],-1,1);                % -Q

% Tercera LMI
%  Q*Ao + Ao'Q - sig0C'C < 0
%
lmiterm([3 1 1 Q],  1,Ao,'s');          % Q*Ao + Ao'Q
lmiterm([3 1 1 0],  -s0*C'*C);          % -sig0*C'*C

% Cuarta LMI
% Igualdad Q  = DT*W3*DT' + C'*W4*C;
% es lo mismo que (Q  - DT*W3*GT' - C'*W4*C)'*(Q  - DT*W3*DT' - C'*W4*C) <= b*I
% y su LMI es:
% [                  -b*I   (Q  - DT*W3*DT' - C'*W4*C)';
%  (Q  - DT*W3*DT' - C'*W4*C)                          -I];

lmiterm([4 1 1 b] ,  -I9,1);            % -b*I9
lmiterm([4 1 2 -Q],    1,1);            % Q'
lmiterm([4 1 2 -W3],  -DT,DT');         % -DT*W3'*DT'
lmiterm([4 1 2 -W4],  -C',C);           % -C'*W4'*C
lmiterm([4 2 1 Q] ,  1,1);              % Q
lmiterm([4 2 1 W3],  -DT,DT');          % - DT*W3*DT'
lmiterm([4 2 1 W4],  -C',C);            % - C'*W4*C
lmiterm([4 2 2 0] ,  -I9);              % -I9

% Formar sistema LMI
lmisys = getlmis;
%[tmin,xfeas] = feasp(lmisys);
n_dec = decnbr(lmisys);
c = zeros(n_dec,1);

for i=1:n_dec
    c(i) = trace(defcx(lmisys,i,d));
end
opts = [1e-4,2000,-1,10,0];
[copt,xopt] = mincx(lmisys,c,opts);
evals = evallmi(lmisys,xopt);

% Matrices optimizadas
dopt        = dec2mat(lmisys,xopt,d);
W3opt       = dec2mat(lmisys,xopt,W3);
W4opt       = dec2mat(lmisys,xopt,W4);
bopt        = dec2mat(lmisys,xopt,b);
Qopt        = dec2mat(lmisys,xopt,Q);
% Ganancia no lineal de retroalimentación (Gn)
Gn          = (Qopt^-1)*C';
Z3          = D'*C'*W4opt;
%[Aod,Gn]    = c2d(Ao,Gn,dt);
%save sintesisWZSMO.mat
save sintesisWZSMOconruido.mat
end
```

Apéndice 2.5: Observador WZSMO

```
function [xhk] = WZSMO(Gl,Gn,Z3,Ao,Gk,yk,uk,k,g,s0)
%
%       Observador de Modo Deslizante de Walcott-Zak
%       Sistema de Estructura Variable
%
%       Observador:
%       xhatk = A*xhatk + B*uk + Gl*ey - Gn*v
%
%       donde,
%
%       xhatk es el vector de estados estimados
%       uk es el vector de entrada del sistema
%       ey es el vector de error de estados medidos
%       v es la función discontinua
%       Gl es la ganancia lineal de Luenberger
%       Gn es la ganancia no lineal de inyección discontinua
%
%       Italo Aranda Cetraro
%       Pontificia Universidad Católica del Perú
%       Email: a20194480@pucp.edu.pe
%       04/08/2020
%
%
%
%-------------------------------------------------------------------------
% Persistencia de variables
%-------------------------------------------------------------------------
persistent yhatk xhatk ey_log_old

if isempty (yhatk)
   yhatk    = [35000;-200;25000;-160;20;20];
   xhatk    = [35000;-200;2*g;25000;-160;2*g;20;20;2*g];
   ey_log_old   = [0;0;0;0;0;0];
end
%-------------------------------------------------------------------------
% Actualizar errores
%-------------------------------------------------------------------------
ey     = yk - yhatk;
ey_log = [ey_log_old ey];
%-------------------------------------------------------------------------
% Función discontinua
%-------------------------------------------------------------------------
n          = 1;     % da mas peso al error inicial
alfa       = 1000;  % límite superior de la perturbación m/s3
%
% Al aumentar el parámetro sig se hace que el observador sea mas robusto a
% las perturbaciones y tenga mayor velocidad de convergencia. Sin embargo,
% esto afecta la sensibilidad al ruido.
sig        = [1;0.0001;0.0001;0.00001;0.1;0.00001].*s0; % simulación sin ruido
%sig         = 0.0005; % simulación con ruido
v          = -sig.*ey - norm(Z3)*alfa.*(ey./(norm(ey_log))) - n*(ey./(norm(ey_log)));
%-------------------------------------------------------------------------
% Observador de Walcott-Zak
%-------------------------------------------------------------------------
xhatk_1    = Ao*xhatk + Gk*uk(:,k) + Gl*yk - Gn*v;
%-------------------------------------------------------------------------
% Separación de estados futuros medibles de no medibles
%-------------------------------------------------------------------------
yhatk_1    = [xhatk_1(1);
              xhatk_1(2);
              xhatk_1(4);
              xhatk_1(5);
              xhatk_1(7);
              xhatk_1(8)];
%-------------------------------------------------------------------------
% Guardar variables para siguiente iteración
%-------------------------------------------------------------------------
yhatk      = yhatk_1;
xhatk      = xhatk_1;
ey_log_old = ey_log;
xhk        = xhatk;
end
```

Apéndice 2.6: Simulación de la trayectoria del misil (WZSMO)

```
%=========================================================================
%
% Título            : sim_WZSMO.m
% Propósito         : Simulación de observador de modo deslizante de
%                     Walcott-Zak
%
% Descripción       : Este programa permite la estimación de parámetros de
%                     posición, velocidad y aceleración de un blanco aéreo
%                     de alta maniobrabilidad a partir de las mediciones
%                     ruidosas de posición y velocidad en coordenadas
%                     cartesianas obtenidas por un radar de seguimiento
%                     automático. Para tal efecto, la ganancia lineal es
%                     sintetizada mediante el algoritmo LQG y la ganancia
%                     no lineal es sintetizada mediante desigualdades
%                     lineales matriciales (LMI).
%                     Simulación con ruido = 1.
%                     Simulación sin ruido = 0.
%
% Fecha de creación : 04/08/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
clear all;
clc;
close all;
%-------------------------------------------------------------------------
% Inicialización de variables
%-------------------------------------------------------------------------
RMSEcold = 0;
RMSEpold = 0;
%-------------------------------------------------------------------------
% Parámetros iniciales de radar (sensor)
%-------------------------------------------------------------------------
T     = 1/50;                       % Período de muestreo (seg)
%-------------------------------------------------------------------------
% Definición de constantes físicas
%-------------------------------------------------------------------------
g     = 9.81;                       % gravedad (m/s2)
M     = 343;                        % velocidad del sonido (m/s)
load_factor = 100*g;                % máxima sobre aceleración (g/s)
%-------------------------------------------------------------------------
% Definición de vectores de tiempo
%-------------------------------------------------------------------------
ti    = 0;                          % Tiempo inicial de simulación
tff   = 104.5;                      % Tiempo final de simulación
tt    = ti:T:tff;                   % Vector de tiempo
tt    = tt';                        % Transpuesta del vector de tiempo
nt    = length(tt);                 % Número de muestras
%-------------------------------------------------------------------------
% Definición de posición y velocidad inicial del blanco en
% coordenadas esféricas
%-------------------------------------------------------------------------
Ad_init  = 35000;                       % Distancia al blanco (m)
Ed_init  = 0.01;                        % Elevación al blanco (°)
Bdn_init = 30;                          % Marcación al blanco (°)
Vm_init  = 0.9*M;                       % Velocidad del blanco (1 Mach = 343 m/s)
Rv_init  = 210;                         % Rumbo inicial del blanco (°)
%-------------------------------------------------------------------------
% Conversión de coordenadas esféricas a cartesianas (North-Sky-East) de la
% posición y velocidad inicial del blanco
%-------------------------------------------------------------------------
[Adx_init, Ady_init, Adz_init] = p2c(Ad_init,Ed_init,Bdn_init);
[Vmx_init, Vmy_init, Vmz_init] = p2c(Vm_init,0,Rv_init);
%-------------------------------------------------------------------------
% Inicialización de vector de estados
%-------------------------------------------------------------------------
xk = [Adx_init Vmx_init 0*g Ady_init Vmy_init 0*g Adz_init Vmz_init 0*g]';
%-------------------------------------------------------------------------
% Definición de matrices del modelo de espacio de estados del sistema
%-------------------------------------------------------------------------
% Matriz de medición (Pulse-Doppler radar)
Ck     = [1 0 0 0 0 0 0 0 0;        % pos x
```

```
          0 1 0 0 0 0 0 0 0;        % vel x
          0 0 0 1 0 0 0 0 0;        % pos y
          0 0 0 0 1 0 0 0 0;        % vel y
          0 0 0 0 0 0 1 0 0;        % pos z
          0 0 0 0 0 0 0 1 0];       % vel z

% Cálculo de matriz de transición Fk
Fk = [1 T (T^2)/2 0 0       0 0 0       0;
      0 1       T 0 0       0 0 0       0;
      0 0       1 0 0       0 0 0       0;
      0 0       0 1 T (T^2)/2 0 0       0;
      0 0       0 0 1       T 0 0       0;
      0 0       0 0 0       1 0 0       0;
      0 0       0 0 0       0 1 T (T^2)/2;
      0 0       0 0 0       0 0 1       T;
      0 0       0 0 0       0 0 0       1];

% Cálculo de la matriz de entrada Gk
Gk = [(T^3)/6       0        0;
      (T^2)/2       0        0;
       T            0        0;
       0       (T^3)/6       0;
       0       (T^2)/2       0;
       0             T       0;
       0             0  (T^3)/6;
       0             0  (T^2)/2;
       0             0        T];

% Matriz de distribución de perturbaciones (aceleraciones)
Dk = [75e4*(T^3)/6       0               0;
      5e3*(T^2)/2        0               0;
       1e2*T             0               0;
       0       75e4*(T^3)/6              0;
       0        5e3*(T^2)/2              0;
       0              1e2*T              0;
       0                 0     75e4*(T^3)/6;
       0                 0      5e3*(T^2)/2;
       0                 0           1e2*T];
%-------------------------------------------------------------------------
% Generación del vector de sobre aceleraciones de entrada uk
%-------------------------------------------------------------------------
uk = gen_jerk_sea_skimming3(nt,load_factor,g,T);
%-------------------------------------------------------------------------
% Generación del vector de perturbaciones ?k
%-------------------------------------------------------------------------
pk = perturb_1(g,ti,T,tff);
%-------------------------------------------------------------------------
% Cargar ruido glint
%-------------------------------------------------------------------------
load glint_puntos.mat
%-------------------------------------------------------------------------
% Simulación del modelo
%-------------------------------------------------------------------------
disp('Observador de Walcott-Zak por síntesis LQG/LMI');
prompt = 'Para simular con ruido presione "1". Para simular sin ruido presione "0": ';
gl     = input(prompt);
%-------------------------------------------------------------------------
% Síntesis de ganancias Gl y Gn
%-------------------------------------------------------------------------
[Gl,Gn,Z3,Ao,s0] = WZSMO_sintesis(Fk,Gk,Ck,Dk,gl);
for k = 1:nt

    % Modelo dinámico o de transición de estados
    xk       = Fk*xk + Gk*uk(:,k) + Dk*pk(:,k);
    % Simulación de adquisición de datos por el radar
    % Datos de posición adquiridos en coordenadas esféricas
    [Ad,Bdn,Ed] = c2p(xk);
    % Modelo de medición polar a cartesiano
    yk = yk_noise(Ad,Bdn,Ed,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,gl,k);
    % Observador de modo deslizante Walcott-Zak
    tic
    [xhk]   = WZSMO(Gl,Gn,Z3,Ao,Gk,yk,uk,k,g,s0);
    TCPU = toc;
    % Conversión de coordenada estimadas cartesianas a coordenadas polares
    [Adhat,Bdnhat,Edhat] = c2p(xhk);

    % Cálculo de errores en coordenadas cartesianas
```

```
    errorc       = xhk - xk;

    % Cálculo de errores en coordenadas esféricas
    errorAd      = Adhat - Ad;                                    %mts
    errorBdn     = Bdnhat*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEd      = Edhat*(pi*1000/180) - Ed*(pi*1000/180);
    errorp       = [errorAd; errorBdn; errorEd];

    % Cálculo de RMSE
    if ( k >= 4650)
        RMSEc         = RMSEcold + ((errorc).^2);
        RMSEp         = RMSEpold + ((errorp).^2);
    else
        RMSEc         = 0;
        RMSEp         = 0;
    end
    % Guardar señales
    bin_log(:,k)      = k;
    xk_log(:,k)       = xk;
    yk_log(:,k)       = yk;
    xhk_log(:,k)      = xhk;
    errorc_log(:,k)   = errorc;
    errorp_log(:,k)   = errorp;
    polar_log(:,k)    = [Ad;Bdn;Ed];
    polarhat_log(:,k) = [Adhat;Bdnhat;Edhat];
    TCPU_log(:,k)     = TCPU;
    RMSEcold          = RMSEc;
    RMSEpold          = RMSEp;
end
RMSEctot = sqrt(RMSEc)/(nt-4650);
RMSEptot = sqrt(RMSEp)/(nt-4650);
%-------------------------------------------------------------------------
% Conversión de vector de estados reales a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
% 0.10197 - de mts/seg2 a g's
conv_xk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197]';
xk_log = xk_log.*conv_xk;
%-------------------------------------------------------------------------
% Conversión de vector de estados medidos a Mn, Mach
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
conv_yk   = [ 0.00054 0.00292 0.00054 0.00292 0.00054 0.00292 ]';
yk_log    = yk_log.*conv_yk;
%-------------------------------------------------------------------------
% Conversión de vector de estados estimados a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - mn
% 0.00292 - Mach
% 0.10197 - g's
conv_xhk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292
0.10197]';
xhk_log = xhk_log.*conv_xhk;
%-------------------------------------------------------------------------
% Conversión de vector de entrada real y estimado a g/s
%-------------------------------------------------------------------------
uk_log    = uk./g;
%-------------------------------------------------------------------------
% Conversión de perturbaciones a g/s
%-------------------------------------------------------------------------
pk_log    = pk./g;
%-------------------------------------------------------------------------
% Plots
%-------------------------------------------------------------------------
%% 3D trayectoria completa

figure(1);p1 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                0,                0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                   0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p1(1),'LineWidth',2.0);
```

```
set(p1(2),'LineWidth',1.8);
set(p1(3),'LineWidth',1.5);
set(p1(4),'LineWidth',2.0);
set(p1(5),'LineWidth',2.0);
set(p1(6),'LineWidth',2.0);
set(p1(7),'LineWidth',2.0);

axis([-5 20 -5 10 0 0.04]);          % Toda la trayectoria
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Trayectoria sea skimming del misil','FontSize',16)
legend({'mediciones','real','WZSMO','Posición del buque observador','Posición inicial
del misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
%% 3D trayectoria completa

figure(2);p2 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                  0,                  0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p2(1),'LineWidth',2.0);
set(p2(2),'LineWidth',1.8);
set(p2(3),'LineWidth',1.5);
set(p2(4),'LineWidth',2.0);
set(p2(5),'LineWidth',2.0);
set(p2(6),'LineWidth',2.0);
set(p2(7),'LineWidth',2.0);
axis([4 8 -2 5 0 0.016]);
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Maniobra terminal del misil','FontSize',16)
legend({'mediciones','real','WZSMO','Posición del buque observador','Posición inicial
del misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14);

%% Coordenada x
% Posición x: Trayectoria completa
figure(3);
subplot(1,2,1);
p3 = plot(bin_log(1,:),yk_log(1,:),'-r',bin_log(1,:),xk_log(1,:),'-
g',bin_log(1,:),xhk_log(1,:),'-k');
title('Trayectoria completa del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p3(1),'LineWidth', 2.0);
set(p3(2),'LineWidth', 1.8);
set(p3(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(1,:) yk_log(1,:)]) max([xk_log(1,:) yk_log(1,:)])])

% Posición x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p4 = plot(bin_log(1,lpx1:nt),yk_log(1,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(1,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(1,lpx1:nt),'-
k');
title('Maniobra terminal del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p4(1),'LineWidth', 2.0);
set(p4(2),'LineWidth', 1.8);
set(p4(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)]) max([xk_log(1,lpx1:nt)
yk_log(1,lpx1:nt)])])
```

```
%%
% Velocidad x: Trayectoria completa
figure(4);
subplot(1,2,1);
p5 = plot(bin_log(1,:),yk_log(2,:),'-r',bin_log(1,:),xk_log(2,:),'-
g',bin_log(1,:),xhk_log(2,:),'-k');
title('Trayectoria completa del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p5(1),'LineWidth', 2.0);
set(p5(2),'LineWidth', 1.8);
set(p5(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(2,:) yk_log(2,:)]) max([xk_log(2,:) yk_log(2,:)])])
%axis([1 nt -10 10])
% Velocidad x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p6 = plot(bin_log(1,lpx1:nt),yk_log(2,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(2,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(2,lpx1:nt),'-
k');
title('Maniobra terminal del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p6(1),'LineWidth', 2.0);
set(p6(2),'LineWidth', 1.8);
set(p6(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt -1 2]) % con ruido
%axis([lpx1 nt min([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])
max([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])])
%%
% Aceleración x: Trayectoria completa
figure(5);
subplot(1,2,1);
p7 = plot(bin_log(1,:),xk_log(3,:),'-g',bin_log(1,:),xhk_log(3,:),'-k');
title('Trayectoria completa del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','WZSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p7(1),'LineWidth', 1.8);
set(p7(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(3,:)) max(xk_log(3,:))])
%axis([1 nt min([xk_log(3,:) xhk_log(3,:)]) max([xk_log(3,:) xhk_log(3,:)])])
% Aceleración x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
p8 = plot(bin_log(1,lpx1:nt),xk_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(3,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','WZSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p8(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(3,lpx1:nt) xhk_log(3,lpx1:nt)]) max([xk_log(3,lpx1:nt)
xhk_log(3,lpx1:nt)])])
%axis([lpx1 nt -20 20])
%% Coordenada y
% Posición y: Trayectoria completa
figure(6);
subplot(1,2,1);
p9 = plot(bin_log(1,:),yk_log(3,:),'-r',bin_log(1,:),xk_log(4,:),'-
g',bin_log(1,:),xhk_log(4,:),'-k');
title('Trayectoria completa del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p9(1),'LineWidth', 2.0);
set(p9(2),'LineWidth', 1.8);
set(p9(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(4,:) yk_log(3,:)]) max([xk_log(4,:) yk_log(3,:)])])
```

```
% Posición y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p10 = plot(bin_log(1,lpx1:nt),yk_log(3,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(4,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(4,lpx1:nt),'-
K');
title('Maniobra terminal del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p10(1),'LineWidth', 2.0);
set(p10(2),'LineWidth', 1.8);
set(p10(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])
max([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])])
%%
% Velocidad y: Trayectoria completa
figure(7);
subplot(1,2,1);
p11 = plot(bin_log(1,:),yk_log(4,:),'-r',bin_log(1,:),xk_log(5,:),'-
g',bin_log(1,:),xhk_log(5,:),'-k');
title('Trayectoria completa del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p11(1),'LineWidth', 2.0);
set(p11(2),'LineWidth', 1.8);
set(p11(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(5,:) yk_log(4,:)]) max([xk_log(5,:) yk_log(4,:)])])
%axis([1 nt -10 10])
% Velocidad y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p12 = plot(bin_log(1,lpx1:nt),yk_log(4,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(5,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(5,lpx1:nt),'-
k');
title('Maniobra terminal del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p12(1),'LineWidth', 2.0);
set(p12(2),'LineWidth', 1.8);
set(p12(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(5,lpx1:nt) yk_log(4,lpx1:nt)]) max([xk_log(5,lpx1:nt)
yk_log(4,lpx1:nt)])])
%%
figure(8);
% Aceleración y: Trayectoria
subplot(1,2,1);
p14 = plot(bin_log(1,:),xk_log(6,:),'-g',bin_log(1,:),xhk_log(6,:),'-k');
title('Maniobra terminal del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','WZSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p14(1),'LineWidth', 1.8);
set(p14(2),'LineWidth', 1.5);
grid on
%axis([1 nt min([xk_log(6,:) xhk_log(6,:)]) max([xk_log(6,:) xhk_log(6,:)])])
axis([1 nt min(xk_log(6,:)) max(xk_log(6,:))])
% Aceleración y: Maniobra terminal
subplot(1,2,2);
p13 = plot(bin_log(1,lpx1:nt),xk_log(6,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(6,lpx1:nt),'-k');
title('Trayectoria completa del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','WZSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p13(1),'LineWidth', 1.8);
set(p13(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(6,lpx1:nt) xhk_log(6,lpx1:nt)]) max([xk_log(6,lpx1:nt)
xhk_log(6,lpx1:nt)])])

%% Coordenada z
```

```
% Posición z: Trayectoria completa
figure(9);
subplot(1,2,1);
p15 = plot(bin_log(1,:),yk_log(5,:)./0.00054,'-r',bin_log(1,:),xk_log(7,:)./0.00054,'-
g',bin_log(1,:),xhk_log(7,:)./0.00054,'-k');
title('Trayectoria completa del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p15(1),'LineWidth', 2.0);
set(p15(2),'LineWidth', 1.8);
set(p15(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(7,:)./0.00054 yk_log(5,:)./0.00054]) max([xk_log(7,:)./0.00054
yk_log(5,:)./0.00054])])

% Posición z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p16 = plot(bin_log(1,lpx1:nt),yk_log(5,lpx1:nt)./0.00054,'-
r',bin_log(1,lpx1:nt),xk_log(7,lpx1:nt)./0.00054,'-
g',bin_log(1,lpx1:nt),xhk_log(7,lpx1:nt)./0.00054,'-k');
title('Maniobra terminal del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mts]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p16(1),'LineWidth', 2.0);
set(p16(2),'LineWidth', 1.8);
set(p16(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054]) max([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054])])
%%
% Velocidad z: Trayectoria completa
figure(10);
subplot(1,2,1);
p17 = plot(bin_log(1,:),yk_log(6,:),'-r',bin_log(1,:),xk_log(8,:),'-
g',bin_log(1,:),xhk_log(8,:),'-k');
title('Trayectoria completa del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p17(1),'LineWidth', 2.0);
set(p17(2),'LineWidth', 1.8);
set(p17(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(8,:) yk_log(6,:)]) max([xk_log(8,:) yk_log(6,:)])])
%axis([1 nt -4 4])
% Velocidad z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p18 = plot(bin_log(1,lpx1:nt),yk_log(6,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(8,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(8,lpx1:nt),'-
k');
title('Maniobra terminal del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'medición','real','WZSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p18(1),'LineWidth', 1.0);
set(p18(2),'LineWidth', 1.8);
set(p18(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(8,lpx1:nt) yk_log(6,lpx1:nt)]) max([xk_log(8,lpx1:nt)
yk_log(6,lpx1:nt)])])
%%
% Aceleración z: Trayectoria completa
figure(11);
subplot(1,2,1);
p19 = plot(bin_log(1,:),xk_log(9,:),'-g',bin_log(1,:),xhk_log(9,:),'-k');
title('Trayectoria completa del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','WZSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p19(1),'LineWidth', 1.8);
set(p19(2),'LineWidth', 1.5);
grid on
```

```
axis([1 nt min(xk_log(9,:)) max(xk_log(9,:))])
%axis([1 nt min([xk_log(9,:) xhk_log(9,:)]) max([xk_log(9,:) xhk_log(9,:)])])

% Aceleración z: Maniobra terminal
subplot(1,2,2);
p20 = plot(bin_log(1,lpx1:nt),xk_log(9,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(9,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','WZSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p20(1),'LineWidth', 1.8);
set(p20(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(9,lpx1:nt) xhk_log(9,lpx1:nt)]) max([xk_log(9,lpx1:nt)
xhk_log(9,lpx1:nt)])])

%% Plots señales de errores cartesianos posición x,y,z
figure(12);
sgtitle('Errores cartesianos','FontSize',18)
lpx1 = 4500;
subplot(3,2,1);
p21 = plot(bin_log(1,:),errorc_log(1,:),'-g',bin_log(1,:),errorc_log(4,:),'-
c',bin_log(1,:),errorc_log(7,:),'-m');
title('Trayectoria completa: posición','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m]','FontSize',16)
legend({'pos x','pos y','pos z'},'Location','north','NumColumns',3,'FontSize',14)
set(p21,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(1,:) errorc_log(4,:) errorc_log(7,:)]) max([errorc_log(1,:)
errorc_log(4,:) errorc_log(7,:)])])

subplot(3,2,2);
p22 = plot(bin_log(1,lpx1:nt),errorc_log(1,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(4,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(7,lpx1:nt),'-m');
title('Maniobra terminal: posición','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p22,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])
max([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])])
% Plots señales de errores cartesianos velocidad x,y,z
subplot(3,2,3);
p23 = plot(bin_log(1,:),errorc_log(2,:),'-g',bin_log(1,:),errorc_log(5,:),'-
c',bin_log(1,:),errorc_log(8,:),'-m');
title('Trayectoria completa: velocidad','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s]','FontSize',16)
legend({'vel x','vel y','vel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p23,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(2,:) errorc_log(5,:) errorc_log(8,:)]) max([errorc_log(2,:)
errorc_log(5,:) errorc_log(8,:)])])

subplot(3,2,4);
p24 = plot(bin_log(1,lpx1:nt),errorc_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(5,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(8,lpx1:nt),'-m');
title('Maniobra terminal: velocidad','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s]','FontSize',16);
set(p24,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])
max([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])])
% Plots señales de errores cartesianos aceleración x,y,z
subplot(3,2,5);
p25 = plot(bin_log(1,:),errorc_log(3,:),'-g',bin_log(1,:),errorc_log(6,:),'-
c',bin_log(1,:),errorc_log(9,:),'-m');
title('Trayectoria completa: aceleración','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s2]','FontSize',16)
legend({'acel x','acel y','acel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p25,'LineWidth',1.5);
```

```
grid on
axis([1 nt min([errorc_log(3,:) errorc_log(6,:) errorc_log(9,:)]) max([errorc_log(3,:)
errorc_log(6,:) errorc_log(9,:)])])

subplot(3,2,6);
p26 = plot(bin_log(1,lpx1:nt),errorc_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(6,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(9,lpx1:nt),'-m');
title('Maniobra terminal: aceleración','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s2]','FontSize',16);
set(p26,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])
max([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])])

%% Plots señales de error polares: Distancia Ad
figure(13);
sgtitle('Errores polares','FontSize',18)
maxAd =  37.5*ones(1,nt);
minAd = -37.5*ones(1,nt);

subplot(3,2,1);
laz1 = 4500;
p27 = plot(bin_log(1,:),errorp_log(1,:),'-g',bin_log(1,:),maxAd(1,:),'--
r',bin_log(1,:),minAd(1,:),'--r');
title('Trayectoria completa: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
legend({'Ad','Límite +/- 37.5 mts'},'NumColumns',2,'Location','North','FontSize',12);
set(p27(1),'LineWidth',1.5);
set(p27(2),'LineWidth',1.0);
set(p27(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(1,:) minAd(1,:)]) max([errorp_log(1,:) maxAd(1,:)])])

subplot(3,2,2);
p28 = plot(bin_log(1,laz1:nt),errorp_log(1,laz1:nt),'-
g',bin_log(1,laz1:nt),maxAd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minAd(1,laz1:nt),'--r');
title('Maniobra terminal: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p28(1),'LineWidth',1.5);
set(p28(2),'LineWidth',1.0);
set(p28(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(1,laz1:nt) minAd(1,laz1:nt)]) max([errorp_log(1,laz1:nt)
maxAd(1,laz1:nt)])])
% Error polar en acimut
maxBdn = 2*ones(1,nt);
minBdn = -2*ones(1,nt);

subplot(3,2,3);
laz1 = 4500;
p29 = plot(bin_log(1,:),errorp_log(2,:),'-g',bin_log(1,:),maxBdn(1,:),'--
r',bin_log(1,:),minBdn(1,:),'--r');
title('Trayectoria completa: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Bdn','Límite +/- 2mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p29(1),'LineWidth',1.5);
set(p29(2),'LineWidth',1.0);
set(p29(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(2,:) minBdn(1,:)]) max([errorp_log(2,:) maxBdn(1,:)])])

subplot(3,2,4);
p30 = plot(bin_log(1,laz1:nt),errorp_log(2,laz1:nt),'-
g',bin_log(1,laz1:nt),maxBdn(1,laz1:nt),'--r',bin_log(1,laz1:nt),minBdn(1,laz1:nt),'--
r');
title('Maniobra terminal: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p30(1),'LineWidth',1.5);
set(p30(2),'LineWidth',1.0);
set(p30(3),'LineWidth',1.0);
grid on
```

```
axis([laz1 nt  min([errorp_log(2,laz1:nt) minBdn(1,laz1:nt)]) max([errorp_log(2,laz1:nt)
maxBdn(1,laz1:nt)])])

maxEd = 2*ones(1,nt);
minEd = -2*ones(1,nt);

subplot(3,2,5);
laz1 = 4500;
p31 = plot(bin_log(1,:),errorp_log(3,:),'-g',bin_log(1,:),maxEd(1,:),'--
r',bin_log(1,:),minEd(1,:),'--r');
title('Trayectoria completa: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Ed','Límite +/- 2mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p31(1),'LineWidth',1.5);
set(p31(2),'LineWidth',1.0);
set(p31(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(3,:) minEd(1,:)]) max([errorp_log(3,:) maxEd(1,:)])])

subplot(3,2,6);
p32 = plot(bin_log(1,laz1:nt),errorp_log(3,laz1:nt),'-
g',bin_log(1,laz1:nt),maxEd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minEd(1,laz1:nt),'--r');
title('Maniobra terminal: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p32(1),'LineWidth',1.5);
set(p32(2),'LineWidth',1.0);
set(p32(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(3,laz1:nt) minEd(1,laz1:nt)]) max([errorp_log(3,laz1:nt)
maxEd(1,laz1:nt)])])

save WZSMO_datos_simulacion.mat
```

Apéndice 2.7: Observador HOSMO

```
function [xhk,phk,sigk] = HOSMO(yk,uk,Fk,Gk,Dk,T)
%=========================================================================
%
% Título            : HOSMO.m
% Propósito         : Observador de modo deslizante super-twisting de orden
%                     superior (HOSMO)
%
% Descripción       : Este programa implementa el observador de modo desli-
%                     zante super-twisting de orden superior (HOSMO), el
%                     cual efectúa la estimación de parámetros de posición
%                     velocidad, aceleración y perturbación por medio de
%                     mediciones de posición.
%
%                     Modelo Discreto para la coordenada x (coordenada y,z similar):
%
%                     x1k_1  = Fk(1,1)*x1k + Fk(1,2)*x2k + Fk(1,3)*x3k + Gk(1,1)*u1k +
Dk(1,1)*p1k
%                     x2k_1  = Fk(2,2)*x2k + Fk(2,3)*x3k + Gk(2,1)*u1k + Dk(2,1)*p1k
%                     x3k_1  = Fk(3,3)*x3k + Gk(3,1)*u1k + Dk(3,1)*p1k
%
%                     Observador Super-Twisting de Orden Superior para la
%                     coordenada x (similar para coordenada y,z)
%
%                     x1hatk_1  = Fk(1,1)*x1hatk + Fk(1,2)*x2hatk + Fk(1,3)*x3hatk +
Gk(1,1)*u1k + Dk(1,1)*x4hatk + z1xk
%                     x2hatk_1  = Fk(2,2)*x2hatk + Fk(2,3)*x3hatk + Gk(2,1)*u1k +
Dk(2,1)*x4hatk + z2xk
%                     x3hatk_1  = Fk(3,3)*x3hatk + Gk(3,1)*u1k + Dk(3,1)*x4hatk + z3xk
%                     x4hatk_1  = x4hatk + z4xk*T;
%
%                     donde,
%                     x1hatk - posición estimada
%                     x2hatk - velocidad estimada
%                     x3hatk - aceleración estimada
%                     x4hatk - perturbación estimada
%                     u1k    - señal de control
%                     z1xk, z2xk, z3xk y z4xk son los términos de corrección super
twisting
%
```

```
%
% Fecha de creación : 20/11/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
%-------------------------------------------------------------------------
% Persistencia de variables del observador
%-------------------------------------------------------------------------
% x1hatk  - posición estimada coordenada x
% x2hatk  - velocidad estimada coordenada x
% x3hatk  - aceleración estimada coordenada x
% p1hatk  - perturbación estimada coordenada x
% x5hatk  - posición estimada coordenada y
% x6hatk  - velocidad estimada coordenada y
% x7hatk  - aceleración estimada coordenada y
% p2hatk  - perturbación estimada coordenada y
% x9hatk  - posición estimada coordenada z
% x10hatk - velocidad estimada coordenada z
% x11hatk - aceleración estimada coordenada z
% p3hatk  - perturbación estimada coordenada z

persistent x1hatk x2hatk x3hatk x4hatk x5hatk x6hatk x7hatk x8hatk x9hatk p1hatk p2hatk
p3hatk

if isempty (x1hatk)
   x1hatk = 35000;
   x2hatk = -200;
   x3hatk = 0;
   x4hatk = 25000;
   x5hatk = -160;
   x6hatk = 0;
   x7hatk = 20;
   x8hatk = 20;
   x9hatk = 0;
   p1hatk = 0;
   p2hatk = 0;
   p3hatk = 0;
end
%-------------------------------------------------------------------------
% Definición de superficies deslizantes (posición)
%-------------------------------------------------------------------------
sigx        = yk(1,1) - x1hatk;
sigy        = yk(3,1) - x4hatk;
sigz        = yk(5,1) - x7hatk;
%-------------------------------------------------------------------------
% Definición de superficies deslizantes (velocidad)
%-------------------------------------------------------------------------
% Estas superficies deslizantes no intervienen en el algoritmo. Solo serán
% utilizadas para demostrar graficamente la torsión generada por el
% algoritmo.
sigdx       = yk(2,1) - x2hatk;
sigdy       = yk(4,1) - x5hatk;
sigdz       = yk(6,1) - x8hatk;

sigk        = [sigx sigy sigz sigdx sigdy sigdz]';
%-------------------------------------------------------------------------
% Ganancias del algoritmo Super-Twisting de orden superior
%-------------------------------------------------------------------------
% Coordenada x
c1x  =  1.9*1.0;
c2x  =  1.5*1.1;
c3x  =  3.9*1.5;
k1x  =  c1x*0.6^(1/5);
k2x  = (c2x*(c1x)^(2/3))*0.6^(2/5);
k3x  = (c3x*((c2x*(c1x)^(2/3)))^(1/2))*0.6^(3/5);
k4x  =  496*0.6*0.6^(4/5);
% Coordenada y
c1y  = 1.2*1.0;
c2y  = 1.2*1.1;
c3y  = 1.2*1.5;
k1y  = c1y*0.6^(1/5);
k2y  = (c2y*(c1y)^(2/3))*0.6^(2/5);
k3y  = (c3y*((c2y*(c1y)^(2/3)))^(1/2))*0.6^(3/5);
```

```
k4y  = 10*0.6*0.6^(4/5);
% Coordenada z
c1z  = 1.3*1.0;
c2z  = 1.3*1.1;
c3z  = 1.3*1.5;
k1z  = c1z*0.6^(1/5);
k2z  = (c2z*(c1z)^(2/3))*0.6^(2/5);
k3z  = (c3z*((c2z*(c1z)^(2/3)))^(1/2))*0.6^(3/5);
k4z  = 35*0.6*0.6^(4/5);
%-------------------------------------------------------------------------
% Términos de compensación del algoritmo Super-Twisting de orden superior
%-------------------------------------------------------------------------
% Coordenada x
z1xk    = k1x*(abs(sigx)^(4/5))*sign(sigx);
z2xk    = k2x*(abs(sigx)^(3/4))*sign(sigx);
z3xk    = k3x*(abs(sigx)^(1/2))*sign(sigx);
z4xk    = k4x*sign(sigx);
% Coordenada y
z1yk    = k1y*(abs(sigy)^(4/5))*sign(sigy);
z2yk    = k2y*(abs(sigy)^(3/4))*sign(sigy);
z3yk    = k3y*(abs(sigy)^(1/2))*sign(sigy);
z4yk    = k4y*sign(sigy);
% Coordenada z
z1zk    = k1z*(abs(sigz)^(4/5))*sign(sigz);
z2zk    = k2z*(abs(sigz)^(3/4))*sign(sigz);
z3zk    = k3z*(abs(sigz)^(1/2))*sign(sigz);
z4zk    = k4z*sign(sigz);
%-------------------------------------------------------------------------
% Observador de modo deslizante Super-Twisting de Orden Superior (HOSMO)
%-------------------------------------------------------------------------
% Observador coordenada x
x1hatk_1  = Fk(1,1)*x1hatk + Fk(1,2)*x2hatk + Fk(1,3)*x3hatk + Gk(1,1)*uk(1,1) +
Dk(1,1)*p1hatk + z1xk;
x2hatk_1  = Fk(2,2)*x2hatk + Fk(2,3)*x3hatk + Gk(2,1)*uk(1,1) + Dk(2,1)*p1hatk + z2xk;
x3hatk_1  = Fk(3,3)*x3hatk + Gk(3,1)*uk(1,1) + Dk(3,1)*p1hatk + z3xk;
p1hatk_1  = p1hatk + z4xk*T;

% Observador coordenada y
x4hatk_1  = Fk(4,4)*x4hatk + Fk(4,5)*x5hatk + Fk(4,6)*x6hatk + Gk(4,2)*uk(2,1) +
Dk(4,2)*p2hatk + z1yk;
x5hatk_1  = Fk(5,5)*x5hatk + Fk(5,6)*x6hatk + Gk(5,2)*uk(2,1) + Dk(5,2)*p2hatk + z2yk;
x6hatk_1  = Fk(6,6)*x6hatk + Gk(6,2)*uk(2,1) + Dk(6,2)*p2hatk + z3yk;
p2hatk_1  = p2hatk + z4yk*T;

% Observador coordenada z
x7hatk_1  = Fk(7,7)*x7hatk + Fk(7,8)*x8hatk + Fk(7,9)*x9hatk + Gk(7,3)*uk(3,1) +
Dk(7,3)*p3hatk + z1zk;
x8hatk_1  = Fk(8,8)*x8hatk + Fk(8,9)*x9hatk + Gk(8,3)*uk(3,1) + Dk(8,3)*p3hatk + z2zk;
x9hatk_1  = Fk(9,9)*x9hatk + Gk(9,3)*uk(3,1) + Dk(9,3)*p3hatk + z3zk;
p3hatk_1  = p3hatk + z4zk*T;
%-------------------------------------------------------------------------
% Vector de estados estimados
%-------------------------------------------------------------------------
xhk = [x1hatk x2hatk x3hatk x4hatk x5hatk x6hatk x7hatk x8hatk x9hatk]';
%-------------------------------------------------------------------------
% Vector de perturbaciones estimadas
%-------------------------------------------------------------------------
phk = [p1hatk p2hatk p3hatk]';
%-------------------------------------------------------------------------
% Guardar variables para la siguiente iteración
%-------------------------------------------------------------------------
x1hatk   = x1hatk_1;
x2hatk   = x2hatk_1;
x3hatk   = x3hatk_1;
x4hatk   = x4hatk_1;
x5hatk   = x5hatk_1;
x6hatk   = x6hatk_1;
x7hatk   = x7hatk_1;
x8hatk   = x8hatk_1;
x9hatk   = x9hatk_1;
p1hatk   = p1hatk_1;
p2hatk   = p2hatk_1;
p3hatk   = p3hatk_1;
end
```

Apéndice 2.8: Simulación de la trayectoria del misil (HOSMO)

```
%=========================================================================
%
% Título            : sim_HOSMO.m
% Propósito         : Simulación de observador de modo deslizante
%                     super-twisting de orden superior
%
% Descripción       : Este programa permite la estimación de parámetros de
%                     posición, velocidad y aceleración de un blanco aéreo
%                     de alta maniobrabilidad a partir de mediciones
%                     ruidosas de posición obtenidas por un radar de
%                     seguimiento automático. Para tal efecto, utiliza el
%                     algoritmo super-twisting de orden superior.
%                     Simulación con ruido = 1.
%                     Simulación sin ruido = 0.
%
% Fecha de creación : 20/11/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
clear all;
clc;
close all;
%-------------------------------------------------------------------------
% Inicialización de variables
%-------------------------------------------------------------------------
RMSEcold = 0;
RMSEpold = 0;
%-------------------------------------------------------------------------
% Parámetros iniciales de radar (sensor)
%-------------------------------------------------------------------------
T     = 1/50;                       % Período de muestreo (seg)
%-------------------------------------------------------------------------
% Definición de constantes físicas
%-------------------------------------------------------------------------
g     = 9.81;                       % gravedad (m/s2)
M     = 343;                        % velocidad del sonido (m/s)
load_factor = 100*g;                % máxima sobre aceleración (g/s)
%-------------------------------------------------------------------------
% Definición de vectores de tiempo
%-------------------------------------------------------------------------
ti    = 0;                          % Tiempo inicial de simulación
tff   = 104.5;                      % Tiempo final de simulación
tt    = ti:T:tff;                   % Vector de tiempo
tt    = tt';                        % Transpuesta del vector de tiempo
nt    = length(tt);                 % Número de muestras
%-------------------------------------------------------------------------
% Definición de posición y velocidad inicial del blanco en
% coordenadas esféricas
%-------------------------------------------------------------------------
Ad_init  = 35000;                       % Distancia al blanco (m)
Ed_init  = 0.01;                        % Elevación al blanco (°)
Bdn_init = 30;                          % Marcación al blanco (°)
Vm_init  = 0.9*M;                       % Velocidad del blanco (1 Mach = 343 m/s)
Rv_init  = 210;                         % Rumbo inicial del blanco (°)
%-------------------------------------------------------------------------
% Conversión de coordenadas esféricas a cartesianas (North-Sky-East) de la
% posición y velocidad inicial del blanco
%-------------------------------------------------------------------------
[Adx_init, Ady_init, Adz_init] = p2c(Ad_init,Ed_init,Bdn_init);
[Vmx_init, Vmy_init, Vmz_init] = p2c(Vm_init,0,Rv_init);
%-------------------------------------------------------------------------
% Inicialización de vector de estados
%-------------------------------------------------------------------------
xk = [Adx_init Vmx_init 0*g Ady_init Vmy_init 0*g Adz_init Vmz_init 0*g]';
%-------------------------------------------------------------------------
% Definición de matrices del modelo de espacio de estados del sistema
%-------------------------------------------------------------------------
% Matriz de medición (Pulse radar)
Ck     = [1 0 0 0 0 0 0 0 0;        % pos x
          0 0 0 0 0 0 0 0 0;        % vel x
          0 0 0 1 0 0 0 0 0;        % pos y
          0 0 0 0 0 0 0 0 0;        % vel y
```

```
          0 0 0 0 0 0 1 0 0;        % pos z
          0 0 0 0 0 0 0 0 0];       % vel z

% Cálculo de matriz de transición Fk
Fk = [1 T (T^2)/2 0 0       0 0 0       0;
      0 1       T 0 0       0 0 0       0;
      0 0       1 0 0       0 0 0       0;
      0 0       0 1 T (T^2)/2 0 0       0;
      0 0       0 0 1       T 0 0       0;
      0 0       0 0 0       1 0 0       0;
      0 0       0 0 0       0 1 T (T^2)/2;
      0 0       0 0 0       0 0 1       T;
      0 0       0 0 0       0 0 0       1];

% Cálculo de la matriz de entrada Gk
Gk = [(T^3)/6       0        0;
      (T^2)/2       0        0;
       T            0        0;
       0       (T^3)/6       0;
       0       (T^2)/2       0;
       0            T        0;
       0            0  (T^3)/6;
       0            0  (T^2)/2;
       0            0        T];

% Matriz de distribución de perturbaciones (aceleraciones)
Dk = [75e4*(T^3)/6       0                0;
      5e3*(T^2)/2        0                0;
       1e2*T             0                0;
       0       75e4*(T^3)/6                0;
       0        5e3*(T^2)/2                0;
       0             1e2*T                 0;
       0                0     75e4*(T^3)/6;
       0                0      5e3*(T^2)/2;
       0                0           1e2*T];
%-------------------------------------------------------------------------
% Generación del vector de sobre aceleraciones de entrada uk
%-------------------------------------------------------------------------
uk = gen_jerk_sea_skimming3(nt,load_factor,g,T);
%-------------------------------------------------------------------------
% Generación del vector de perturbaciones ?k
%-------------------------------------------------------------------------
pk = perturb_1(g,ti,T,tff);
%-------------------------------------------------------------------------
% Cargar ruido glint
%-------------------------------------------------------------------------
load glint_puntos.mat
%-------------------------------------------------------------------------
% Simulación del observador
%-------------------------------------------------------------------------
disp('Observador de modo deslizante Super-Twisting de orden superior (HOSMO)');
prompt = 'Para simular con ruido presione "1". Para simular sin ruido presione "0": ';
gl     = input(prompt);
for k = 1:nt

    % Modelo dinámico o de transición de estados
    xk       = Fk*xk + Gk*uk(:,k) + Dk*pk(:,k);
    % Simulación de adquisición de datos por el radar
    % Datos de posición adquiridos en coordenadas esféricas
    [Ad,Bdn,Ed] = c2p(xk);
    % Modelo de medición polar a cartesiano
    yk = yk_noise(Ad,Bdn,Ed,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,gl,k);
    % Observador de modo deslizante Super-Twisting de Orden Superior
    tic
    [xhk,phk,sigk] = HOSMO(yk,uk,Fk,Gk,Dk,T);
    TCPU = toc;

    % Conversión de coordenada estimadas cartesianas a coordenadas polares
    [Adhat,Bdnhat,Edhat] = c2p(xhk);

    % Cálculo de errores en coordenadas cartesianas
    errorc       = xhk - xk;

    % Cálculo de errores en coordenadas esféricas
    errorAd      = Adhat - Ad;                                  %mts
    errorBdn     = Bdnhat*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEd      = Edhat*(pi*1000/180) - Ed*(pi*1000/180);
```

```
    errorp       = [errorAd; errorBdn; errorEd];

   % Cálculo de RMSE
    if ( k >= 4650)
        RMSEc          = RMSEcold + ((errorc).^2);
        RMSEp          = RMSEpold + ((errorp).^2);
    else
        RMSEc          = 0;
        RMSEp          = 0;
    end

    % Guardar señales
    bin_log(:,k)       = k;
    xk_log(:,k)        = xk;
    xhk_log(:,k)       = xhk;
    phk_log(:,k)       = phk;
    errorc_log(:,k)    = errorc;
    errorp_log(:,k)    = errorp;
    polar_log(:,k)     = [Ad;Bdn;Ed];
    polarhat_log(:,k)  = [Adhat;Bdnhat;Edhat];
    yk_log(:,k)        = yk;
    sigk_log(:,k)      = sigk;
    TCPU_log(:,k)      = TCPU;
    % Age variables
    RMSEcold           = RMSEc;
    RMSEpold           = RMSEp;
end
RMSEctot = sqrt(RMSEc)/(nt-4650);
RMSEptot = sqrt(RMSEp)/(nt-4650);
%-------------------------------------------------------------------------
% Conversión de vector de estados reales a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
% 0.10197 - de mts/seg2 a g's
conv_xk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197]';
xk_log = xk_log.*conv_xk;
%-------------------------------------------------------------------------
% Conversión de vector de estados medidos a Mn, Mach
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
conv_yk   = [ 0.00054 0.00292 0.00054 0.00292 0.00054 0.00292 ]';
yk_log    = yk_log.*conv_yk;
%-------------------------------------------------------------------------
% Conversión de vector de estados estimados a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - mn
% 0.00292 - Mach
% 0.10197 - g's
conv_xhk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292
0.10197]';
xhk_log = xhk_log.*conv_xhk;
%-------------------------------------------------------------------------
% Conversión de vector de entrada real y estimado a g/s
%-------------------------------------------------------------------------
uk_log    = uk./g;
%-------------------------------------------------------------------------
% Conversión de perturbaciones a g/s
%-------------------------------------------------------------------------
pk_log    = pk./g;
phk_log   = phk_log./g;
%-------------------------------------------------------------------------
% Plots
%-------------------------------------------------------------------------
%% 3D trayectoria completa

figure(1);p1 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                0,                0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                   0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p1(1),'LineWidth',2.0);
set(p1(2),'LineWidth',1.8);
```

```
set(p1(3),'LineWidth',1.5);
set(p1(4),'LineWidth',2.0);
set(p1(5),'LineWidth',2.0);
set(p1(6),'LineWidth',2.0);
set(p1(7),'LineWidth',2.0);

axis([-5 20 -5 10 0 0.04]);          % Toda la trayectoria
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Trayectoria sea skimming del misil','FontSize',16)
legend({'mediciones','real','HOSMO','Posición del buque observador','Posición inicial
del misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
%% 3D trayectoria completa

figure(2);p2 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                  0,                  0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p2(1),'LineWidth',2.0);
set(p2(2),'LineWidth',1.8);
set(p2(3),'LineWidth',1.5);
set(p2(4),'LineWidth',2.0);
set(p2(5),'LineWidth',2.0);
set(p2(6),'LineWidth',2.0);
set(p2(7),'LineWidth',2.0);
axis([4 8 -2 5 0 0.016]);
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Maniobra terminal del misil','FontSize',16)
legend({'mediciones','real','HOSMO','Posición del buque observador','Posición inicial
del misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14);

%% Coordenada x
% Posición x: Trayectoria completa
figure(3);
subplot(1,2,1);
p3 = plot(bin_log(1,:),yk_log(1,:),'-r',bin_log(1,:),xk_log(1,:),'-
g',bin_log(1,:),xhk_log(1,:),'-k');
title('Trayectoria completa del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p3(1),'LineWidth', 2.0);
set(p3(2),'LineWidth', 1.8);
set(p3(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(1,:) yk_log(1,:)]) max([xk_log(1,:) yk_log(1,:)])])

% Posición x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p4 = plot(bin_log(1,lpx1:nt),yk_log(1,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(1,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(1,lpx1:nt),'-
k');
title('Maniobra terminal del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p4(1),'LineWidth', 2.0);
set(p4(2),'LineWidth', 1.8);
set(p4(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)]) max([xk_log(1,lpx1:nt)
yk_log(1,lpx1:nt)])])
%%
```

```
% Velocidad x: Trayectoria completa
figure(4);
subplot(1,2,1);
p5 = plot(bin_log(1,:),xk_log(2,:),'-g',bin_log(1,:),xhk_log(2,:),'-k');
title('Trayectoria completa del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p5(1),'LineWidth', 1.8);
set(p5(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(2,:)) max(xk_log(2,:))])
%axis([1 nt -10 10])
% Velocidad x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p6 = plot(bin_log(1,lpx1:nt),xk_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(2,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p6(1),'LineWidth', 1.8);
set(p6(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt -1 2]) % con ruido
%axis([lpx1 nt min([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])
max([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])])
%%
% Aceleración x: Trayectoria completa
figure(5);
subplot(1,2,1);
p7 = plot(bin_log(1,:),xk_log(3,:),'-g',bin_log(1,:),xhk_log(3,:),'-k');
title('Trayectoria completa del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p7(1),'LineWidth', 1.8);
set(p7(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(3,:)) max(xk_log(3,:))])
%axis([1 nt min([xk_log(3,:) xhk_log(3,:)]) max([xk_log(3,:) xhk_log(3,:)])])
% Aceleración x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
p8 = plot(bin_log(1,lpx1:nt),xk_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(3,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p8(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(3,lpx1:nt) xhk_log(3,lpx1:nt)]) max([xk_log(3,lpx1:nt)
xhk_log(3,lpx1:nt)])])
%axis([lpx1 nt -20 20])
%% Coordenada y
% Posición y: Trayectoria completa
figure(6);
subplot(1,2,1);
p9 = plot(bin_log(1,:),yk_log(3,:),'-r',bin_log(1,:),xk_log(4,:),'-
g',bin_log(1,:),xhk_log(4,:),'-k');
title('Trayectoria completa del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p9(1),'LineWidth', 2.0);
set(p9(2),'LineWidth', 1.8);
set(p9(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(4,:) yk_log(3,:)]) max([xk_log(4,:) yk_log(3,:)])])

% Posición y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
```

```
p10 = plot(bin_log(1,lpx1:nt),yk_log(3,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(4,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(4,lpx1:nt),'-
K');
title('Maniobra terminal del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p10(1),'LineWidth', 2.0);
set(p10(2),'LineWidth', 1.8);
set(p10(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])
max([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])])
%%
% Velocidad y: Trayectoria completa
figure(7);
subplot(1,2,1);
p11 = plot(bin_log(1,:),xk_log(5,:),'-g',bin_log(1,:),xhk_log(5,:),'-k');
title('Trayectoria completa del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','HOSMO'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p11(1),'LineWidth', 1.8);
set(p11(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(5,:)) max(xk_log(5,:))])
%axis([1 nt -10 10])
% Velocidad y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p12 = plot(bin_log(1,lpx1:nt),xk_log(5,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(5,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p12(1),'LineWidth', 1.8);
set(p12(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min(xk_log(5,lpx1:nt)) max(xk_log(5,lpx1:nt))])
%%
figure(8);
% Aceleración y: Trayectoria
subplot(1,2,1);
p14 = plot(bin_log(1,:),xk_log(6,:),'-g',bin_log(1,:),xhk_log(6,:),'-k');
title('Maniobra terminal del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p14(1),'LineWidth', 1.8);
set(p14(2),'LineWidth', 1.5);
grid on
%axis([1 nt min([xk_log(6,:) xhk_log(6,:)]) max([xk_log(6,:) xhk_log(6,:)])])
axis([1 nt min(xk_log(6,:)) max(xk_log(6,:))])
% Aceleración y: Maniobra terminal
subplot(1,2,2);
p13 = plot(bin_log(1,lpx1:nt),xk_log(6,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(6,lpx1:nt),'-k');
title('Trayectoria completa del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p13(1),'LineWidth', 1.8);
set(p13(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(6,lpx1:nt) xhk_log(6,lpx1:nt)]) max([xk_log(6,lpx1:nt)
xhk_log(6,lpx1:nt)])])

%% Coordenada z
% Posición z: Trayectoria completa
figure(9);
subplot(1,2,1);
p15 = plot(bin_log(1,:),yk_log(5,:)./0.00054,'-r',bin_log(1,:),xk_log(7,:)./0.00054,'-
g',bin_log(1,:),xhk_log(7,:)./0.00054,'-k');
title('Trayectoria completa del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
```

```
legend({'medición','real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p15(1),'LineWidth', 2.0);
set(p15(2),'LineWidth', 1.8);
set(p15(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(7,:)./0.00054 yk_log(5,:)./0.00054]) max([xk_log(7,:)./0.00054
yk_log(5,:)./0.00054])])

% Posición z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p16 = plot(bin_log(1,lpx1:nt),yk_log(5,lpx1:nt)./0.00054,'-
r',bin_log(1,lpx1:nt),xk_log(7,lpx1:nt)./0.00054,'-
g',bin_log(1,lpx1:nt),xhk_log(7,lpx1:nt)./0.00054,'-k');
title('Maniobra terminal del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mts]','FontSize',22)
legend({'medición','real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p16(1),'LineWidth', 2.0);
set(p16(2),'LineWidth', 1.8);
set(p16(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054]) max([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054])])
%%
% Velocidad z: Trayectoria completa
figure(10);
subplot(1,2,1);
p17 = plot(bin_log(1,:),xk_log(8,:),'-g',bin_log(1,:),xhk_log(8,:),'-k');
title('Trayectoria completa del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','HOSMO'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p17(1),'LineWidth', 1.8);
set(p17(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(8,:)) max(xk_log(8,:))])
%axis([1 nt -4 4])
% Velocidad z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p18 = plot(bin_log(1,lpx1:nt),xk_log(8,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(8,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',3,'FontSize',18)
set(p18(1),'LineWidth', 1.8);
set(p18(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min(xk_log(8,lpx1:nt)) max(xk_log(8,lpx1:nt))])
%%
% Aceleración z: Trayectoria completa
figure(11);
subplot(1,2,1);
p19 = plot(bin_log(1,:),xk_log(9,:),'-g',bin_log(1,:),xhk_log(9,:),'-k');
title('Trayectoria completa del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p19(1),'LineWidth', 1.8);
set(p19(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(9,:)) max(xk_log(9,:))])
%axis([1 nt min([xk_log(9,:) xhk_log(9,:)]) max([xk_log(9,:) xhk_log(9,:)])])

% Aceleración z: Maniobra terminal
subplot(1,2,2);
p20 = plot(bin_log(1,lpx1:nt),xk_log(9,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(9,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','HOSMO'},'Location','north','NumColumns',2,'FontSize',18)
set(p20(1),'LineWidth', 1.8);
set(p20(2),'LineWidth', 1.5);
```

```
grid on
axis([lpx1 nt min([xk_log(9,lpx1:nt) xhk_log(9,lpx1:nt)]) max([xk_log(9,lpx1:nt)
xhk_log(9,lpx1:nt)])])

%% Plots señales de errores cartesianos posición x,y,z
figure(12);
sgtitle('Errores cartesianos','FontSize',18)
lpx1 = 4500;
subplot(3,2,1);
p21 = plot(bin_log(1,:),errorc_log(1,:),'-g',bin_log(1,:),errorc_log(4,:),'-
c',bin_log(1,:),errorc_log(7,:),'-m');
title('Trayectoria completa: posición','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m]','FontSize',16)
legend({'pos x','pos y','pos z'},'Location','north','NumColumns',3,'FontSize',14)
set(p21,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(1,:) errorc_log(4,:) errorc_log(7,:)]) max([errorc_log(1,:)
errorc_log(4,:) errorc_log(7,:)])])

subplot(3,2,2);
p22 = plot(bin_log(1,lpx1:nt),errorc_log(1,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(4,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(7,lpx1:nt),'-m');
title('Maniobra terminal: posición','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p22,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])
max([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])])
% Plots señales de errores cartesianos velocidad x,y,z
subplot(3,2,3);
p23 = plot(bin_log(1,:),errorc_log(2,:),'-g',bin_log(1,:),errorc_log(5,:),'-
c',bin_log(1,:),errorc_log(8,:),'-m');
title('Trayectoria completa: velocidad','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s]','FontSize',16)
legend({'vel x','vel y','vel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p23,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(2,:) errorc_log(5,:) errorc_log(8,:)]) max([errorc_log(2,:)
errorc_log(5,:) errorc_log(8,:)])])

subplot(3,2,4);
p24 = plot(bin_log(1,lpx1:nt),errorc_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(5,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(8,lpx1:nt),'-m');
title('Maniobra terminal: velocidad','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s]','FontSize',16);
set(p24,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])
max([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])])
% Plots señales de errores cartesianos aceleración x,y,z
subplot(3,2,5);
p25 = plot(bin_log(1,:),errorc_log(3,:),'-g',bin_log(1,:),errorc_log(6,:),'-
c',bin_log(1,:),errorc_log(9,:),'-m');
title('Trayectoria completa: aceleración','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s2]','FontSize',16)
legend({'acel x','acel y','acel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p25,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(3,:) errorc_log(6,:) errorc_log(9,:)]) max([errorc_log(3,:)
errorc_log(6,:) errorc_log(9,:)])])

subplot(3,2,6);
p26 = plot(bin_log(1,lpx1:nt),errorc_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(6,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(9,lpx1:nt),'-m');
title('Maniobra terminal: aceleración','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s2]','FontSize',16);
set(p26,'LineWidth',1.5);
grid on
```

```
axis([lpx1 nt min([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])
max([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])])

%% Plots señales de error polares: Distancia Ad
figure(13);
sgtitle('Errores polares','FontSize',18)
maxAd =  37.5*ones(1,nt);
minAd = -37.5*ones(1,nt);

subplot(3,2,1);
laz1 = 4500;
p27 = plot(bin_log(1,:),errorp_log(1,:),'-g',bin_log(1,:),maxAd(1,:),'--
r',bin_log(1,:),minAd(1,:),'--r');
title('Trayectoria completa: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
legend({'Ad','Límite +/- 37.5 mts'},'NumColumns',2,'Location','North','FontSize',12);
set(p27(1),'LineWidth',1.5);
set(p27(2),'LineWidth',1.0);
set(p27(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(1,:) minAd(1,:)]) max([errorp_log(1,:) maxAd(1,:)])])

subplot(3,2,2);
p28 = plot(bin_log(1,laz1:nt),errorp_log(1,laz1:nt),'-
g',bin_log(1,laz1:nt),maxAd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minAd(1,laz1:nt),'--r');
title('Maniobra terminal: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p28(1),'LineWidth',1.5);
set(p28(2),'LineWidth',1.0);
set(p28(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(1,laz1:nt) minAd(1,laz1:nt)]) max([errorp_log(1,laz1:nt)
maxAd(1,laz1:nt)])])
% Error polar en acimut
maxBdn = 2*ones(1,nt);
minBdn = -2*ones(1,nt);

subplot(3,2,3);
laz1 = 4500;
p29 = plot(bin_log(1,:),errorp_log(2,:),'-g',bin_log(1,:),maxBdn(1,:),'--
r',bin_log(1,:),minBdn(1,:),'--r');
title('Trayectoria completa: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Bdn','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p29(1),'LineWidth',1.5);
set(p29(2),'LineWidth',1.0);
set(p29(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(2,:) minBdn(1,:)]) max([errorp_log(2,:) maxBdn(1,:)])])

subplot(3,2,4);
p30 = plot(bin_log(1,laz1:nt),errorp_log(2,laz1:nt),'-
g',bin_log(1,laz1:nt),maxBdn(1,laz1:nt),'--r',bin_log(1,laz1:nt),minBdn(1,laz1:nt),'--
r');
title('Maniobra terminal: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p30(1),'LineWidth',1.5);
set(p30(2),'LineWidth',1.0);
set(p30(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(2,laz1:nt) minBdn(1,laz1:nt)]) max([errorp_log(2,laz1:nt)
maxBdn(1,laz1:nt)])])

maxEd = 2*ones(1,nt);
minEd = -2*ones(1,nt);

subplot(3,2,5);
laz1 = 4500;
p31 = plot(bin_log(1,:),errorp_log(3,:),'-g',bin_log(1,:),maxEd(1,:),'--
r',bin_log(1,:),minEd(1,:),'--r');
title('Trayectoria completa: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
```

```
legend({'Ed','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p31(1),'LineWidth',1.5);
set(p31(2),'LineWidth',1.0);
set(p31(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(3,:) minEd(1,:)]) max([errorp_log(3,:) maxEd(1,:)])])

subplot(3,2,6);
p32 = plot(bin_log(1,laz1:nt),errorp_log(3,laz1:nt),'-
g',bin_log(1,laz1:nt),maxEd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minEd(1,laz1:nt),'--r');
title('Maniobra terminal: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p32(1),'LineWidth',1.5);
set(p32(2),'LineWidth',1.0);
set(p32(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(3,laz1:nt) minEd(1,laz1:nt)]) max([errorp_log(3,laz1:nt)
maxEd(1,laz1:nt)])])
%% Perturbaciones

figure(14);
sgtitle('Perturbaciones','FontSize',18)
lpx1 = 4500;
subplot(3,2,1);
p33 = plot(bin_log(1,:),pk_log(1,:),'-g',bin_log(1,:),phk_log(1,:),'-k');
title('Trayectoria completa: perturbación x','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('[g/s]','FontSize',16)
legend({'real','HOSMO'},'Location','north','NumColumns',2,'FontSize',14)
set(p33,'LineWidth',1.5);
grid on
axis([1 nt min([pk_log(1,:) phk_log(1,:)]) max([pk_log(1,:) phk_log(1,:)])])

subplot(3,2,2);
p34 = plot(bin_log(1,lpx1:nt),pk_log(1,lpx1:nt),'-
g',bin_log(1,lpx1:nt),phk_log(1,lpx1:nt),'-k');
title('Maniobra terminal: perturbación x','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p34,'LineWidth',1.5);
grid on
axis([lpx1 nt min([pk_log(1,lpx1:nt) phk_log(1,lpx1:nt)]) max([pk_log(1,lpx1:nt)
phk_log(1,lpx1:nt)])])

subplot(3,2,3);
p35 = plot(bin_log(1,:),pk_log(2,:),'-g',bin_log(1,:),phk_log(2,:),'-k');
title('Trayectoria completa: perturbación y','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('[g/s]','FontSize',16)
legend({'real','HOSMO'},'Location','north','NumColumns',2,'FontSize',14)
set(p35,'LineWidth',1.5);
grid on
axis([1 nt min([pk_log(2,:) phk_log(2,:)]) max([pk_log(2,:) phk_log(2,:)])])

subplot(3,2,4);
p36 = plot(bin_log(1,lpx1:nt),pk_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),phk_log(2,lpx1:nt),'-k');
title('Maniobra terminal: perturbación y','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p36,'LineWidth',1.5);
grid on
axis([lpx1 nt min([pk_log(2,lpx1:nt) phk_log(2,lpx1:nt)]) max([pk_log(2,lpx1:nt)
phk_log(2,lpx1:nt)])])

subplot(3,2,5);
p37 = plot(bin_log(1,:),pk_log(3,:),'-g',bin_log(1,:),phk_log(3,:),'-k');
title('Trayectoria completa: perturbación z','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('[g/s]','FontSize',16)
legend({'real','HOSMO'},'Location','north','NumColumns',2,'FontSize',14)
set(p37,'LineWidth',1.5);
grid on
axis([1 nt min([pk_log(3,:) phk_log(3,:)]) max([pk_log(3,:) phk_log(3,:)])])

subplot(3,2,6);
```

```
p38 = plot(bin_log(1,lpx1:nt),pk_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),phk_log(3,lpx1:nt),'-k');
title('Maniobra terminal: perturbación z','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p38,'LineWidth',1.5);
grid on
axis([lpx1 nt min([pk_log(3,lpx1:nt) phk_log(3,lpx1:nt)]) max([pk_log(3,lpx1:nt)
phk_log(3,lpx1:nt)])])

%% Gráfica de plano de estados (colector deslizante en el origen)
figure(15);

subplot(3,2,1);
laz1 = 4000;
p25 = plot(sigk_log(1,1:laz1),sigk_log(4,1:laz1),'-b');
title('Plano de estados (muestra = [1,4000])','FontSize',16);
xlabel('ex','FontSize',16);
ylabel('dex/dt','FontSize',16);
set(p25(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(1,1:laz1)) max(sigk_log(1,1:laz1)) -max(sigk_log(4,1:laz1))
max(sigk_log(4,1:laz1))])

subplot(3,2,2);
p26 = plot(sigk_log(1,laz1:nt),sigk_log(4,laz1:nt),'-b');
title('Plano de estados (muestra = [4000,5226])','FontSize',16);
xlabel('ex','FontSize',16);
ylabel('dex/dt','FontSize',16);
set(p26(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(1,laz1:nt)) max(sigk_log(1,laz1:nt)) -max(sigk_log(4,laz1:nt))
max(sigk_log(4,laz1:nt))])

subplot(3,2,3);
laz1 = 50;
p27 = plot(sigk_log(2,1:laz1),sigk_log(5,1:laz1),'-g');
title('Plano de estados (muestra = [1,50])','FontSize',16);
xlabel('ey','FontSize',16);
ylabel('dey/dt','FontSize',16);
set(p27(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(2,1:laz1)) max(sigk_log(2,1:laz1)) -max(sigk_log(5,1:laz1))
max(sigk_log(5,1:laz1))])

subplot(3,2,4);
p28 = plot(sigk_log(2,laz1:nt),sigk_log(5,laz1:nt),'-g');
title('Plano de estados (muestra = [50,5226])','FontSize',16);
xlabel('ey','FontSize',16);
ylabel('dey/dt','FontSize',16);
set(p28(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(2,laz1:nt)) max(sigk_log(2,laz1:nt)) -max(sigk_log(5,laz1:nt))
max(sigk_log(5,laz1:nt))])

subplot(3,2,5);
laz1 = 2000;
p29 = plot(sigk_log(3,1:laz1),sigk_log(6,1:laz1),'-r');
title('Plano de estados (muestra = [1,2000])','FontSize',16);
xlabel('ez','FontSize',16);
ylabel('dez/dt','FontSize',16);
set(p29(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(3,1:laz1)) max(sigk_log(3,1:laz1)) -max(sigk_log(6,1:laz1))
max(sigk_log(6,1:laz1))])

subplot(3,2,6);
p30 = plot(sigk_log(3,laz1:nt),sigk_log(6,laz1:nt),'-r');
title('Plano de estados (muestra = [2000,5226])','FontSize',16);
xlabel('ez','FontSize',16);
ylabel('dez/dt','FontSize',16);
set(p30(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(3,laz1:nt)) max(sigk_log(3,laz1:nt)) -max(sigk_log(6,laz1:nt))
max(sigk_log(6,laz1:nt))])
```

Apéndice 2.9: Diferenciador robusto exacto estándar (RED)

```
%=========================================================================
%
% Título            : RED.m
% Propósito         : Función del diferenciador robusto exacto estándar
%
% Descripción       : Este programa permite la estimación de parámetros de
%                     posición, velocidad, aceleración, sobre aceleración
%                     y derivada de la sobre aceleración de un blanco aéreo
%                     de alta maniobrabilidad a partir de las mediciones
%                     ruidosas de posición en coordenadas cartesianas
%                     obtenidas por un radar de seguimiento automático. Al
%                     colocar el parámetro theta = 1 el diferenciador
%                     robusto exacto utiliza exponentes de orden bajo,
%                     observándose que el diferenciador no observa
%                     adecuadamente las variables de estado de la planta o
%                     proceso. Por otro lado, al colocar el parámetro theta
%                     = 0, se utilizan exponentes de alto orden que
%                     permiten que el diferenciador converja en un tiempo
%                     finito al colector deslizante.
%                     El diferenciador robusto exacto utiliza adaptación de
%                     su constante de Lipschitz mediante la aplicación de
%                     una función de lyapunov apropiada.
%
% Fecha de creación : 11/10/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
function [xhk,uhk,duhk,sigk] = RED(yk,T)

persistent z0x z1x z2x z3x z4x z0y z1y z2y z3y z4y z0z z1z z2z z3z z4z

if isempty (z0x)
   z0x = yk(1,1);
   z1x = 0;
   z2x = 0;
   z3x = 0;
   z4x = 0;
   z0y = yk(3,1);
   z1y = 0;
   z2y = 0;
   z3y = 0;
   z4y = 0;
   z0z = yk(5,1);
   z1z = 0;
   z2z = 0;
   z3z = 0;
   z4z = 0;
end
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar (RED): coordenada x
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0x       posición
% z1x       velocidad
% z2x       aceleración
% z3x       sobre aceleración
% z4x       snap (derivada de la sobre aceleración)
Lx  = 454;  % Constante de Lipschitz (debe ser mayor que límite superior de
%           perturbación en la coordenada x
% yk(1,1)   posición medida
% T         Tiempo de muestreo
% n         Orden de diferenciación n = r - 1
%
% Párametros lambda (Homogeneidad de coeficientes)
%
n          = 4;
k0x        = 5.5*1;
k1x        = 10.03*1;
k2x        = 9.30*1;
k3x        = 4.57*1;
```

```
k4x        = 1.1*1;
%
% Series de Taylor para discretización del diferenciador
%
sum0x    = ((T^1)/1)*z1x + ((T^2)/(2*1))*z2x + ((T^3)/(3*2*1))*z3x +
((T^4)/(4*3*2*1))*z4x;
sum1x    = ((T^1)/1)*z2x + ((T^2)/(2*1))*z3x + ((T^3)/(3*2*1))*z4x;
sum2x    = ((T^1)/1)*z3x + ((T^2)/(2*1))*z4x;
sum3x    = ((T^1)/1)*z4x;
%
% Variable de error del diferenciador
%
sigx     = yk(1,1) - z0x;
%
% Diferenciador robusto exacto estándar: coordenada x
%
z0x_1    = z0x + T*k0x*(Lx^(1/(n+1)))*((abs(sigx))^((n)/(n+1)))*sign(sigx)   + sum0x;
z1x_1    = z1x + T*k1x*(Lx^(2/(n+1)))*((abs(sigx))^((n-1)/(n+1)))*sign(sigx) + sum1x;
z2x_1    = z2x + T*k2x*(Lx^(3/(n+1)))*((abs(sigx))^((n-2)/(n+1)))*sign(sigx) + sum2x;
z3x_1    = z3x + T*k3x*(Lx^(4/(n+1)))*((abs(sigx))^((n-3)/(n+1)))*sign(sigx) + sum3x;
z4x_1    = z4x + T*k4x*(Lx^(5/(n+1)))*sign(sigx);
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar (RED): coordenada y
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0y       posición
% z1y       velocidad
% z2y       aceleración
% z3y       sobre aceleración
% z4y       snap (derivada de la sobre aceleración)
Ly  = 1162;  % Constante de Lipschitz (debe ser mayor que límite superior de
%           perturbación en la coordenada x
% yk(1,1)   posición medida
% T         Tiempo de muestreo
% n         Orden de diferenciación n = r - 1
%
% Párametros lambda (Homogeneidad de coeficientes)
%
n          = 4;
k0y        = 5.5*1;
k1y        = 10.03*1;
k2y        = 9.30*1;
k3y        = 4.57*1;
k4y        = 1.1*1;
%
% Series de Taylor para discretización del diferenciador
%
sum0y    = ((T^1)/1)*z1y + ((T^2)/(2*1))*z2y + ((T^3)/(3*2*1))*z3y +
((T^4)/(4*3*2*1))*z4y;
sum1y    = ((T^1)/1)*z2y + ((T^2)/(2*1))*z3y + ((T^3)/(3*2*1))*z4y;
sum2y    = ((T^1)/1)*z3y + ((T^2)/(2*1))*z4y;
sum3y    = ((T^1)/1)*z4y;
%
% Variable de error del diferenciador
%
sigy     = yk(3,1) - z0y;
%
% Diferenciador robusto exacto estándar: coordenada y
%
z0y_1    = z0y + T*k0y*(Ly^(1/(n+1)))*((abs(sigy))^((n)/(n+1)))*sign(sigy)   + sum0y;
z1y_1    = z1y + T*k1y*(Ly^(2/(n+1)))*((abs(sigy))^((n-1)/(n+1)))*sign(sigy) + sum1y;
z2y_1    = z2y + T*k2y*(Ly^(3/(n+1)))*((abs(sigy))^((n-2)/(n+1)))*sign(sigy) + sum2y;
z3y_1    = z3y + T*k3y*(Ly^(4/(n+1)))*((abs(sigy))^((n-3)/(n+1)))*sign(sigy) + sum3y;
z4y_1    = z4y + T*k4y*(Ly^(5/(n+1)))*sign(sigy);
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar (RED): coordenada z
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0z       posición coordenada z
% z1z       velocidad coordenada z
% z2z       aceleración coordenada z
% z3z       sobre aceleración coordenada z
% z4z       snap (derivada de la sobre aceleración) coordenada z
```

```
Lz  = 306;  % Constante de Lipschitz (debe ser mayor que límite superior de
%           perturbación en la coordenada z
% yk(3,1)   posición medida coordenada z
% T         Tiempo de muestreo
% n         Orden de diferenciación n = r - 1
%
% Párametros lambda (Homogeneidad de coeficientes)
%
n          = 4;
k0z        = 5.5*1;
k1z        = 10.03*1;
k2z        = 9.30*1;
k3z        = 4.57*1;
k4z        = 1.1*1;
%
% Series de Taylor para discretización del diferenciador
%
sum0z    = ((T^1)/1)*z1z + ((T^2)/(2*1))*z2z + ((T^3)/(3*2*1))*z3z +
((T^4)/(4*3*2*1))*z4z;
sum1z    = ((T^1)/1)*z2z + ((T^2)/(2*1))*z3z + ((T^3)/(3*2*1))*z4z;
sum2z    = ((T^1)/1)*z3z + ((T^2)/(2*1))*z4z;
sum3z    = ((T^1)/1)*z4z;
%
% Variable de error del diferenciador
%
sigz     = yk(5,1) - z0z;
%
% Diferenciador robusto exacto estándar: coordenada z
%
z0z_1    = z0z + T*k0z*(Lz^(1/(n+1)))*((abs(sigz))^((n)/(n+1)))*sign(sigz)   + sum0z;
z1z_1    = z1z + T*k1z*(Lz^(2/(n+1)))*((abs(sigz))^((n-1)/(n+1)))*sign(sigz) + sum1z;
z2z_1    = z2z + T*k2z*(Lz^(3/(n+1)))*((abs(sigz))^((n-2)/(n+1)))*sign(sigz) + sum2z;
z3z_1    = z3z + T*k3z*(Lz^(4/(n+1)))*((abs(sigz))^((n-3)/(n+1)))*sign(sigz) + sum3z;
z4z_1    = z4z + T*k4z*(Lz^(5/(n+1)))*sign(sigz);
%--------------------------------------------------------------------------
% Definición de superficies deslizantes (error de velocidad)
%--------------------------------------------------------------------------
% Estas superficies deslizantes no intervienen en el algoritmo. Solo serán
% utilizadas para demostrar graficamente la torsión generada por el
% algoritmo.
sigdx       = yk(2,1) - z1x_1;
sigdy       = yk(4,1) - z1y_1;
sigdz       = yk(6,1) - z1z_1;

%--------------------------------------------------------------------------
% Guardar variables para siguiente iteración
%--------------------------------------------------------------------------
z0x = z0x_1;
z1x = z1x_1;
z2x = z2x_1;
z3x = z3x_1;
z4x = z4x_1;
z0y = z0y_1;
z1y = z1y_1;
z2y = z2y_1;
z3y = z3y_1;
z4y = z4y_1;
z0z = z0z_1;
z1z = z1z_1;
z2z = z2z_1;
z3z = z3z_1;
z4z = z4z_1;
xhk = [z0x;z1x;z2x;z0y;z1y;z2y;z0z;z1z;z2z];
uhk = [z3x;z3y;z3z];
duhk = [z4x;z4y;z4z];
sigk        = [sigx sigy sigz sigdx sigdy sigdz]';
end
```

Apéndice 2.10: Simulación de la trayectoria del misil (RED)

```
%==========================================================================
%
% Título            : sim_RED.m
% Propósito         : Simulación de diferenciador robusto exacto
%
% Descripción       : Este programa permite la estimación de parámetros de
```

```
%                    posición, velocidad, aceleración, sobre aceleración
%                    y derivada de la sobre aceleración de un blanco aéreo
%                    de alta maniobrabilidad a partir de las mediciones
%                    ruidosas de posición en coordenadas cartesianas
%                    obtenidas por un radar de seguimiento automático.
%
% Fecha de creación : 11/10/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%==========================================================================
clear all;
clc;
close all;
%--------------------------------------------------------------------------
% Inicialización de variables
%--------------------------------------------------------------------------
RMSEcold = 0;
RMSEpold = 0;
%--------------------------------------------------------------------------
% Parámetros iniciales de radar (sensor)
%--------------------------------------------------------------------------
T     = 1/50;                       % Período de muestreo (seg)
%--------------------------------------------------------------------------
% Definición de constantes físicas
%--------------------------------------------------------------------------
g     = 9.81;                       % gravedad (m/s2)
M     = 343;                        % velocidad del sonido (m/s)
load_factor = 100*g;                % máxima sobre aceleración (g/s)
%--------------------------------------------------------------------------
% Definición de vectores de tiempo
%--------------------------------------------------------------------------
ti    = 0;                          % Tiempo inicial de simulación
tff   = 104.5;                      % Tiempo final de simulación
tt    = ti:T:tff;                   % Vector de tiempo
tt    = tt';                        % Transpuesta del vector de tiempo
nt    = length(tt);                 % Número de muestras
%--------------------------------------------------------------------------
% Definición de posición y velocidad inicial del blanco en
% coordenadas esféricas
%--------------------------------------------------------------------------
Ad_init  = 35000;                       % Distancia al blanco (m)
Ed_init  = 0.01;                        % Elevación al blanco (°)
Bdn_init = 30;                          % Marcación al blanco (°)
Vm_init  = 0.9*M;                       % Velocidad del blanco (1 Mach = 343 m/s)
Rv_init  = 210;                         % Rumbo inicial del blanco (°)
%--------------------------------------------------------------------------
% Conversión de coordenadas esféricas a cartesianas (North-Sky-East) de la
% posición y velocidad inicial del blanco
%--------------------------------------------------------------------------
[Adx_init, Ady_init, Adz_init] = p2c(Ad_init,Ed_init,Bdn_init);
[Vmx_init, Vmy_init, Vmz_init] = p2c(Vm_init,0,Rv_init);
%--------------------------------------------------------------------------
% Inicialización de vector de estados
%--------------------------------------------------------------------------
xk = [Adx_init Vmx_init 0*g Ady_init Vmy_init 0*g Adz_init Vmz_init 0*g]';
%--------------------------------------------------------------------------
% Definición de matrices del modelo de espacio de estados del sistema
%--------------------------------------------------------------------------
% Matriz de medición (Pulse radar)
Ck     = [1 0 0 0 0 0 0 0 0;        % pos x
          0 0 0 0 0 0 0 0 0;        % vel x
          0 0 0 1 0 0 0 0 0;        % pos y
          0 0 0 0 0 0 0 0 0;        % vel y
          0 0 0 0 0 0 1 0 0;        % pos z
          0 0 0 0 0 0 0 0 0];       % vel z

% Cálculo de matriz de transición Fk
Fk = [1 T (T^2)/2 0 0       0 0 0       0;
      0 1       T 0 0       0 0 0       0;
      0 0       1 0 0       0 0 0       0;
      0 0       0 1 T (T^2)/2 0 0       0;
      0 0       0 0 1       T 0 0       0;
      0 0       0 0 0       1 0 0       0;
```

```
      0 0       0 0 0       0 1 T (T^2)/2;
      0 0       0 0 0       0 0 1       T;
      0 0       0 0 0       0 0 0       1];

% Cálculo de la matriz de entrada Gk
Gk = [(T^3)/6       0        0;
      (T^2)/2       0        0;
       T            0        0;
       0      (T^3)/6        0;
       0      (T^2)/2        0;
       0            T        0;
       0            0  (T^3)/6;
       0            0  (T^2)/2;
       0            0        T];

% Matriz de distribución de perturbaciones (aceleraciones)
Dk = [75e4*(T^3)/6       0                0;
      5e3*(T^2)/2        0                0;
       1e2*T             0                0;
       0      75e4*(T^3)/6                0;
       0       5e3*(T^2)/2                0;
       0             1e2*T                0;
       0                 0     75e4*(T^3)/6;
       0                 0      5e3*(T^2)/2;
       0                 0           1e2*T];
Dk = [75e4*(T^3)/6       0                0;
      5e3*(T^2)/2        0                0;
       1e2*T             0                0;
       0      75e4*(T^3)/6                0;
       0       5e3*(T^2)/2                0;
       0             1e2*T                0;
       0                 0     75e4*(T^3)/6;
       0                 0      5e3*(T^2)/2;
       0                 0           1e2*T];
%-------------------------------------------------------------------------
% Generación del vector de sobre aceleraciones de entrada uk
%-------------------------------------------------------------------------
uk = gen_jerk_sea_skimming3(nt,load_factor,g,T);
%-------------------------------------------------------------------------
% Generación del vector de perturbaciones ?k
%-------------------------------------------------------------------------
pk = perturb_1(g,ti,T,tff);
%-------------------------------------------------------------------------
% Cargar ruido glint
%-------------------------------------------------------------------------
load glint_puntos.mat
%-------------------------------------------------------------------------
% Simulación del diferenciador
%-------------------------------------------------------------------------
disp('Diferenciador Robusto Exacto de Levant');
prompt = 'Para simular con ruido presione "1". Para simular sin ruido presione "0": ';
gl     = input(prompt);
for k = 1:nt

    % Modelo dinámico o de transición de estados
    xk       = Fk*xk + Gk*uk(:,k) + Dk*pk(:,k);
    % Simulación de adquisición de datos por el radar
    % Datos de posición adquiridos en coordenadas esféricas
    [Ad,Bdn,Ed] = c2p(xk);
    % Modelo de medición polar a cartesiano
    yk = yk_noise(Ad,Bdn,Ed,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,gl,k);
    % Diferenciador Robusto Exacto
    tic
    [xhk,uhk,duhk,sigk] = RED(yk,T);
    TCPU = toc;

    % Conversión de coordenada estimadas cartesianas a coordenadas polares
    [Adhat,Bdnhat,Edhat] = c2p(xhk);

    % Cálculo de errores en coordenadas cartesianas
    errorc       = xhk - xk;

    % Cálculo de errores en coordenadas esféricas
    errorAd      = Adhat - Ad;                                    %mts
    errorBdn     = Bdnhat*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEd      = Edhat*(pi*1000/180) - Ed*(pi*1000/180);
    errorp       = [errorAd; errorBdn; errorEd];
```

```
    % Cálculo de RMSE
    if ( k >= 4650)
        RMSEc          = RMSEcold + ((errorc).^2);
        RMSEp          = RMSEpold + ((errorp).^2);
    else
        RMSEc          = 0;
        RMSEp          = 0;
    end
    bin_log(:,k)       = k;
    xk_log(:,k)        = xk;
    xhk_log(:,k)       = xhk;
    uhk_log(:,k)       = uhk;
    duhk_log(:,k)      = duhk;
    sigk_log(:,k)      = sigk;
    errorc_log(:,k)    = errorc;
    errorp_log(:,k)    = errorp;
    polar_log(:,k)     = [Ad;Bdn;Ed];
    polarhat_log(:,k)  = [Adhat;Bdnhat;Edhat];
    yk_log(:,k)        = yk;
    TCPU_log(:,k)      = TCPU;
    % Guardar variables
    RMSEcold           = RMSEc;
    RMSEpold           = RMSEp;
end
RMSEctot = sqrt(RMSEc)/(nt-4650);
RMSEptot = sqrt(RMSEp)/(nt-4650);
%-------------------------------------------------------------------------
% Conversión de vector de estados reales a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
% 0.10197 - de mts/seg2 a g's
conv_xk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197]';
xk_log = xk_log.*conv_xk;
%-------------------------------------------------------------------------
% Conversión de vector de estados medidos a Mn, Mach
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
conv_yk   = [ 0.00054 0.00292 0.00054 0.00292 0.00054 0.00292 ]';
yk_log    = yk_log.*conv_yk;
%-------------------------------------------------------------------------
% Conversión de vector de estados estimados a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - mn
% 0.00292 - Mach
% 0.10197 - g's
conv_xhk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292
0.10197]';
xhk_log = xhk_log.*conv_xhk;
%-------------------------------------------------------------------------
% Conversión de vector de entrada real y estimado a g/s
%-------------------------------------------------------------------------
uk_log    = uk./g;
uhk_log   = uhk_log./g;
duhk_log   = duhk_log./g;
%-------------------------------------------------------------------------
% Conversión de perturbaciones a g/s
%-------------------------------------------------------------------------
pk_log    = pk./g;
%-------------------------------------------------------------------------
% Plots
%-------------------------------------------------------------------------
%%
figure(1);p1 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                              0,                  0,                  0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                   0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p1(1),'LineWidth',2.0);
set(p1(2),'LineWidth',1.8);
set(p1(3),'LineWidth',1.5);
set(p1(4),'LineWidth',2.0);
```

```
set(p1(5),'LineWidth',2.0);
set(p1(6),'LineWidth',2.0);
set(p1(7),'LineWidth',2.0);

%axis([-5 20 -5 10 0 0.04]);          % Toda la trayectoria
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Trayectoria sea skimming del misil','FontSize',16)
legend({'mediciones','real','RED','Posición del buque observador','Posición inicial del
misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
%% 3D trayectoria completa

figure(2);p2 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                0,                0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p2(1),'LineWidth',2.0);
set(p2(2),'LineWidth',1.8);
set(p2(3),'LineWidth',1.5);
set(p2(4),'LineWidth',2.0);
set(p2(5),'LineWidth',2.0);
set(p2(6),'LineWidth',2.0);
set(p2(7),'LineWidth',2.0);
axis([4 8 -2 5 0 0.016]);
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Maniobra terminal del misil','FontSize',16)
legend({'mediciones','real','RED','Posición del buque observador','Posición inicial del
misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14);

%% Coordenada x
% Posición x: Trayectoria completa
figure(3);
subplot(1,2,1);
p3 = plot(bin_log(1,:),yk_log(1,:),'-r',bin_log(1,:),xk_log(1,:),'-
g',bin_log(1,:),xhk_log(1,:),'-k');
title('Trayectoria completa del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p3(1),'LineWidth', 2.0);
set(p3(2),'LineWidth', 1.8);
set(p3(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(1,:) yk_log(1,:)]) max([xk_log(1,:) yk_log(1,:)])])

% Posición x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p4 = plot(bin_log(1,lpx1:nt),yk_log(1,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(1,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(1,lpx1:nt),'-
k');
title('Maniobra terminal del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p4(1),'LineWidth', 2.0);
set(p4(2),'LineWidth', 1.8);
set(p4(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)]) max([xk_log(1,lpx1:nt)
yk_log(1,lpx1:nt)])])
%%
% Velocidad x: Trayectoria completa
figure(4);
```

```
subplot(1,2,1);
p5 = plot(bin_log(1,:),xk_log(2,:),'-g',bin_log(1,:),xhk_log(2,:),'-k');
title('Trayectoria completa del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p5(1),'LineWidth', 1.8);
set(p5(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(2,:)) max(xk_log(2,:))])
%axis([1 nt -10 10])
% Velocidad x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p6 = plot(bin_log(1,lpx1:nt),xk_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(2,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p6(1),'LineWidth', 1.8);
set(p6(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt -1 2]) % con ruido
%axis([lpx1 nt min([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])
max([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])])
%%
% Aceleración x: Trayectoria completa
figure(5);
subplot(1,2,1);
p7 = plot(bin_log(1,:),xk_log(3,:),'-g',bin_log(1,:),xhk_log(3,:),'-k');
title('Trayectoria completa del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p7(1),'LineWidth', 1.8);
set(p7(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(3,:)) max(xk_log(3,:))])
%axis([1 nt min([xk_log(3,:) xhk_log(3,:)]) max([xk_log(3,:) xhk_log(3,:)])])
% Aceleración x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
p8 = plot(bin_log(1,lpx1:nt),xk_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(3,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p8(1),'LineWidth', 1.8);
set(p8(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(3,lpx1:nt) xhk_log(3,lpx1:nt)]) max([xk_log(3,lpx1:nt)
xhk_log(3,lpx1:nt)])])
%axis([lpx1 nt -20 20])
%% Coordenada y
% Posición y: Trayectoria completa
figure(6);
subplot(1,2,1);
p9 = plot(bin_log(1,:),yk_log(3,:),'-r',bin_log(1,:),xk_log(4,:),'-
g',bin_log(1,:),xhk_log(4,:),'-k');
title('Trayectoria completa del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p9(1),'LineWidth', 2.0);
set(p9(2),'LineWidth', 1.8);
set(p9(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(4,:) yk_log(3,:)]) max([xk_log(4,:) yk_log(3,:)])])

% Posición y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p10 = plot(bin_log(1,lpx1:nt),yk_log(3,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(4,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(4,lpx1:nt),'-
K');
```

```
title('Maniobra terminal del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p10(1),'LineWidth', 2.0);
set(p10(2),'LineWidth', 1.8);
set(p10(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])
max([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])])
%%
% Velocidad y: Trayectoria completa
figure(7);
subplot(1,2,1);
p11 = plot(bin_log(1,:),xk_log(5,:),'-g',bin_log(1,:),xhk_log(5,:),'-k');
title('Trayectoria completa del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','RED'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p11(1),'LineWidth', 1.8);
set(p11(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(5,:)) max(xk_log(5,:))])
%axis([1 nt -10 10])
% Velocidad y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p12 = plot(bin_log(1,lpx1:nt),xk_log(5,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(5,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p12(1),'LineWidth', 1.8);
set(p12(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min(xk_log(5,lpx1:nt)) max(xk_log(5,lpx1:nt))])
%%
figure(8);
% Aceleración y: Trayectoria
subplot(1,2,1);
p14 = plot(bin_log(1,:),xk_log(6,:),'-g',bin_log(1,:),xhk_log(6,:),'-k');
title('Maniobra terminal del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p14(1),'LineWidth', 1.8);
set(p14(2),'LineWidth', 1.5);
grid on
%axis([1 nt min([xk_log(6,:) xhk_log(6,:)]) max([xk_log(6,:) xhk_log(6,:)])])
axis([1 nt min(xk_log(6,:)) max(xk_log(6,:))])
% Aceleración y: Maniobra terminal
subplot(1,2,2);
p13 = plot(bin_log(1,lpx1:nt),xk_log(6,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(6,lpx1:nt),'-k');
title('Trayectoria completa del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p13(1),'LineWidth', 1.8);
set(p13(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(6,lpx1:nt) xhk_log(6,lpx1:nt)]) max([xk_log(6,lpx1:nt)
xhk_log(6,lpx1:nt)])])

%% Coordenada z
% Posición z: Trayectoria completa
figure(9);
subplot(1,2,1);
p15 = plot(bin_log(1,:),yk_log(5,:)./0.00054,'-r',bin_log(1,:),xk_log(7,:)./0.00054,'-
g',bin_log(1,:),xhk_log(7,:)./0.00054,'-k');
title('Trayectoria completa del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p15(1),'LineWidth', 2.0);
set(p15(2),'LineWidth', 1.8);
```

```
set(p15(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(7,:)./0.00054 yk_log(5,:)./0.00054]) max([xk_log(7,:)./0.00054
yk_log(5,:)./0.00054])])

% Posición z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p16 = plot(bin_log(1,lpx1:nt),yk_log(5,lpx1:nt)./0.00054,'-
r',bin_log(1,lpx1:nt),xk_log(7,lpx1:nt)./0.00054,'-
g',bin_log(1,lpx1:nt),xhk_log(7,lpx1:nt)./0.00054,'-k');
title('Maniobra terminal del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mts]','FontSize',22)
legend({'medición','real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p16(1),'LineWidth', 2.0);
set(p16(2),'LineWidth', 1.8);
set(p16(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054]) max([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054])])
%%
% Velocidad z: Trayectoria completa
figure(10);
subplot(1,2,1);
p17 = plot(bin_log(1,:),xk_log(8,:),'-g',bin_log(1,:),xhk_log(8,:),'-k');
title('Trayectoria completa del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','RED'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p17(1),'LineWidth', 1.8);
set(p17(2),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(8,:) xhk_log(8,:)]) max([xk_log(8,:) xhk_log(8,:)])])
%axis([1 nt -4 4])
% Velocidad z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p18 = plot(bin_log(1,lpx1:nt),xk_log(8,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(8,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p18(1),'LineWidth', 1.8);
set(p18(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(8,lpx1:nt) xhk_log(8,lpx1:nt)]) max([xk_log(8,lpx1:nt)
xhk_log(8,lpx1:nt)])])
%%
% Aceleración z: Trayectoria completa
figure(11);
subplot(1,2,1);
p19 = plot(bin_log(1,:),xk_log(9,:),'-g',bin_log(1,:),xhk_log(9,:),'-k');
title('Trayectoria completa del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p19(1),'LineWidth', 1.8);
set(p19(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(9,:)) max(xk_log(9,:))])
%axis([1 nt min([xk_log(9,:) xhk_log(9,:)]) max([xk_log(9,:) xhk_log(9,:)])])

% Aceleración z: Maniobra terminal
subplot(1,2,2);
p20 = plot(bin_log(1,lpx1:nt),xk_log(9,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(9,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p20(1),'LineWidth', 1.8);
set(p20(2),'LineWidth', 1.5);
grid on
```

```
axis([lpx1 nt min([xk_log(9,lpx1:nt) xhk_log(9,lpx1:nt)]) max([xk_log(9,lpx1:nt)
xhk_log(9,lpx1:nt)])])

%% Plots señales de errores cartesianos posición x,y,z
figure(12);
sgtitle('Errores cartesianos','FontSize',18)
lpx1 = 4500;
subplot(3,2,1);
p21 = plot(bin_log(1,:),errorc_log(1,:),'-g',bin_log(1,:),errorc_log(4,:),'-
c',bin_log(1,:),errorc_log(7,:),'-m');
title('Trayectoria completa: posición','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m]','FontSize',16)
legend({'pos x','pos y','pos z'},'Location','north','NumColumns',3,'FontSize',14)
set(p21,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(1,:) errorc_log(4,:) errorc_log(7,:)]) max([errorc_log(1,:)
errorc_log(4,:) errorc_log(7,:)])])

subplot(3,2,2);
p22 = plot(bin_log(1,lpx1:nt),errorc_log(1,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(4,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(7,lpx1:nt),'-m');
title('Maniobra terminal: posición','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p22,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])
max([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])])
% Plots señales de errores cartesianos velocidad x,y,z
subplot(3,2,3);
p23 = plot(bin_log(1,:),errorc_log(2,:),'-g',bin_log(1,:),errorc_log(5,:),'-
c',bin_log(1,:),errorc_log(8,:),'-m');
title('Trayectoria completa: velocidad','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s]','FontSize',16)
legend({'vel x','vel y','vel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p23,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(2,:) errorc_log(5,:) errorc_log(8,:)]) max([errorc_log(2,:)
errorc_log(5,:) errorc_log(8,:)])])

subplot(3,2,4);
p24 = plot(bin_log(1,lpx1:nt),errorc_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(5,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(8,lpx1:nt),'-m');
title('Maniobra terminal: velocidad','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s]','FontSize',16);
set(p24,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])
max([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])])
% Plots señales de errores cartesianos aceleración x,y,z
subplot(3,2,5);
p25 = plot(bin_log(1,:),errorc_log(3,:),'-g',bin_log(1,:),errorc_log(6,:),'-
c',bin_log(1,:),errorc_log(9,:),'-m');
title('Trayectoria completa: aceleración','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s2]','FontSize',16)
legend({'acel x','acel y','acel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p25,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(3,:) errorc_log(6,:) errorc_log(9,:)]) max([errorc_log(3,:)
errorc_log(6,:) errorc_log(9,:)])])

subplot(3,2,6);
p26 = plot(bin_log(1,lpx1:nt),errorc_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(6,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(9,lpx1:nt),'-m');
title('Maniobra terminal: aceleración','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s2]','FontSize',16);
set(p26,'LineWidth',1.5);
grid on
```

```
axis([lpx1 nt min([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])
max([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])])

%% Plots señales de error polares: Distancia Ad
figure(13);
sgtitle('Errores polares','FontSize',18)
maxAd =  37.5*ones(1,nt);
minAd = -37.5*ones(1,nt);

subplot(3,2,1);
laz1 = 4500;
p27 = plot(bin_log(1,:),errorp_log(1,:),'-g',bin_log(1,:),maxAd(1,:),'--
r',bin_log(1,:),minAd(1,:),'--r');
title('Trayectoria completa: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
legend({'Ad','Límite +/- 37.5 mts'},'NumColumns',2,'Location','North','FontSize',12);
set(p27(1),'LineWidth',1.5);
set(p27(2),'LineWidth',1.0);
set(p27(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(1,:) minAd(1,:)]) max([errorp_log(1,:) maxAd(1,:)])])

subplot(3,2,2);
p28 = plot(bin_log(1,laz1:nt),errorp_log(1,laz1:nt),'-
g',bin_log(1,laz1:nt),maxAd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minAd(1,laz1:nt),'--r');
title('Maniobra terminal: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p28(1),'LineWidth',1.5);
set(p28(2),'LineWidth',1.0);
set(p28(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(1,laz1:nt) minAd(1,laz1:nt)]) max([errorp_log(1,laz1:nt)
maxAd(1,laz1:nt)])])
% Error polar en acimut
maxBdn = 2*ones(1,nt);
minBdn = -2*ones(1,nt);

subplot(3,2,3);
laz1 = 4500;
p29 = plot(bin_log(1,:),errorp_log(2,:),'-g',bin_log(1,:),maxBdn(1,:),'--
r',bin_log(1,:),minBdn(1,:),'--r');
title('Trayectoria completa: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Bdn','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p29(1),'LineWidth',1.5);
set(p29(2),'LineWidth',1.0);
set(p29(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(2,:) minBdn(1,:)]) max([errorp_log(2,:) maxBdn(1,:)])])

subplot(3,2,4);
p30 = plot(bin_log(1,laz1:nt),errorp_log(2,laz1:nt),'-
g',bin_log(1,laz1:nt),maxBdn(1,laz1:nt),'--r',bin_log(1,laz1:nt),minBdn(1,laz1:nt),'--
r');
title('Maniobra terminal: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p30(1),'LineWidth',1.5);
set(p30(2),'LineWidth',1.0);
set(p30(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(2,laz1:nt) minBdn(1,laz1:nt)]) max([errorp_log(2,laz1:nt)
maxBdn(1,laz1:nt)])])

maxEd = 2*ones(1,nt);
minEd = -2*ones(1,nt);

subplot(3,2,5);
laz1 = 4500;
p31 = plot(bin_log(1,:),errorp_log(3,:),'-g',bin_log(1,:),maxEd(1,:),'--
r',bin_log(1,:),minEd(1,:),'--r');
title('Trayectoria completa: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
```

```
legend({'Ed','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p31(1),'LineWidth',1.5);
set(p31(2),'LineWidth',1.0);
set(p31(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(3,:) minEd(1,:)]) max([errorp_log(3,:) maxEd(1,:)])])

subplot(3,2,6);
p32 = plot(bin_log(1,laz1:nt),errorp_log(3,laz1:nt),'-
g',bin_log(1,laz1:nt),maxEd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minEd(1,laz1:nt),'--r');
title('Maniobra terminal: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p32(1),'LineWidth',1.5);
set(p32(2),'LineWidth',1.0);
set(p32(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(3,laz1:nt) minEd(1,laz1:nt)]) max([errorp_log(3,laz1:nt)
maxEd(1,laz1:nt)])])
%% Estimación de vector de entrada de control
figure(25);
p44 = plot(bin_log(1,:),uk_log(1,:),'-g',bin_log(1,:),uhk_log(1,:),'-
k',bin_log(1,:),pk_log(1,:),'-m');
title('Trayectoria del misil: sobre aceleración x','FontSize',16);
legend({'real uk','RED uhatk','real
pk'},'Location','North','NumColumns',3,'FontSize',14);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p44(1),'LineWidth',1.5);
set(p44(2),'LineWidth',1.5);
set(p44(3),'LineWidth',1.5);
grid on
axis([1 nt min([uk_log(1,:) uhk_log(1,:)]) max([uk_log(1,:) uhk_log(1,:)])])

figure(26);
p45 = plot(bin_log(1,:),uk_log(2,:),'-g',bin_log(1,:),uhk_log(2,:),'-
k',bin_log(1,:),pk_log(2,:),'-m');
title('Trayectoria del misil: sobre aceleración y','FontSize',16);
legend({'real uk','RED uhatk','real
pk'},'Location','North','NumColumns',3,'FontSize',14);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p45(1),'LineWidth',1.5);
set(p45(2),'LineWidth',1.5);
set(p45(3),'LineWidth',1.5);
grid on
axis([1 nt min([uk_log(2,:) uhk_log(2,:)]) max([uk_log(2,:) uhk_log(2,:)])])

figure(27);
p46 = plot(bin_log(1,:),uk_log(3,:),'-g',bin_log(1,:),uhk_log(3,:),'-
k',bin_log(1,:),pk_log(3,:),'-m');
title('Trayectoria del misil: sobre aceleración z','FontSize',16);
legend({'real uk','RED uhatk','real
pk'},'Location','North','NumColumns',3,'FontSize',14);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p46(1),'LineWidth',1.5);
set(p46(2),'LineWidth',1.5);
set(p46(3),'LineWidth',1.5);
grid on
axis([1 nt min([uk_log(3,:) uhk_log(3,:)]) max([uk_log(3,:) uhk_log(3,:)])])
%% Gráfica de plano de estados (colector deslizante en el origen)
figure(28);

subplot(3,2,1);
laz1 = 500;
p25 = plot(sigk_log(1,1:laz1),sigk_log(4,1:laz1),'-b');
title('Plano de estados (muestra = [1,500])','FontSize',16);
xlabel('ex','FontSize',16);
ylabel('dex/dt','FontSize',16);
set(p25(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(1,1:laz1)) max(sigk_log(1,1:laz1)) -max(sigk_log(4,1:laz1))
max(sigk_log(4,1:laz1))])

subplot(3,2,2);
p26 = plot(sigk_log(1,laz1:nt),sigk_log(4,laz1:nt),'-b');
```

```
title('Plano de estados (muestra = [500,5226])','FontSize',16);
xlabel('ex','FontSize',16);
ylabel('dex/dt','FontSize',16);
set(p26(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(1,laz1:nt)) max(sigk_log(1,laz1:nt)) -max(sigk_log(4,laz1:nt))
max(sigk_log(4,laz1:nt))])

subplot(3,2,3);
laz1 = 2000;
p27 = plot(sigk_log(2,1:laz1),sigk_log(5,1:laz1),'-g');
title('Plano de estados (muestra = [1,2000])','FontSize',16);
xlabel('ey','FontSize',16);
ylabel('dey/dt','FontSize',16);
set(p27(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(2,1:laz1)) max(sigk_log(2,1:laz1)) -max(sigk_log(5,1:laz1))
max(sigk_log(5,1:laz1))])

subplot(3,2,4);
p28 = plot(sigk_log(2,laz1:nt),sigk_log(5,laz1:nt),'-g');
title('Plano de estados (muestra = [2000,5226])','FontSize',16);
xlabel('ey','FontSize',16);
ylabel('dey/dt','FontSize',16);
set(p28(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(2,laz1:nt)) max(sigk_log(2,laz1:nt)) -max(sigk_log(5,laz1:nt))
max(sigk_log(5,laz1:nt))])

subplot(3,2,5);
laz1 = 100;
p29 = plot(sigk_log(3,1:laz1),sigk_log(6,1:laz1),'-r');
title('Plano de estados (muestra = [1,100])','FontSize',16);
xlabel('ez','FontSize',16);
ylabel('dez/dt','FontSize',16);
set(p29(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(3,1:laz1)) max(sigk_log(3,1:laz1)) -max(sigk_log(6,1:laz1))
max(sigk_log(6,1:laz1))])

subplot(3,2,6);
p30 = plot(sigk_log(3,laz1:nt),sigk_log(6,laz1:nt),'-r');
title('Plano de estados (muestra = [100,5226])','FontSize',16);
xlabel('ez','FontSize',16);
ylabel('dez/dt','FontSize',16);
set(p30(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(3,laz1:nt)) max(sigk_log(3,laz1:nt)) -max(sigk_log(6,laz1:nt))
max(sigk_log(6,laz1:nt))])
```

Apéndice 2.11: Diferenciador robusto exacto adaptativo (ARED)

```
%==========================================================================
%
% Título            : ARED.m
% Propósito         : Función del diferenciador robusto exacto estándar
%                     adaptativo
%
% Descripción       : Este programa permite la estimación de parámetros de
%                     posición, velocidad, aceleración, sobre aceleración
%                     y derivada de la sobre aceleración de un blanco aéreo
%                     de alta maniobrabilidad a partir de las mediciones
%                     ruidosas de posición en coordenadas cartesianas
%                     obtenidas por un radar de seguimiento automático.
%                     El diferenciador robusto exacto utiliza adaptación de
%                     su constante de Lipschitz mediante la aplicación de
%                     una función de Lyapunov que permite el establecer un
%                     modo deslizante sin necesidad de conocer el límite
%                     superior del colector deslizante.
%
% Fecha de creación : 15/02/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
```

```
%
%=========================================================================
function [xhk,uhk,duhk,Lk,MUk,thetak,sigk] = ARED2(yk,T,k)

persistent Lx0 Ly0 Lz0 mux0 muy0 muz0 z0x z1x z2x z3x z4x z0y z1y z2y z3y z4y z0z z1z
z2z z3z z4z

if isempty (z0x)
   z0x = 0.98*yk(1,1);
   z1x = 0;
   z2x = 0;
   z3x = 0;
   z4x = 0;
   z0y = 0.98*yk(3,1);
   z1y = 0;
   z2y = 0;
   z3y = 0;
   z4y = 0;
   z0z = 0.98*yk(5,1);
   z1z = 0;
   z2z = 0;
   z3z = 0;
   z4z = 0;
   Lx0  = 2.0;
   Ly0  = 2.0;
   Lz0  = 2.0;
   mux0 = 1.0;
   muy0 = 1.0;
   muz0 = 1.0;
end
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar (RED): coordenada x
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0x       posición
% z1x       velocidad
% z2x       aceleración
% z3x       sobre aceleración
% z4x       snap (derivada de la sobre aceleración)
% Lx0       Constante de Lipschitz (adaptable)
% yk(1,1)   posición medida en la coordenada "x"
% T         Tiempo de muestreo
% n = 4     Orden de diferenciación
%
% Párametros para adaptación
%
epsx         = 0.2;             % ruido promedio: 0.3 mts aprox
a            = 0.08;
% Ganancias
k0x          = 5.5;
k1x          = 10.03;
k2x          = 9.30;
k3x          = 4.57;
k4x          = 1.1;
%
% Series de Taylor para discretización del diferenciador
%
sum0x    = ((T^1)/1)*z1x + ((T^2)/(2*1))*z2x + ((T^3)/(3*2*1))*z3x +
((T^4)/(4*3*2*1))*z4x;
sum1x    = ((T^1)/1)*z2x + ((T^2)/(2*1))*z3x + ((T^3)/(3*2*1))*z4x;
sum2x    = ((T^1)/1)*z3x + ((T^2)/(2*1))*z4x;
sum3x    = ((T^1)/1)*z4x;
%
% Error de posición
%
sigx       = yk(1,1) - z0x;
%
% Error de velocidad (solo para gráficos de plano de estados)
sigdx      = yk(2,1) - z1x;
%
% Adaptación de constante de Lipschitz
%
ee    = 1.05;
phiLx = -16*((k0x^2)*(k4x^2)*(k3x^2)/(5*k2x))*abs(sigx)^(3/5)+...
        -((k1x^2)*(k4x^2)*(k3x^2)*(60*k1x+48*(k0x^2))/(5*k0x*k2x^2))*abs(sigx)^(4/5)...
```

```
        -(8*(k4x^2)*k3x*(k2x^2)/k2x)*abs(sigx)^(1/5)...
        +((24*(k0x^2)*(k4x^2)*(k3x^2))/(5*k2x))*abs(sigx)^(3/5)...
        +((12*(k1x^3)*(k4x^2)*(k3x^2))/(k0x*k2x^2))*abs(sigx)^(4/5)...
        +((48*(k1x^2)*(k0x^2)*(k4x^2)*(k3x^2))/(5*k0x*k2x^2))*abs(sigx)^(4/5)...
        -((8*(ee-1)*k1x*(k4x^2)*(k3x^2))/k2x)*abs(sigx)^(3/5)...
        -((4*k1x*(k4x^2)*(k3x^3))/k2x)*abs(sigx)^(3/5)...
        +(16*k2x*k3x*(k4x^2))*abs(sigx)^(3/5)...
        +(20*ee*k1x*(k4x^2)*(k3x^2)/k2x)*abs(sigx)^(2/5)...
        -((k4x^2)*(k3x^2)*(60*k1x+48*k0x^2)/k2x)*abs(sigx)^(4/5)...
        -(8*k1x*(k4x^2)*(k3x^3)/k2x)*abs(sigx)^(3/5);
phiux1  = -(20+4*a)*((k0x^2)*(k4x^2)*(k3x^2)/(5*k2x))*abs(sigx)^(17*a/5)+...
          +(20+4*a)*((k0x^2)*(k4x^2)*(k3x^2)/(5*k2x))*abs(sigx)^(34*a/5);
phiux2  = -(((40-
8*a)*ee*(k1x^3)*(k4x^2)*(k3x^2))/((5+a)*k0x*k2x^2))*abs(sigx)^((36*a)/10)...
          -(8*(k2x^2)*k3x*(k4x^2)/k2x)*abs(sigx)^((17*a)/5)...
          +((((40-
8*a))*ee*(k1x^3)*(k4x^2)*(k3x^2))/((5+a)*k0x*(k2x^2)))*abs(sigx)^(37*a/5)...
          -((8*(ee-1)*k1x*(k4x^2)*(k3x^2))/k2x)*abs(sigx)^((34*a+20)/10)...
          -(4*(k4x)*(k3x^3))*abs(sigx)^((17*a)/5)...
          +(16*(k2x)*(k3x)*(k4x^2))*abs(sigx)^((17*a)/5)...
          +((80*ee*k1x*(k3x^2)*(k4x^2))/((5+a)*k2x))*abs(sigx)^((34*a+5)/5);
if (abs(sigx) > 1)
    thtx = 0;
    wUx     = 0.00000015;
    b13     = 13*1.02^16;
    b7      = 7*1.02^16;
    mux0Dot = wUx*(b13*phiux1+b7*phiux2);
    mux0_1  = mux0 + T*mux0Dot;
    if mux0_1 < 1
       mux0_1 = 1;
    end
    Lx0_1   = Lx0;
    Lx0Dot  = 0;
elseif ((abs(sigx) <= 1) && (abs(sigx) > epsx))
    thtx   = 1;
    wLx     = 0.0005;
    b17     = 17*1^16;
    Lx0Dot  = wLx*b17*phiLx;
    Lx0_1   = Lx0  + T*Lx0Dot;
    if Lx0_1 < 1
       Lx0_1 = 1;
    end
    mux0_1  = mux0;
else
    thtx   = 1;
    wLx     = 0.00003;
    b17     = 13*1^16;
    Lx0Dot  = -wLx*b17*phiLx;
    Lx0_1   = Lx0  + T*Lx0Dot;% + T*w2x*(1-bbx)*k0x*abs(sigx)^(9/5);
    mux0_1  = mux0;
    if Lx0_1 < 1
       Lx0_1 = 1;
    end
end
%
% Diferenciador robusto exacto estándar adaptativo: coordenada x
%
z0x_1 = z0x + thtx*T*k0x*((Lx0_1)^(1))*((abs(sigx))^(4/5))*sign(sigx) + (1-
thtx)*T*k0x*(mux0_1^5)*((abs(sigx))^((5+a)/5))*sign(sigx) + sum0x;
z1x_1 = z1x + thtx*T*k1x*((Lx0_1)^(2))*((abs(sigx))^(3/5))*sign(sigx) + (1-
thtx)*T*k1x*(mux0_1^4)*((abs(sigx))^((5+(2*a))/5))*sign(sigx) + sum1x;
z2x_1 = z2x + thtx*T*k2x*((Lx0_1)^(3))*((abs(sigx))^(2/5))*sign(sigx) + (1-
thtx)*T*k2x*(14/5)*(mux0_1^3)*((abs(sigx))^((5+(3*a))/5))*sign(sigx) + sum2x;
z3x_1 = z3x + thtx*T*k3x*((Lx0_1)^(4))*((abs(sigx))^(1/5))*sign(sigx) + (1-
thtx)*T*k3x*(mux0_1^2)*((abs(sigx))^((5+(4*a))/5))*sign(sigx) + sum3x;
z4x_1 = z4x + thtx*T*k4x*((Lx0_1)^(5))*((abs(sigx))^(0/5))*sign(sigx) + (1-
thtx)*T*k4x*(mux0_1^1)*((abs(sigx))^((5+(5*a))/5))*sign(sigx);
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar (RED): coordenada y
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0y       posición
% z1y       velocidad
% z2y       aceleración
% z3y       sobre aceleración
```

```
% z4y       snap (derivada de la sobre aceleración)
% Ly0       Constante de Lipschitz (adaptable)
% yk(3,1)   posición medida
% T         Tiempo de muestreo
% n         Orden de diferenciación de bajo orden n  = r - 1
% n1        Orden de diferenciación de alto orden n1 = n+1 = r
%
% Párametros
%
epsy       = 0.375;            % umbral de ruido (mts)
k0y        = 5.5;
k1y        = 10.03;
k2y        = 9.30;
k3y        = 4.57;
k4y        = 1.1;
%
% Series de Taylor para discretización del diferenciador
%
sum0y  = ((T^1)/1)*z1y + ((T^2)/(2*1))*z2y + ((T^3)/(3*2*1))*z3y +
((T^4)/(4*3*2*1))*z4y;
sum1y  = ((T^1)/1)*z2y + ((T^2)/(2*1))*z3y + ((T^3)/(3*2*1))*z4y;
sum2y  = ((T^1)/1)*z3y + ((T^2)/(2*1))*z4y;
sum3y  = ((T^1)/1)*z4y;
%
% Error de posición
%
sigy   = yk(3,1) - z0y;
%
sigdy  = yk(4,1) - z1y;
%
% Adaptación de constante de Lipschitz
%
ee    = 1.05;
phiLy = -16*((k0y^2)*(k4y^2)*(k3y^2)/(5*k2y))*abs(sigy)^(3/5)+...
        -((k1y^2)*(k4y^2)*(k3y^2)*(60*k1y+48*(k0y^2))/(5*k0y*k2y^2))*abs(sigy)^(4/5)...
        -(8*(k4y^2)*k3y*(k2y^2)/k2y)*abs(sigy)^(1/5)...
        +((24*(k0y^2)*(k4y^2)*(k3y^2))/(5*k2y))*abs(sigy)^(3/5)...
        +((12*(k1y^3)*(k4y^2)*(k3y^2))/(k0y*k2y^2))*abs(sigy)^(4/5)...
        +((48*(k1y^2)*(k0y^2)*(k4y^2)*(k3y^2))/(5*k0y*k2y^2))*abs(sigy)^(4/5)...
        -((8*(ee-1)*k1y*(k4y^2)*(k3y^2))/k2y)*abs(sigy)^(3/5)...
        -((4*k1y*(k4y^2)*(k3y^3))/k2y)*abs(sigy)^(3/5)...
        +(16*k2y*k3y*(k4y^2))*abs(sigy)^(3/5)...
        +(20*ee*k1y*(k4y^2)*(k3y^2)/k2y)*abs(sigy)^(2/5)...
        -((k4y^2)*(k3y^2)*(60*k1y+48*k0y^2)/k2y)*abs(sigy)^(4/5)...
        -(8*k1y*(k4y^2)*(k3y^3)/k2y)*abs(sigy)^(3/5);
phiuy1  = -(20+4*a)*((k0y^2)*(k4y^2)*(k3y^2)/(5*k2y))*abs(sigy)^(17*a/5)+...
          +(20+4*a)*((k0y^2)*(k4y^2)*(k3y^2)/(5*k2y))*abs(sigy)^(34*a/5);
phiuy2  = -(((40-
8*a)*ee*(k1y^3)*(k4y^2)*(k3y^2))/((5+a)*k0y*k2y^2))*abs(sigy)^((36*a)/10)...
          -(8*(k2y^2)*k3y*(k4y^2)/k2y)*abs(sigy)^((17*a)/5)...
          +((((40-
8*a))*ee*(k1y^3)*(k4y^2)*(k3y^2))/((5+a)*k0y*(k2y^2)))*abs(sigy)^(37*a/5)...
          -((8*(ee-1)*k1y*(k4y^2)*(k3y^2))/k2y)*abs(sigy)^((34*a+20)/10)...
          -(4*(k4y)*(k3y^3))*abs(sigy)^((17*a)/5)...
          +(16*(k2y)*(k3y)*(k4y^2))*abs(sigy)^((17*a)/5)...
          +((80*ee*k1y*(k3y^2)*(k4y^2))/((5+a)*k2y))*abs(sigy)^((34*a+5)/5);
if (abs(sigy) > 1)
    thty = 0;
    wUy     = 0.000000020;
    b13     = 13*1.00^16;
    b7      = 7*1.00^16;
    muy0Dot = wUy*(b13*phiuy1+b7*phiuy2);
    muy0_1  = muy0 + T*muy0Dot;
    if muy0_1 < 1
       muy0_1 = 1;
    end
    Ly0_1   = Ly0;
    Ly0Dot  = 0;
elseif ((abs(sigy) <= 1) && (abs(sigy) > epsy))
    thty   = 1;
    wLy     = 0.00005;
    b17     = 17*1^16;
    Ly0Dot  = -wLy*b17*phiLy;
    Ly0_1   = Ly0  + T*Ly0Dot;
    if Ly0_1 < 1
       Ly0_1 = 1;
    end
```

```
    muy0_1  = muy0;
else
    thty   = 1;
    wLy     = 0.000009;
    b17     = 13*1^16;
    Ly0Dot  = wLy*b17*phiLy;
    Ly0_1   = Ly0  + T*Ly0Dot;% + T*w2y*(1-bby)*k0y*abs(sigy)^(9/5);
    muy0_1  = muy0;
    if Ly0_1 < 1
       Ly0_1 = 1;
    end
end
if (k>4650)
    Ly0_1  = 2.0;
    muy0_1 = 1.0;
end
%
% Diferenciador robusto exacto estándar adaptativo: coordenada y
%
z0y_1 = z0y + thty*T*k0y*((Ly0_1)^(1))*((abs(sigy))^(4/5))*sign(sigy) + (1-
thty)*T*k0y*(muy0_1^5)*((abs(sigy))^((5+a)/5))*sign(sigy) + sum0y;
z1y_1 = z1y + thty*T*k1y*((Ly0_1)^(2))*((abs(sigy))^(3/5))*sign(sigy) + (1-
thty)*T*k1y*(muy0_1^4)*((abs(sigy))^((5+(2*a))/5))*sign(sigy) + sum1y;
z2y_1 = z2y + thty*T*k2y*((Ly0_1)^(3))*((abs(sigy))^(2/5))*sign(sigy) + (1-
thty)*T*k2y*(muy0_1^3)*((abs(sigy))^((5+(3*a))/5))*sign(sigy) + sum2y;
z3y_1 = z3y + thty*T*k3y*((Ly0_1)^(4))*((abs(sigy))^(1/5))*sign(sigy) + (1-
thty)*T*k3y*(muy0_1^2)*((abs(sigy))^((5+(4*a))/5))*sign(sigy) + sum3y;
z4y_1 = z4y + thty*T*k4y*((Ly0_1)^(5))*((abs(sigy))^(0/5))*sign(sigy) + (1-
thty)*T*k4y*(muy0_1^1)*((abs(sigy))^((5+(5*a))/5))*sign(sigy);
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar (RED): coordenada z
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0z       posición
% z1z       velocidad
% z2z       aceleración
% z3z       sobre aceleración
% z4z       snap (derivada de la sobre aceleración)
% Lz0       Constante de Lipschitz (adaptable)
% yk(5,1)   posición medida
% T         Tiempo de muestreo
% n         Orden de diferenciación de bajo orden n  = r - 1
% n1        Orden de diferenciación de alto orden n1 = n+1 = r
%
% Párametros lambda (Homogeneidad de coeficientes)
%
%
% Párametros
%
epsz    = 0.3;
k0z     = 5.5*1.0;
k1z     = 10.03*1.0;
k2z     = 9.30*1.86;
k3z     = 4.57*0.96;
k4z     = 1.1*1.1;
%
% Series de Taylor para discretización del diferenciador
%
sum0z  = ((T^1)/1)*z1z + ((T^2)/(2*1))*z2z + ((T^3)/(3*2*1))*z3z +
((T^4)/(4*3*2*1))*z4z;
sum1z  = ((T^1)/1)*z2z + ((T^2)/(2*1))*z3z + ((T^3)/(3*2*1))*z4z;
sum2z  = ((T^1)/1)*z3z + ((T^2)/(2*1))*z4z;
sum3z  = ((T^1)/1)*z4z;
%
% Error de posición
%
sigz       = yk(5,1) - z0z;
%
sigdz      = yk(6,1) - z1z;
%
% Adaptación de constante de Lipschitz
%
ee     = 1.01;
phiLz = -16*((k0z^2)*(k4z^2)*(k3z^2)/(5*k2z))*abs(sigz)^(3/5)+...
        -((k1z^2)*(k4z^2)*(k3z^2)*(60*k1z+48*(k0z^2))/(5*k0z*k2z^2))*abs(sigz)^(4/5)...
```

```
        -(8*(k4z^2)*k3z*(k2z^2)/k2z)*abs(sigz)^(1/5)...
        +((24*(k0z^2)*(k4z^2)*(k3z^2))/(5*k2z))*abs(sigz)^(3/5)...
        +((12*(k1z^3)*(k4z^2)*(k3z^2))/(k0z*k2z^2))*abs(sigz)^(4/5)...
        +((48*(k1z^2)*(k0z^2)*(k4z^2)*(k3z^2))/(5*k0z*k2z^2))*abs(sigz)^(4/5)...
        -((8*(ee-1)*k1z*(k4z^2)*(k3z^2))/k2z)*abs(sigz)^(3/5)...
        -((4*k1z*(k4z^2)*(k3z^3))/k2z)*abs(sigz)^(3/5)...
        +(16*k2z*k3z*(k4z^2))*abs(sigz)^(3/5)...
        +(20*ee*k1z*(k4z^2)*(k3z^2)/k2z)*abs(sigz)^(2/5)...
        -((k4z^2)*(k3z^2)*(60*k1z+48*k0z^2)/k2z)*abs(sigz)^(4/5)...
        -(8*k1z*(k4z^2)*(k3z^3)/k2z)*abs(sigz)^(3/5);
phiuz1  = -(20+4*a)*((k0z^2)*(k4z^2)*(k3z^2)/(5*k2z))*abs(sigz)^(17*a/5)+...
          +(20+4*a)*((k0z^2)*(k4z^2)*(k3z^2)/(5*k2z))*abs(sigz)^(34*a/5);
phiuz2  = -(((40-
8*a)*ee*(k1z^3)*(k4z^2)*(k3z^2))/((5+a)*k0z*k2z^2))*abs(sigz)^((36*a)/10)...
          -(8*(k2z^2)*k3z*(k4z^2)/k2z)*abs(sigz)^((17*a)/5)...
          +((((40-
8*a))*ee*(k1z^3)*(k4z^2)*(k3z^2))/((5+a)*k0z*(k2z^2)))*abs(sigz)^(37*a/5)...
          -((8*(ee-1)*k1z*(k4z^2)*(k3z^2))/k2z)*abs(sigz)^((34*a+20)/10)...
          -(4*(k4z)*(k3z^3))*abs(sigz)^((17*a)/5)...
          +(16*(k2z)*(k3z)*(k4z^2))*abs(sigz)^((17*a)/5)...
          +((80*ee*k1z*(k3z^2)*(k4z^2))/((5+a)*k2z))*abs(sigz)^((34*a+5)/5);
if (abs(sigz) > 1)
    thtz = 0;
    wUz     = 0.000015;
    b13     = 13*1.02^16;
    b7      = 7*1.02^16;
    muz0Dot = wUz*(b13*phiuz1+b7*phiuz2);
    muz0_1  = muz0 + T*muz0Dot;
    if muz0_1 < 1
       muz0_1 = 1;
    end
    Lz0_1   = Lz0;
    Lz0Dot  = 0;
elseif ((abs(sigz) <= 1) && (abs(sigz) > epsz))
    thtz   = 1;
    wLz     = 0.000035;
    b17     = 17*1.01^16;
    Lz0Dot  = wLz*b17*phiLz;
    Lz0_1   = Lz0  + T*Lz0Dot;
    if Lz0_1 < 1
       Lz0_1 = 1;
    end
    muz0_1  = muz0;
else
    thtz   = 1;
    wLz     = 0.000003;
    b17     = 17*1.01^16;
    Lz0Dot  = -wLz*b17*phiLz;
    Lz0_1   = Lz0  + T*Lz0Dot;% + T*w2z*(1-bbz)*k0z*abs(sigz)^(9/5);
    muz0_1  = muz0;
    if Lz0_1 < 1
       Lz0_1 = 1;
    end
end
%
% Diferenciador robusto exacto estándar adaptativo: coordenada z
%
z0z_1 = z0z + thtz*T*k0z*((Lz0_1)^(1))*((abs(sigz))^(4/5))*sign(sigz) + (1-
thtz)*T*k0z*(muz0_1^5)*((abs(sigz))^((5+a)/5))*sign(sigz) + sum0z;
z1z_1 = z1z + thtz*T*k1z*((Lz0_1)^(2))*((abs(sigz))^(3/5))*sign(sigz) + (1-
thtz)*T*k1z*(muz0_1^4)*((abs(sigz))^((5+(2*a))/5))*sign(sigz) + sum1z;
z2z_1 = z2z + thtz*T*k2z*((Lz0_1)^(3))*((abs(sigz))^(2/5))*sign(sigz) + (1-
thtz)*T*k2z*(muz0_1^3)*((abs(sigz))^((5+(3*a))/5))*sign(sigz) + sum2z;
z3z_1 = z3z + thtz*T*k3z*((Lz0_1)^(4))*((abs(sigz))^(1/5))*sign(sigz) + (1-
thtz)*T*k3z*(muz0_1^2)*((abs(sigz))^((5+(4*a))/5))*sign(sigz) + sum3z;
z4z_1 = z4z + thtz*T*k4z*((Lz0_1)^(5))*((abs(sigz))^(0/5))*sign(sigz) + (1-
thtz)*T*k4z*(muz0_1^1)*((abs(sigz))^((5+(5*a))/5))*sign(sigz);
%-------------------------------------------------------------------------
% Guardar variables para siguiente iteración
%-------------------------------------------------------------------------
Lx0  = Lx0_1;
Ly0  = Ly0_1;
Lz0  = Lz0_1;
mux0 = mux0_1;
muy0 = muy0_1;
muz0 = muz0_1;
z0x  = z0x_1;
```

```
z1x  = z1x_1;
z2x  = z2x_1;
z3x  = z3x_1;
z4x  = z4x_1;
z0y  = z0y_1;
z1y  = z1y_1;
z2y  = z2y_1;
z3y  = z3y_1;
z4y  = z4y_1;
z0z  = z0z_1;
z1z  = z1z_1;
z2z  = z2z_1;
z3z  = z3z_1;
z4z  = z4z_1;
xhk  = [z0x;z1x;z2x;z0y;z1y;z2y;z0z;z1z;z2z];
uhk  = [z3x;z3y;z3z];
duhk = [z4x;z4y;z4z];
sigk = [sigx;sigdx;sigy;sigdy;sigz;sigdz];
Lk   = [Lx0_1;Ly0_1;Lz0_1];
MUk    = [mux0_1,muy0_1,muz0_1];
thetak = [thtx,thty,thtz];
end
```

Apéndice 2.11: Simulación de la trayectoria del misil (ARED)

```
%=========================================================================
%
% Título            : sim_ARED.m
% Propósito         : Simulación de diferenciador robusto exacto
%
% Descripción       : Este programa permite la estimación de parámetros de
%                     posición, velocidad, aceleración, sobre aceleración
%                     y derivada de la sobre aceleración de un blanco aéreo
%                     de alta maniobrabilidad a partir de las mediciones
%                     ruidosas de posición en coordenadas cartesianas
%                     obtenidas por un radar de seguimiento automático.
%
% Fecha de creación : 11/10/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
clear all;
clc;
close all;
%-------------------------------------------------------------------------
% Inicialización de variables
%-------------------------------------------------------------------------
RMSEcold = 0;
RMSEpold = 0;
%-------------------------------------------------------------------------
% Parámetros iniciales de radar (sensor)
%-------------------------------------------------------------------------
T     = 1/50;                       % Período de muestreo (seg)
%-------------------------------------------------------------------------
% Definición de constantes físicas
%-------------------------------------------------------------------------
g     = 9.81;                       % gravedad (m/s2)
M     = 343;                        % velocidad del sonido (m/s)
load_factor = 100*g;                % máxima sobre aceleración (g/s)
%-------------------------------------------------------------------------
% Definición de vectores de tiempo
%-------------------------------------------------------------------------
ti    = 0;                          % Tiempo inicial de simulación
tff   = 104.5;                      % Tiempo final de simulación
tt    = ti:T:tff;                   % Vector de tiempo
tt    = tt';                        % Transpuesta del vector de tiempo
nt    = length(tt);                 % Número de muestras
%-------------------------------------------------------------------------
% Definición de posición y velocidad inicial del blanco en
% coordenadas esféricas
%-------------------------------------------------------------------------
Ad_init  = 35000;                       % Distancia al blanco (m)
```

```
Ed_init  = 0.01;                          % Elevación al blanco (°)
Bdn_init = 30;                            % Marcación al blanco (°)
Vm_init  = 0.9*M;                         % Velocidad del blanco (1 Mach = 343 m/s)
Rv_init  = 210;                           % Rumbo inicial del blanco (°)
%-------------------------------------------------------------------------
% Conversión de coordenadas esféricas a cartesianas (North-Sky-East) de la
% posición y velocidad inicial del blanco
%-------------------------------------------------------------------------
[Adx_init, Ady_init, Adz_init] = p2c(Ad_init,Ed_init,Bdn_init);
[Vmx_init, Vmy_init, Vmz_init] = p2c(Vm_init,0,Rv_init);
%-------------------------------------------------------------------------
% Inicialización de vector de estados
%-------------------------------------------------------------------------
xk = [Adx_init Vmx_init 0*g Ady_init Vmy_init 0*g Adz_init Vmz_init 0*g]';
%-------------------------------------------------------------------------
% Definición de matrices del modelo de espacio de estados del sistema
%-------------------------------------------------------------------------
% Matriz de medición (Pulse radar)
Ck     = [1 0 0 0 0 0 0 0 0;        % pos x
          0 0 0 0 0 0 0 0 0;        % vel x
          0 0 0 1 0 0 0 0 0;        % pos y
          0 0 0 0 0 0 0 0 0;        % vel y
          0 0 0 0 0 0 0 1 0;        % pos z
          0 0 0 0 0 0 0 0 0];       % vel z

% Cálculo de matriz de transición Fk
Fk = [1 T (T^2)/2 0 0       0 0 0       0;
      0 1       T 0 0       0 0 0       0;
      0 0       1 0 0       0 0 0       0;
      0 0       0 1 T (T^2)/2 0 0       0;
      0 0       0 0 1       T 0 0       0;
      0 0       0 0 0       1 0 0       0;
      0 0       0 0 0       0 1 T (T^2)/2;
      0 0       0 0 0       0 0 1       T;
      0 0       0 0 0       0 0 0       1];

% Cálculo de la matriz de entrada Gk
Gk = [(T^3)/6       0         0;
      (T^2)/2       0         0;
       T            0         0;
       0       (T^3)/6        0;
       0       (T^2)/2        0;
       0            T         0;
       0            0   (T^3)/6;
       0            0   (T^2)/2;
       0            0         T];

% Matriz de distribución de perturbaciones (aceleraciones)
Dk = [75e4*(T^3)/6       0                0;
      5e3*(T^2)/2        0                0;
       1e2*T             0                0;
       0      75e4*(T^3)/6                0;
       0       5e3*(T^2)/2                0;
       0            1e2*T                 0;
       0                 0     75e4*(T^3)/6;
       0                 0      5e3*(T^2)/2;
       0                 0           1e2*T];
%-------------------------------------------------------------------------
% Generación del vector de sobre aceleraciones de entrada uk
%-------------------------------------------------------------------------
uk = gen_jerk_sea_skimming3(nt,load_factor,g,T);
%-------------------------------------------------------------------------
% Generación del vector de perturbaciones ?k
%-------------------------------------------------------------------------
pk = perturb_1(g,ti,T,tff);
%-------------------------------------------------------------------------
% Cargar ruido glint
%-------------------------------------------------------------------------
load glint_puntos.mat
%-------------------------------------------------------------------------
% Simulación del modelo
%-------------------------------------------------------------------------
disp('Diferenciador Robusto Exacto Estándar Adaptativo (ARED)');
prompt = 'Para simular con ruido presione "1". Para simular sin ruido presione "0": ';
gl     = input(prompt);
for k = 1:nt
```

```
     % Modelo dinámico o de transición de estados
     xk       = Fk*xk + Gk*uk(:,k) + Dk*pk(:,k);
     % Simulación de adquisición de datos por el radar
     % Datos de posición adquiridos en coordenadas esféricas
     [Ad,Bdn,Ed] = c2p(xk);
     % Modelo de medición polar a cartesiano
     yk = yk_noise(Ad,Bdn,Ed,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,gl,k);
     % Diferenciador Robusto Exacto Adaptativo
     tic
     [xhk,uhk,duhk,Lk,MUk,sigk] = ARED(yk,T,k);
     TCPU = toc;

     % Conversión de coordenada estimadas cartesianas a coordenadas polares
     [Adhat,Bdnhat,Edhat] = c2p(xhk);

     % Cálculo de errores en coordenadas cartesianas
     errorc       = xhk - xk;

     % Cálculo de errores en coordenadas esféricas
     errorAd      = Adhat - Ad;                                    %mts
     errorBdn     = Bdnhat*(pi*1000/180) - Bdn*(pi*1000/180);
     errorEd      = Edhat*(pi*1000/180) - Ed*(pi*1000/180);
     errorp       = [errorAd; errorBdn; errorEd];
    % Calculo de error RMSE
    if ( k >= 4650)
         RMSEc         = RMSEcold + ((errorc).^2);
         RMSEp         = RMSEpold + ((errorp).^2);
     else
         RMSEc         = 0;
         RMSEp         = 0;
    end
     bin_log(:,k)       = k;
     xk_log(:,k)        = xk;
     xhk_log(:,k)       = xhk;
     uhk_log(:,k)       = uhk;
     duhk_log(:,k)      = duhk;
     sigk_log(:,k)      = sigk;
     errorc_log(:,k)    = errorc;
     errorp_log(:,k)    = errorp;
     polar_log(:,k)     = [Ad;Bdn;Ed];
     polarhat_log(:,k)  = [Adhat;Bdnhat;Edhat];
     yk_log(:,k)        = yk;
     TCPU_log(:,k)      = TCPU;
     L_log(:,k)         = Lk;
     MU_log(:,k)        = MUk;
     % Guardar variables
     RMSEcold           = RMSEc;
     RMSEpold           = RMSEp;
end
RMSEctot = sqrt(RMSEc)/(nt-4650);
RMSEptot = sqrt(RMSEp)/(nt-4650);
%-------------------------------------------------------------------------
% Conversión de vector de estados reales a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
% 0.10197 - de mts/seg2 a g's
conv_xk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197]';
xk_log = xk_log.*conv_xk;
%-------------------------------------------------------------------------
% Conversión de vector de estados medidos a Mn, Mach
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
conv_yk   = [ 0.00054 0.00292 0.00054 0.00292 0.00054 0.00292 ]';
yk_log    = yk_log.*conv_yk;
%-------------------------------------------------------------------------
% Conversión de vector de estados estimados a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - mn
% 0.00292 - Mach
% 0.10197 - g's
conv_xhk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292
0.10197]';
xhk_log = xhk_log.*conv_xhk;
%-------------------------------------------------------------------------
% Conversión de vector de entrada real y estimado a g/s
```

```
%-------------------------------------------------------------------------
uk_log    = uk./g;
uhk_log   = uhk_log./g;
duhk_log   = duhk_log./g;
%-------------------------------------------------------------------------
% Conversión de perturbaciones a g/s
%-------------------------------------------------------------------------
pk_log    = pk./g;
%-------------------------------------------------------------------------
% Plots
%-------------------------------------------------------------------------
%%
figure(1);p1 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                0,                0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p1(1),'LineWidth',2.0);
set(p1(2),'LineWidth',1.8);
set(p1(3),'LineWidth',1.5);
set(p1(4),'LineWidth',2.0);
set(p1(5),'LineWidth',2.0);
set(p1(6),'LineWidth',2.0);
set(p1(7),'LineWidth',2.0);

axis([-5 20 -5 10 0 0.04]);          % Toda la trayectoria
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Trayectoria sea skimming del misil','FontSize',16)
legend({'mediciones','real','ARED','Posición del buque observador','Posición inicial del
misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
%% 3D trayectoria completa

figure(2);p2 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-r',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-g',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                0,                0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p2(1),'LineWidth',2.0);
set(p2(2),'LineWidth',1.8);
set(p2(3),'LineWidth',1.5);
set(p2(4),'LineWidth',2.0);
set(p2(5),'LineWidth',2.0);
set(p2(6),'LineWidth',2.0);
set(p2(7),'LineWidth',2.0);
axis([4 8 -2 5 0 0.016]);
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Maniobra terminal del misil','FontSize',16)
legend({'mediciones','real','ARED','Posición del buque observador','Posición inicial del
misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14);

%% Coordenada x
% Posición x: Trayectoria completa
figure(3);
subplot(1,2,1);
p3 = plot(bin_log(1,:),yk_log(1,:),'-r',bin_log(1,:),xk_log(1,:),'-
g',bin_log(1,:),xhk_log(1,:),'-k');
title('Trayectoria completa del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p3(1),'LineWidth', 2.0);
```

```
set(p3(2),'LineWidth', 1.8);
set(p3(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(1,:) yk_log(1,:)]) max([xk_log(1,:) yk_log(1,:)])])

% Posición x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p4 = plot(bin_log(1,lpx1:nt),yk_log(1,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(1,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(1,lpx1:nt),'-
k');
title('Maniobra terminal del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p4(1),'LineWidth', 2.0);
set(p4(2),'LineWidth', 1.8);
set(p4(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)]) max([xk_log(1,lpx1:nt)
yk_log(1,lpx1:nt)])])
%%
% Velocidad x: Trayectoria completa
figure(4);
subplot(1,2,1);
p5 = plot(bin_log(1,:),xk_log(2,:),'-g',bin_log(1,:),xhk_log(2,:),'-k');
title('Trayectoria completa del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p5(1),'LineWidth', 1.8);
set(p5(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(2,:)) max(xk_log(2,:))])
%axis([1 nt -10 10])
% Velocidad x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p6 = plot(bin_log(1,lpx1:nt),xk_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(2,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p6(1),'LineWidth', 1.8);
set(p6(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt -1 2]) % con ruido
%axis([lpx1 nt min([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])
max([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])])
%%
% Aceleración x: Trayectoria completa
figure(5);
subplot(1,2,1);
p7 = plot(bin_log(1,:),xk_log(3,:),'-g',bin_log(1,:),xhk_log(3,:),'-k');
title('Trayectoria completa del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',2,'FontSize',18)
set(p7(1),'LineWidth', 1.8);
set(p7(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(3,:)) max(xk_log(3,:))])
%axis([1 nt min([xk_log(3,:) xhk_log(3,:)]) max([xk_log(3,:) xhk_log(3,:)])])
% Aceleración x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2)
p8 = plot(bin_log(1,lpx1:nt),xk_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(3,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',2,'FontSize',18)
set(p8(1),'LineWidth', 1.8);
set(p8(2),'LineWidth', 1.5);
grid on
```

```
axis([lpx1 nt min([xk_log(3,lpx1:nt) xhk_log(3,lpx1:nt)]) max([xk_log(3,lpx1:nt)
xhk_log(3,lpx1:nt)])])
%axis([lpx1 nt -20 20])
%% Coordenada y
% Posición y: Trayectoria completa
figure(6);
subplot(1,2,1);
p9 = plot(bin_log(1,:),yk_log(3,:),'-r',bin_log(1,:),xk_log(4,:),'-
g',bin_log(1,:),xhk_log(4,:),'-k');
title('Trayectoria completa del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p9(1),'LineWidth', 2.0);
set(p9(2),'LineWidth', 1.8);
set(p9(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(4,:) yk_log(3,:)]) max([xk_log(4,:) yk_log(3,:)])])

% Posición y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p10 = plot(bin_log(1,lpx1:nt),yk_log(3,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(4,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhk_log(4,lpx1:nt),'-
K');
title('Maniobra terminal del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p10(1),'LineWidth', 2.0);
set(p10(2),'LineWidth', 1.8);
set(p10(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])
max([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])])

% Velocidad y: Trayectoria completa
figure(7);
subplot(1,2,1);
p11 = plot(bin_log(1,:),xk_log(5,:),'-g',bin_log(1,:),xhk_log(5,:),'-k');
title('Trayectoria completa del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','ARED'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p11(1),'LineWidth', 1.8);
set(p11(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(5,:)) max(xk_log(5,:))])
%axis([1 nt -10 10])
% Velocidad y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p12 = plot(bin_log(1,lpx1:nt),xk_log(5,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(5,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p12(1),'LineWidth', 1.8);
set(p12(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min(xk_log(5,lpx1:nt)) max(xk_log(5,lpx1:nt))])

figure(8);
% Aceleración y: Trayectoria
subplot(1,2,1);
p14 = plot(bin_log(1,:),xk_log(6,:),'-g',bin_log(1,:),xhk_log(6,:),'-k');
title('Maniobra terminal del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',2,'FontSize',18)
set(p14(1),'LineWidth', 1.8);
set(p14(2),'LineWidth', 1.5);
grid on
%axis([1 nt min([xk_log(6,:) xhk_log(6,:)]) max([xk_log(6,:) xhk_log(6,:)])])
axis([1 nt min(xk_log(6,:)) max(xk_log(6,:))])
% Aceleración y: Maniobra terminal
```

```
subplot(1,2,2);
p13 = plot(bin_log(1,lpx1:nt),xk_log(6,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(6,lpx1:nt),'-k');
title('Trayectoria completa del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',2,'FontSize',18)
set(p13(1),'LineWidth', 1.8);
set(p13(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(6,lpx1:nt) xhk_log(6,lpx1:nt)]) max([xk_log(6,lpx1:nt)
xhk_log(6,lpx1:nt)])])

%% Coordenada z
% Posición z: Trayectoria completa
figure(9);
subplot(1,2,1);
p15 = plot(bin_log(1,:),yk_log(5,:)./0.00054,'-r',bin_log(1,:),xk_log(7,:)./0.00054,'-
g',bin_log(1,:),xhk_log(7,:)./0.00054,'-k');
title('Trayectoria completa del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'medición','real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p15(1),'LineWidth', 2.0);
set(p15(2),'LineWidth', 1.8);
set(p15(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(7,:)./0.00054 yk_log(5,:)./0.00054]) max([xk_log(7,:)./0.00054
yk_log(5,:)./0.00054])])

% Posición z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p16 = plot(bin_log(1,lpx1:nt),yk_log(5,lpx1:nt)./0.00054,'-
r',bin_log(1,lpx1:nt),xk_log(7,lpx1:nt)./0.00054,'-
g',bin_log(1,lpx1:nt),xhk_log(7,lpx1:nt)./0.00054,'-k');
title('Maniobra terminal del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mts]','FontSize',22)
legend({'medición','real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p16(1),'LineWidth', 2.0);
set(p16(2),'LineWidth', 1.8);
set(p16(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054]) max([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054])])
%%
% Velocidad z: Trayectoria completa
figure(10);
subplot(1,2,1);
p17 = plot(bin_log(1,:),xk_log(8,:),'-g',bin_log(1,:),xhk_log(8,:),'-k');
title('Trayectoria completa del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','ARED'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p17(1),'LineWidth', 1.8);
set(p17(2),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(8,:) xhk_log(8,:)]) max([xk_log(8,:) xhk_log(8,:)])])
%axis([1 nt -4 4])
% Velocidad z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p18 = plot(bin_log(1,lpx1:nt),xk_log(8,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(8,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',3,'FontSize',18)
set(p18(1),'LineWidth', 1.8);
set(p18(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(8,lpx1:nt) xhk_log(8,lpx1:nt)]) max([xk_log(8,lpx1:nt)
xhk_log(8,lpx1:nt)])])
%%
% Aceleración z: Trayectoria completa
```

```
figure(11);
subplot(1,2,1);
p19 = plot(bin_log(1,:),xk_log(9,:),'-g',bin_log(1,:),xhk_log(9,:),'-k');
title('Trayectoria completa del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',2,'FontSize',18)
set(p19(1),'LineWidth', 1.8);
set(p19(2),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(9,:)) max(xk_log(9,:))])
%axis([1 nt min([xk_log(9,:) xhk_log(9,:)]) max([xk_log(9,:) xhk_log(9,:)])])

% Aceleración z: Maniobra terminal
subplot(1,2,2);
p20 = plot(bin_log(1,lpx1:nt),xk_log(9,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(9,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','ARED'},'Location','north','NumColumns',2,'FontSize',18)
set(p20(1),'LineWidth', 1.8);
set(p20(2),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(9,lpx1:nt) xhk_log(9,lpx1:nt)]) max([xk_log(9,lpx1:nt)
xhk_log(9,lpx1:nt)])])

%% Plots señales de errores cartesianos posición x,y,z
figure(12);
sgtitle('Errores cartesianos','FontSize',18)
lpx1 = 4500;
subplot(3,2,1);
p21 = plot(bin_log(1,:),errorc_log(1,:),'-g',bin_log(1,:),errorc_log(4,:),'-
c',bin_log(1,:),errorc_log(7,:),'-m');
title('Trayectoria completa: posición','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m]','FontSize',16)
legend({'pos x','pos y','pos z'},'Location','north','NumColumns',3,'FontSize',14)
set(p21,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(1,:) errorc_log(4,:) errorc_log(7,:)]) max([errorc_log(1,:)
errorc_log(4,:) errorc_log(7,:)])])

subplot(3,2,2);
p22 = plot(bin_log(1,lpx1:nt),errorc_log(1,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(4,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(7,lpx1:nt),'-m');
title('Maniobra terminal: posición','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p22,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])
max([errorc_log(1,lpx1:nt) errorc_log(4,lpx1:nt) errorc_log(7,lpx1:nt)])])
% Plots señales de errores cartesianos velocidad x,y,z
subplot(3,2,3);
p23 = plot(bin_log(1,:),errorc_log(2,:),'-g',bin_log(1,:),errorc_log(5,:),'-
c',bin_log(1,:),errorc_log(8,:),'-m');
title('Trayectoria completa: velocidad','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s]','FontSize',16)
legend({'vel x','vel y','vel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p23,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(2,:) errorc_log(5,:) errorc_log(8,:)]) max([errorc_log(2,:)
errorc_log(5,:) errorc_log(8,:)])])

subplot(3,2,4);
p24 = plot(bin_log(1,lpx1:nt),errorc_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(5,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(8,lpx1:nt),'-m');
title('Maniobra terminal: velocidad','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s]','FontSize',16);
set(p24,'LineWidth',1.5);
grid on
```

```
axis([lpx1 nt min([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])
max([errorc_log(2,lpx1:nt) errorc_log(5,lpx1:nt) errorc_log(8,lpx1:nt)])])
% Plots señales de errores cartesianos aceleración x,y,z
subplot(3,2,5);
p25 = plot(bin_log(1,:),errorc_log(3,:),'-g',bin_log(1,:),errorc_log(6,:),'-
c',bin_log(1,:),errorc_log(9,:),'-m');
title('Trayectoria completa: aceleración','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s2]','FontSize',16)
legend({'acel x','acel y','acel z'},'Location','north','NumColumns',3,'FontSize',14)
set(p25,'LineWidth',1.5);
grid on
axis([1 nt min([errorc_log(3,:) errorc_log(6,:) errorc_log(9,:)]) max([errorc_log(3,:)
errorc_log(6,:) errorc_log(9,:)])])

subplot(3,2,6);
p26 = plot(bin_log(1,lpx1:nt),errorc_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),errorc_log(6,lpx1:nt),'-
c',bin_log(1,lpx1:nt),errorc_log(9,lpx1:nt),'-m');
title('Maniobra terminal: aceleración','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s2]','FontSize',16);
set(p26,'LineWidth',1.5);
grid on
axis([lpx1 nt min([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])
max([errorc_log(3,lpx1:nt) errorc_log(6,lpx1:nt) errorc_log(9,lpx1:nt)])])

%% Plots señales de error polares: Distancia Ad
figure(13);
sgtitle('Errores polares','FontSize',18)
maxAd =  37.5*ones(1,nt);
minAd = -37.5*ones(1,nt);

subplot(3,2,1);
laz1 = 4500;
p27 = plot(bin_log(1,:),errorp_log(1,:),'-g',bin_log(1,:),maxAd(1,:),'--
r',bin_log(1,:),minAd(1,:),'--r');
title('Trayectoria completa: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
legend({'Ad','Límite +/- 37.5 mts'},'NumColumns',2,'Location','North','FontSize',12);
set(p27(1),'LineWidth',1.5);
set(p27(2),'LineWidth',1.0);
set(p27(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(1,:) minAd(1,:)]) max([errorp_log(1,:) maxAd(1,:)])])

subplot(3,2,2);
p28 = plot(bin_log(1,laz1:nt),errorp_log(1,laz1:nt),'-
g',bin_log(1,laz1:nt),maxAd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minAd(1,laz1:nt),'--r');
title('Maniobra terminal: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p28(1),'LineWidth',1.5);
set(p28(2),'LineWidth',1.0);
set(p28(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(1,laz1:nt) minAd(1,laz1:nt)]) max([errorp_log(1,laz1:nt)
maxAd(1,laz1:nt)])])
% Error polar en acimut
maxBdn = 2*ones(1,nt);
minBdn = -2*ones(1,nt);

subplot(3,2,3);
laz1 = 4500;
p29 = plot(bin_log(1,:),errorp_log(2,:),'-g',bin_log(1,:),maxBdn(1,:),'--
r',bin_log(1,:),minBdn(1,:),'--r');
title('Trayectoria completa: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Bdn','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p29(1),'LineWidth',1.5);
set(p29(2),'LineWidth',1.0);
set(p29(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(2,:) minBdn(1,:)]) max([errorp_log(2,:) maxBdn(1,:)])])
```

```
subplot(3,2,4);
p30 = plot(bin_log(1,laz1:nt),errorp_log(2,laz1:nt),'-
g',bin_log(1,laz1:nt),maxBdn(1,laz1:nt),'--r',bin_log(1,laz1:nt),minBdn(1,laz1:nt),'--
r');
title('Maniobra terminal: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p30(1),'LineWidth',1.5);
set(p30(2),'LineWidth',1.0);
set(p30(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(2,laz1:nt) minBdn(1,laz1:nt)]) max([errorp_log(2,laz1:nt)
maxBdn(1,laz1:nt)])])

maxEd = 2*ones(1,nt);
minEd = -2*ones(1,nt);

subplot(3,2,5);
laz1 = 4500;
p31 = plot(bin_log(1,:),errorp_log(3,:),'-g',bin_log(1,:),maxEd(1,:),'--
r',bin_log(1,:),minEd(1,:),'--r');
title('Trayectoria completa: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Ed','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p31(1),'LineWidth',1.5);
set(p31(2),'LineWidth',1.0);
set(p31(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(3,:) minEd(1,:)]) max([errorp_log(3,:) maxEd(1,:)])])

subplot(3,2,6);
p32 = plot(bin_log(1,laz1:nt),errorp_log(3,laz1:nt),'-
g',bin_log(1,laz1:nt),maxEd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minEd(1,laz1:nt),'--r');
title('Maniobra terminal: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p32(1),'LineWidth',1.5);
set(p32(2),'LineWidth',1.0);
set(p32(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(3,laz1:nt) minEd(1,laz1:nt)]) max([errorp_log(3,laz1:nt)
maxEd(1,laz1:nt)])])
%% Estimación de vector de entrada de control
figure(25);
p44 = plot(bin_log(1,:),uk_log(1,:),'-g',bin_log(1,:),uhk_log(1,:),'-
k',bin_log(1,:),pk_log(1,:),'-m');
title('Trayectoria del misil: sobre aceleración x','FontSize',16);
legend({'real uk','ARED uhatk','real
pk'},'Location','North','NumColumns',3,'FontSize',14);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p44(1),'LineWidth',1.5);
set(p44(2),'LineWidth',1.5);
set(p44(3),'LineWidth',1.5);
grid on
axis([1 nt min([uk_log(1,:) uhk_log(1,:)]) max([uk_log(1,:) uhk_log(1,:)])])

figure(26);
p45 = plot(bin_log(1,:),uk_log(2,:),'-g',bin_log(1,:),uhk_log(2,:),'-
k',bin_log(1,:),pk_log(2,:),'-m');
title('Trayectoria del misil: sobre aceleración y','FontSize',16);
legend({'real uk','ARED uhatk','real
pk'},'Location','North','NumColumns',3,'FontSize',14);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p45(1),'LineWidth',1.5);
set(p45(2),'LineWidth',1.5);
set(p45(3),'LineWidth',1.5);
grid on
axis([1 nt min([uk_log(2,:) uhk_log(2,:)]) max([uk_log(2,:) uhk_log(2,:)])])

figure(27);
p46 = plot(bin_log(1,:),uk_log(3,:),'-g',bin_log(1,:),uhk_log(3,:),'-
k',bin_log(1,:),pk_log(3,:),'-m');
title('Trayectoria del misil: sobre aceleración z','FontSize',16);
```

```
legend({'real uk','ARED uhatk','real
pk'},'Location','North','NumColumns',3,'FontSize',14);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p46(1),'LineWidth',1.5);
set(p46(2),'LineWidth',1.5);
set(p46(3),'LineWidth',1.5);
grid on
axis([1 nt min([uk_log(3,:) uhk_log(3,:)]) max([uk_log(3,:) uhk_log(3,:)])])
%% Gráfica de plano de estados (colector deslizante en el origen)
figure(28);
subplot(3,2,1);
laz1 = 500;
p25 = plot(sigk_log(1,1:laz1),sigk_log(4,1:laz1),'-b');
title('Plano de estados (muestra = [1,500])','FontSize',16);
xlabel('ex','FontSize',16);
ylabel('dex/dt','FontSize',16);
set(p25(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(1,1:laz1)) max(sigk_log(1,1:laz1)) -max(sigk_log(4,1:laz1))
max(sigk_log(4,1:laz1))])

subplot(3,2,2);
p26 = plot(sigk_log(1,laz1:nt),sigk_log(4,laz1:nt),'-b');
title('Plano de estados (muestra = [500,5226])','FontSize',16);
xlabel('ex','FontSize',16);
ylabel('dex/dt','FontSize',16);
set(p26(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(1,laz1:nt)) max(sigk_log(1,laz1:nt)) -max(sigk_log(4,laz1:nt))
max(sigk_log(4,laz1:nt))])

subplot(3,2,3);
laz1 = 2000;
p27 = plot(sigk_log(2,1:laz1),sigk_log(5,1:laz1),'-g');
title('Plano de estados (muestra = [1,2000])','FontSize',16);
xlabel('ey','FontSize',16);
ylabel('dey/dt','FontSize',16);
set(p27(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(2,1:laz1)) max(sigk_log(2,1:laz1)) -max(sigk_log(5,1:laz1))
max(sigk_log(5,1:laz1))])

subplot(3,2,4);
p28 = plot(sigk_log(2,laz1:nt),sigk_log(5,laz1:nt),'-g');
title('Plano de estados (muestra = [2000,5226])','FontSize',16);
xlabel('ey','FontSize',16);
ylabel('dey/dt','FontSize',16);
set(p28(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(2,laz1:nt)) max(sigk_log(2,laz1:nt)) -max(sigk_log(5,laz1:nt))
max(sigk_log(5,laz1:nt))])

subplot(3,2,5);
laz1 = 100;
p29 = plot(sigk_log(3,1:laz1),sigk_log(6,1:laz1),'-r');
title('Plano de estados (muestra = [1,100])','FontSize',16);
xlabel('ez','FontSize',16);
ylabel('dez/dt','FontSize',16);
set(p29(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(3,1:laz1)) max(sigk_log(3,1:laz1)) -max(sigk_log(6,1:laz1))
max(sigk_log(6,1:laz1))])

subplot(3,2,6);
p30 = plot(sigk_log(3,laz1:nt),sigk_log(6,laz1:nt),'-r');
title('Plano de estados (muestra = [100,5226])','FontSize',16);
xlabel('ez','FontSize',16);
ylabel('dez/dt','FontSize',16);
set(p30(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(3,laz1:nt)) max(sigk_log(3,laz1:nt)) -max(sigk_log(6,laz1:nt))
max(sigk_log(6,laz1:nt))])
%% Gráficas de parámetros adaptados
figure(29)
sgtitle('Adaptación de parámetros: Mu (Azul), Lipschitz(Verde)','FontSize',18)
subplot(3,1,1);
```

```
p31 = plot(bin_log(1,:),MU_log(1,:),'-b',bin_log(1,:),L_log(1,:),'-g');
title('Coordenada x','Fontsize',16);
xlabel('muestra','FontSize',16)
ylabel('[un]','FontSize',16)
set(p31(1),'LineWidth', 1.5);
set(p31(2),'LineWidth', 1.5);
grid on
axis([1 nt min([L_log(1,:) MU_log(1,:)]) max([L_log(1,:) MU_log(1,:)])])

subplot(3,1,2);
p32 = plot(bin_log(1,:),MU_log(2,:),'-b',bin_log(1,:),L_log(2,:),'-g');
title('Coordenada y','Fontsize',16);
xlabel('muestra','FontSize',16)
ylabel('[un]','FontSize',16)
set(p32(1),'LineWidth', 1.5);
set(p32(2),'LineWidth', 1.8);
grid on
axis([1 nt min([L_log(2,:) MU_log(2,:)]) max([L_log(2,:) MU_log(2,:)])])

subplot(3,1,3);
p33 = plot(bin_log(1,:),MU_log(3,:),'-b',bin_log(1,:),L_log(3,:),'-g');
title('Coordenada z','Fontsize',14);
xlabel('muestra','FontSize',16)
ylabel('[un]','FontSize',16)
set(p33(1),'LineWidth', 1.5);
set(p33(2),'LineWidth', 1.8);
grid on
axis([1 nt min([L_log(3,:) MU_log(3,:)]) max([L_log(3,:) MU_log(3,:)])])
save trayectorias.mat
```

Apéndice 2.12: Diferenciador robusto exacto de filtrado estándar (REDF)

```
%=========================================================================
%
% Título            : FRED.m
% Propósito         : Función del diferenciador robusto exacto filtrado
%                     estándar
%
%
% Descripción       : Este programa permite filtrar el ruido de la variable
%                     de estado de posición medida de un blanco aéreo
%                     de alta maniobrabilidad en coordenadas cartesianas
%                     obtenidas por un radar de seguimiento automático.
%
% Fecha de creación : 10/03/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
function [xhk,uhk,duhk,sigk] = FRED(yk,T)

persistent z0x z1x z2x z3x z4x w1x w2x w3x z0y z1y z2y z3y z4y w1y w2y w3y...
           z0z z1z z2z z3z z4z w1z w2z w3z

if isempty (z0x)
   z0x = yk(1,1);
   z1x = 0;
   z2x = 0;
   z3x = 0;
   z4x = 0;
   w1x = 0;
   w2x = 0;
   w3x = 0;
   z0y = 0.98*yk(3,1);
   z1y = 0;
   z2y = 0;
   z3y = 0;
   z4y = 0;
   w1y = 0;
   w2y = 0;
   w3y = 0;
   z0z = 0;
   z1z = 0;
```

```
    z2z = 0;
    z3z = 0;
    z4z = 0;
    w1z = 0;
    w2z = 0;
    w3z = 0;
end



%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar filtrado: coordenada x
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0x           posición
% z1x           velocidad
% z2x           aceleración
% z3x           sobre aceleración
% z4x           snap (derivada de la sobre aceleración)
% Lx            Constante de Lipschitz (constante)
% yk(1,1)       posición medida en la coordenada "x"
% T             Tiempo de muestreo
% nx            Orden de diferenciación
% nfx           Orden de filtrado
% k0x,k1x,...   ganancias del diferenciador
% sigx          variable deslizante
% sigdx         derivada de la variable deslizante
% Párametros
%
k0x   = 1.1; k1x = 14.13;k2x = 88.78;k3x = 295.74;k4x = 455.40; k5x = 281.37; k6x =
84.12; k7x =12;
nx    = 4; nfx  = 3;
numx  = nx + nfx; denx = nx + nfx + 1;
Lx    = 40;
%-------------------------------------------------------------------------
% Variable deslizante coordenada z
%-------------------------------------------------------------------------
    sigx    = -(yk(1,1) - z0x);
    sigdx   = -(yk(2,1) - z1x);
%-------------------------------------------------------------------------
% Diferenciador filtrado y uniforme coordenada x
%-------------------------------------------------------------------------
    w1x_dot = -k7x*(Lx^((numx-3)/(denx)))*((abs(w1x)^((numx-0)/(denx)))*sign(w1x)) +
w2x;
    w2x_dot = -k6x*(Lx^((numx-2)/(denx)))*((abs(w1x)^((numx-1)/(denx)))*sign(w1x)) +
w3x;
    w3x_dot = -k5x*(Lx^((numx-1)/(denx)))*((abs(w1x)^((numx-2)/(denx)))*sign(w1x)) +
sigx;
    z0x_dot = -k4x*(Lx^((numx-0)/(denx)))*((abs(w1x)^((numx-3)/(denx)))*sign(w1x)) +
z1x;
    z1x_dot = -k3x*(Lx^((numx+1)/(denx)))*((abs(w1x)^((numx-4)/(denx)))*sign(w1x)) +
z2x;
    z2x_dot = -k2x*(Lx^((numx+2)/(denx)))*((abs(w1x)^((numx-5)/(denx)))*sign(w1x)) +
z3x;
    z3x_dot = -k1x*(Lx^((numx+3)/(denx)))*((abs(w1x)^((numx-6)/(denx)))*sign(w1x)) +
z4x;
    z4x_dot = -k0x*(Lx^((numx+4)/(denx)))*((abs(w1x)^((numx-7)/(denx)))*sign(w1x));

    % Integración de términos
    w1x_1   = w1x + w1x_dot*T;
    w2x_1   = w2x + w2x_dot*T;
    w3x_1   = w3x + w3x_dot*T;
    % Integración del diferenciador
    z0x_1   = z0x + z0x_dot*T;
    z1x_1   = z1x + z1x_dot*T;
    z2x_1   = z2x + z2x_dot*T;
    z3x_1   = z3x + z3x_dot*T;
    z4x_1   = z4x + z4x_dot*T;
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar filtrado: coordenada y
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
```

```
% z0y           posición
% z1y           velocidad
% z2y           aceleración
% z3y           sobre aceleración
% z4y           snap (derivada de la sobre aceleración)
% Ly            Constante de Lipschitz (constante)
% yk(1,1)       posición medida en la coordenada "y"
% T             Tiempo de muestreo
% ny            Orden de diferenciación
% nfy           Orden de filtrado
% k0y,k1y,...   ganancias del diferenciador
% sigy          variable deslizante
% sigdy         derivada de la variable deslizante
% Párametros
%
k0y   = 1.1; k1y = 14.13;k2y = 88.78;k3y = 295.74;k4y = 455.40; k5y = 281.37; k6y =
84.12; k7y =12;
ny    = 4; nfy  = 3;
numy  = ny + nfy; deny = ny + nfy + 1;
Ly    = 40;
%-------------------------------------------------------------------------
% Variable deslizante coordenada y
%-------------------------------------------------------------------------
    sigy    = -(yk(3,1) - z0y);
    sigdy   = -(yk(4,1) - z1y);
%-------------------------------------------------------------------------
% Diferenciador filtrado y uniforme coordenada y
%-------------------------------------------------------------------------
    w1y_dot = -k7y*(Ly^((numy-3)/(deny)))*((abs(w1y)^((numy-0)/(deny)))*sign(w1y)) +
w2y;
    w2y_dot = -k6y*(Ly^((numy-2)/(deny)))*((abs(w1y)^((numy-1)/(deny)))*sign(w1y)) +
w3y;
    w3y_dot = -k5y*(Ly^((numy-1)/(deny)))*((abs(w1y)^((numy-2)/(deny)))*sign(w1y)) +
sigy;
    z0y_dot = -k4y*(Ly^((numy-0)/(deny)))*((abs(w1y)^((numy-3)/(deny)))*sign(w1y)) +
z1y;
    z1y_dot = -k3y*(Ly^((numy+1)/(deny)))*((abs(w1y)^((numy-4)/(deny)))*sign(w1y)) +
z2y;
    z2y_dot = -k2y*(Ly^((numy+2)/(deny)))*((abs(w1y)^((numy-5)/(deny)))*sign(w1y)) +
z3y;
    z3y_dot = -k1y*(Ly^((numy+3)/(deny)))*((abs(w1y)^((numy-6)/(deny)))*sign(w1y)) +
z4y;
    z4y_dot = -k0y*(Ly^((numy+4)/(deny)))*((abs(w1y)^((numy-7)/(deny)))*sign(w1y));

    % Integración de términos
    w1y_1   = w1y + w1y_dot*T;
    w2y_1   = w2y + w2y_dot*T;
    w3y_1   = w3y + w3y_dot*T;
    % Integración del diferenciador
    z0y_1   = z0y + z0y_dot*T;
    z1y_1   = z1y + z1y_dot*T;
    z2y_1   = z2y + z2y_dot*T;
    z3y_1   = z3y + z3y_dot*T;
    z4y_1   = z4y + z4y_dot*T;
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar filtrado: coordenada z
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0z           posición
% z1z           velocidad
% z2z           aceleración
% z3z           sobre aceleración
% z4z           snap (derivada de la sobre aceleración)
% Lz            Constante de Lipschitz (constante)
% yk(4,1)       posición medida en la coordenada "x"
% T             Tiempo de muestreo
% nz            Orden de diferenciación
% nfz           Orden de filtrado
% k0z,k1z,...   ganancias del diferenciador
% sigz          variable deslizante
% sigdz         derivada de la variable deslizante
% Párametros
%
k0z   = 1.1; k1z = 14.13;k2z = 88.78;k3z = 295.74;k4z = 455.40; k5z = 281.37; k6z =
84.12; k7z =12;
```

```
nz    = 4; nfz  = 3;
numz  = nz + nfz; denz = nz + nfz + 1;
Lz    = 1;
%-------------------------------------------------------------------------
% Variable deslizante coordenada z
%-------------------------------------------------------------------------
    sigz   = -(yk(5,1) - z0z);
    sigdz  = -(yk(6,1) - z1z);



%-------------------------------------------------------------------------
% Diferenciador filtrado y uniforme coordenada z
%-------------------------------------------------------------------------
    w1z_dot = -k7z*(Lz^((numz-3)/(denz)))*((abs(w1z)^((numz-0)/(denz)))*sign(w1z)) +
w2z;
    w2z_dot = -k6z*(Lz^((numz-2)/(denz)))*((abs(w1z)^((numz-1)/(denz)))*sign(w1z)) +
w3z;
    w3z_dot = -k5z*(Lz^((numz-1)/(denz)))*((abs(w1z)^((numz-2)/(denz)))*sign(w1z)) +
sigz;
    z0z_dot = -k4z*(Lz^((numz-0)/(denz)))*((abs(w1z)^((numz-3)/(denz)))*sign(w1z)) +
z1z;
    z1z_dot = -k3z*(Lz^((numz+1)/(denz)))*((abs(w1z)^((numz-4)/(denz)))*sign(w1z)) +
z2z;
    z2z_dot = -k2z*(Lz^((numz+2)/(denz)))*((abs(w1z)^((numz-5)/(denz)))*sign(w1z)) +
z3z;
    z3z_dot = -k1z*(Lz^((numz+3)/(denz)))*((abs(w1z)^((numz-6)/(denz)))*sign(w1z)) +
z4z;
    z4z_dot = -k0z*(Lz^((numz+4)/(denz)))*((abs(w1z)^((numz-7)/(denz)))*sign(w1z));

    % Integración de términos
    w1z_1   = w1z + w1z_dot*T;
    w2z_1   = w2z + w2z_dot*T;
    w3z_1   = w3z + w3z_dot*T;
    % Integración del diferenciador
    z0z_1   = z0z + z0z_dot*T;
    z1z_1   = z1z + z1z_dot*T;
    z2z_1   = z2z + z2z_dot*T;
    z3z_1   = z3z + z3z_dot*T;
    z4z_1   = z4z + z4z_dot*T;
%-------------------------------------------------------------------------
% Guardar variables para siguiente iteración
%-------------------------------------------------------------------------
z0x = z0x_1;
z1x = z1x_1;
z2x = z2x_1;
z3x = z3x_1;
z4x = z4x_1;
w1x = w1x_1;
w2x = w2x_1;
w3x = w3x_1;
z0y = z0y_1;
z1y = z1y_1;
z2y = z2y_1;
z3y = z3y_1;
z4y = z4y_1;
w1y = w1y_1;
w2y = w2y_1;
w3y = w3y_1;
z0z = z0z_1;
z1z = z1z_1;
z2z = z2z_1;
z3z = z3z_1;
z4z = z4z_1;
w1z = w1z_1;
w2z = w2z_1;
w3z = w3z_1;
xhk = [z0x;z1x;z2x;z0y;z1y;z2y;z0z;z1z;z2z];
uhk = [z3x;z3y;z3z];
duhk = [z4x;z4y;z4z];
sigk        = [sigx sigy sigz sigdx sigdy sigdz]';
end
```

Apéndice 2.13: Simulación de la trayectoria del misil (REDF)

```
%=========================================================================
%
```

```
% Título            : sim_FRED.m
% Propósito         : Simulación de diferenciador robusto exacto de
%                     filtrado estándar
%
% Descripción       : Este programa permite la estimación de variables de
%                     estado filtradas de posición, velocidad, aceleración,
%                     sobre aceleración y derivada de la sobre aceleración
%                     de un blanco aéreo de alta maniobrabilidad a partir
%                     de las mediciones ruidosas de posición en coordenadas
%                     cartesianas obtenidas por un radar de seguimiento
%                     automático. Al efectuar simulaciones se observa que
%                     el presente diferenciador presenta buena filtración
%                     del ruido pero carece de exactitud y uniformismo
%                     debido a que no efectúa la adaptación de la constante
%                     de Lipschitz y carece de términos super-twisting de
%                     orden alto.
%
% Fecha de creación : 01/03/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
clear all;
clc;
close all;
%-------------------------------------------------------------------------
% Inicialización de variables
%-------------------------------------------------------------------------
RMSEcold = 0;
RMSEpold = 0;
%-------------------------------------------------------------------------
% Parámetros iniciales de radar (sensor)
%-------------------------------------------------------------------------
T     = 1/50;                       % Período de muestreo (seg)
%-------------------------------------------------------------------------
% Definición de constantes físicas
%-------------------------------------------------------------------------
g     = 9.81;                       % gravedad (m/s2)
M     = 343;                        % velocidad del sonido (m/s)
load_factor = 100*g;                % máxima sobre aceleración (g/s)
%-------------------------------------------------------------------------
% Definición de vectores de tiempo
%-------------------------------------------------------------------------
ti    = 0;                          % Tiempo inicial de simulación
tff   = 104.5;                      % Tiempo final de simulación
tt    = ti:T:tff;                   % Vector de tiempo
tt    = tt';                        % Transpuesta del vector de tiempo
nt    = length(tt);                 % Número de muestras
%-------------------------------------------------------------------------
% Definición de posición y velocidad inicial del blanco en
% coordenadas esféricas
%-------------------------------------------------------------------------
Ad_init  = 35000;                       % Distancia al blanco (m)
Ed_init  = 0.01;                        % Elevación al blanco (°)
Bdn_init = 30;                          % Marcación al blanco (°)
Vm_init  = 0.9*M;                       % Velocidad del blanco (1 Mach = 343 m/s)
Rv_init  = 210;                         % Rumbo inicial del blanco (°)
%-------------------------------------------------------------------------
% Conversión de coordenadas esféricas a cartesianas (North-Sky-East) de la
% posición y velocidad inicial del blanco
%-------------------------------------------------------------------------
[Adx_init, Ady_init, Adz_init] = p2c(Ad_init,Ed_init,Bdn_init);
[Vmx_init, Vmy_init, Vmz_init] = p2c(Vm_init,0,Rv_init);
%-------------------------------------------------------------------------
% Inicialización de vector de estados
%-------------------------------------------------------------------------
xk = [Adx_init Vmx_init 0*g Ady_init Vmy_init 0*g Adz_init Vmz_init 0*g]';
%-------------------------------------------------------------------------
% Definición de matrices del modelo de espacio de estados del sistema
%-------------------------------------------------------------------------
% Matriz de medición (Pulse radar)
Ck     = [1 0 0 0 0 0 0 0 0;        % pos x
          0 0 0 0 0 0 0 0 0;        % vel x
          0 0 0 1 0 0 0 0 0;        % pos y
```

```
          0 0 0 0 0 0 0 0 0;        % vel y
          0 0 0 0 0 0 1 0 0;        % pos z
          0 0 0 0 0 0 0 0 0];       % vel z

% Cálculo de matriz de transición Fk
Fk = [1 T (T^2)/2 0 0       0 0 0       0;
      0 1       T 0 0       0 0 0       0;
      0 0       1 0 0       0 0 0       0;
      0 0       0 1 T (T^2)/2 0 0       0;
      0 0       0 0 1       T 0 0       0;
      0 0       0 0 0       1 0 0       0;
      0 0       0 0 0       0 1 T (T^2)/2;
      0 0       0 0 0       0 0 1       T;
      0 0       0 0 0       0 0 0       1];

% Cálculo de la matriz de entrada Gk
Gk = [(T^3)/6       0        0;
      (T^2)/2       0        0;
       T            0        0;
       0      (T^3)/6        0;
       0      (T^2)/2        0;
       0            T        0;
       0            0  (T^3)/6;
       0            0  (T^2)/2;
       0            0        T];

% Matriz de distribución de perturbaciones (aceleraciones)
Dk = [75e4*(T^3)/6       0                0;
      5e3*(T^2)/2        0                0;
       1e2*T             0                0;
       0      75e4*(T^3)/6                0;
       0       5e3*(T^2)/2                0;
       0             1e2*T                0;
       0                0     75e4*(T^3)/6;
       0                0      5e3*(T^2)/2;
       0                0           1e2*T];
%-------------------------------------------------------------------------
% Generación del vector de sobre aceleraciones de entrada uk
%-------------------------------------------------------------------------
uk = gen_jerk_sea_skimming3(nt,load_factor,g,T);
%-------------------------------------------------------------------------
% Generación del vector de perturbaciones ?k
%-------------------------------------------------------------------------
pk = perturb_1(g,ti,T,tff);
%-------------------------------------------------------------------------
% Cargar ruido glint
%-------------------------------------------------------------------------
load glint_puntos.mat
%-------------------------------------------------------------------------
% Simulación del diferenciador
%-------------------------------------------------------------------------
disp('Diferenciador Robusto Exacto de filtrado estándar');
prompt = 'Para simular con ruido presione "1". Para simular sin ruido presione "0": ';
gl     = input(prompt);
for k = 1:nt

    % Modelo dinámico o de transición de estados
    xk       = Fk*xk + Gk*uk(:,k) + Dk*pk(:,k);
    % Simulación de adquisición de datos por el radar
    % Datos de posición adquiridos en coordenadas esféricas
    [Ad,Bdn,Ed] = c2p(xk);
    % Modelo de medición polar a cartesiano
    yk = yk_noise(Ad,Bdn,Ed,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,gl,k);
    % Diferenciador Robusto Exacto
    tic
    [xhk,uhk,duhk,sigk] = FRED(yk,T);
    TCPU = toc;
    % Conversión de coordenada estimadas cartesianas a coordenadas polares
    [Adhat,Bdnhat,Edhat] = c2p(xhk);
    % Cálculo de errores en coordenadas cartesianas
    errorc       = xhk - xk;
    % Cálculo de errores en coordenadas esféricas
    errorAd      = Adhat - Ad;                                    %mts
    errorBdn     = Bdnhat*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEd      = Edhat*(pi*1000/180) - Ed*(pi*1000/180);
    errorp       = [errorAd; errorBdn; errorEd];
     if ( k >= 4650)
```

```
        RMSEc          = RMSEcold + ((errorc).^2);
        RMSEp          = RMSEpold + ((errorp).^2);
    else
        RMSEc          = 0;
        RMSEp          = 0;
     end
     % Guardar variables
    bin_log(:,k)       = k;
    xk_log(:,k)        = xk;
    yk_log(:,k)        = yk;
    xhk_log(:,k)       = xhk;
    uhk_log(:,k)       = uhk;
    duhk_log(:,k)      = duhk;
    sigk_log(:,k)      = sigk;
    errorc_log(:,k)    = errorc;
    errorp_log(:,k)    = errorp;
    polar_log(:,k)     = [Ad;Bdn;Ed];
    polarhat_log(:,k)  = [Adhat;Bdnhat;Edhat];
    RMSEcold           = RMSEc;
    RMSEpold           = RMSEp;
    TCPU_log(:,k)      = TCPU;
end
RMSEctot = sqrt(RMSEc)/(nt-4650);
RMSEptot = sqrt(RMSEp)/(nt-4650);
%-------------------------------------------------------------------------
% Conversión de vector de estados reales a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
% 0.10197 - de mts/seg2 a g's
conv_xk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197]';
xk_log = xk_log.*conv_xk;
%-------------------------------------------------------------------------
% Conversión de vector de estados medidos a Mn, Mach
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
conv_yk   = [ 0.00054 0.00292 0.00054 0.00292 0.00054 0.00292 ]';
yk_log    = yk_log.*conv_yk;
%-------------------------------------------------------------------------
% Conversión de vector de estados estimados a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - mn
% 0.00292 - Mach
% 0.10197 - g's
conv_xhk   = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292
0.10197]';
xhk_log = xhk_log.*conv_xhk;
%-------------------------------------------------------------------------
% Conversión de vector de entrada real y estimado a g/s
%-------------------------------------------------------------------------
uk_log    = uk./g;
uhk_log   = uhk_log./g;
duhk_log   = duhk_log./g;
%-------------------------------------------------------------------------
% Conversión de perturbaciones a g/s
%-------------------------------------------------------------------------
pk_log    = pk./g;
%-------------------------------------------------------------------------
% Plots
%-------------------------------------------------------------------------
%% 3D trayectoria completa

figure(1);p1 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-g',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-r',...
                       xhk_log(1,:),       xhk_log(4,:),       xhk_log(7,:),'-k',...
                                0,                  0,                  0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p1(1),'LineWidth',1.5);
         set(p1(2),'LineWidth',2.0);
         set(p1(3),'LineWidth',2.0);
         set(p1(4),'LineWidth',2.0);
         set(p1(5),'LineWidth',3.0);
         set(p1(6),'LineWidth',3.0);
```

```
        set(p1(7),'LineWidth',3.0);
axis([-5 20 -5 10 0 0.04]);          % Toda la trayectoria
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Trayectoria sea skimming del misil','FontSize',16)
legend({'mediciones','real','RED filtrado estandar','Posición del buque
observador','Posición inicial del misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
%% 3D trayectoria completa

figure(2);p2 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-g',...
                       xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-r',...
                       xhk_log(1,:),        xhk_log(4,:),        xhk_log(7,:),'-k',...
                                0,                   0,                   0, 'o',...
                       xk_log(1,1),        xk_log(4,1),        xk_log(7,1), 'o',...
                       xk_log(1,nt),       xk_log(4,nt),       xk_log(7,nt),'o',...
                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),   0.9*xk_log(7,nt),'o');
grid on
set(p2(1),'LineWidth',1.5);
set(p2(2),'LineWidth',2.0);
set(p2(3),'LineWidth',2.0);
set(p2(4),'LineWidth',2.0);
set(p2(5),'LineWidth',3.0);
set(p2(6),'LineWidth',3.0);
set(p2(7),'LineWidth',3.0);
axis([4 8 -2 5 0 0.016]);
axis ij
xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
zlabel('Altitud (eje z) en [mn]','FontSize',16)
title('Maniobra terminal del misil','FontSize',16)
legend({'mediciones','real','RED','Posición del buque observador','Posición inicial del
misil',...
       'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14);

%% Coordenada x
% Posición x: Trayectoria completa
figure(3);
subplot(1,2,1);
p3 = plot(bin_log(1,:),xk_log(1,:),'-g',bin_log(1,:),yk_log(1,:),'-
r',bin_log(1,:),xhk_log(1,:),'-k');
title('Trayectoria completa del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'real','medición','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p3(1),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(1,:) yk_log(1,:)]) max([xk_log(1,:) yk_log(1,:)])])

% Posición x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p4 = plot(bin_log(1,lpx1:nt),xk_log(1,lpx1:nt),'-
g',bin_log(1,lpx1:nt),yk_log(1,lpx1:nt),'-r',bin_log(1,lpx1:nt),xhk_log(1,lpx1:nt),'-
k');
title('Maniobra terminal del misil: posición x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'real','medición','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p4(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)]) max([xk_log(1,lpx1:nt)
yk_log(1,lpx1:nt)])])
%%
% Velocidad x: Trayectoria completa
figure(4);
subplot(1,2,1);
p5 = plot(bin_log(1,:),xk_log(2,:),'-g',bin_log(1,:),xhk_log(2,:),'-k');
title('Trayectoria completa del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p5(1),'LineWidth', 1.5);
```

```
grid on
axis([1 nt min([xk_log(2,:) xhk_log(2,:)]) max([xk_log(2,:) xhk_log(2,:)])])
axis([1 nt -2 2.5])
% Velocidad x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p6 = plot(bin_log(1,lpx1:nt),xk_log(2,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(2,lpx1:nt),'-k');
title('Maniobra terminal del misil: velocidad x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p6(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])
max([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt) xhk_log(2,lpx1:nt)])])
%%
% Aceleración x: Trayectoria completa
figure(5);
subplot(1,2,1);
p7 = plot(bin_log(1,:),xk_log(3,:),'-g',bin_log(1,:),xhk_log(3,:),'-k');
title('Trayectoria completa del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p7(1),'LineWidth', 1.5);
grid on
%axis([1 nt min(xk_log(3,:)) max(xk_log(3,:))])
axis([1 nt -100 70])
% Aceleración x: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p8 = plot(bin_log(1,lpx1:nt),xk_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(3,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración x','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p8(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(3,lpx1:nt) xhk_log(3,lpx1:nt)]) max([xk_log(3,lpx1:nt)
xhk_log(3,lpx1:nt)])])
%axis([lpx1 nt -20 20])
%% Coordenada y
% Posición y: Trayectoria completa
figure(6);
subplot(1,2,1);
p9 = plot(bin_log(1,:),xk_log(4,:),'-g',bin_log(1,:),yk_log(3,:),'-
r',bin_log(1,:),xhk_log(4,:),'-k');
title('Trayectoria completa del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'real','medición','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p9(1),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(4,:) yk_log(3,:)]) max([xk_log(4,:) yk_log(3,:)])])

% Posición y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p10 = plot(bin_log(1,lpx1:nt),xk_log(4,lpx1:nt),'-
g',bin_log(1,lpx1:nt),yk_log(3,lpx1:nt),'-r',bin_log(1,lpx1:nt),xhk_log(4,lpx1:nt),'-
K');
title('Maniobra terminal del misil: posición y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mn]','FontSize',22)
legend({'real','medición','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p10(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])
max([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt) xhk_log(4,lpx1:nt)])])

% Velocidad y: Trayectoria completa
figure(7);
subplot(1,2,1);
p11 = plot(bin_log(1,:),xk_log(5,:),'-g',bin_log(1,:),yk_log(4,:),'-
r',bin_log(1,:),xhk_log(5,:),'-k');
```

```
title('Trayectoria completa del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','medición','RED'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p11(1),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(5,:) yk_log(4,:)]) max([xk_log(5,:) yk_log(4,:)])])
%axis([1 nt -10 10])
% Velocidad y: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p12 = plot(bin_log(1,lpx1:nt),xk_log(5,lpx1:nt),'-
g',bin_log(1,lpx1:nt),yk_log(4,lpx1:nt),'-r',bin_log(1,lpx1:nt),xhk_log(5,lpx1:nt),'-
k');
title('Maniobra terminal del misil: velocidad y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','medición','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p12(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(5,lpx1:nt) yk_log(4,lpx1:nt)]) max([xk_log(5,lpx1:nt)
yk_log(4,lpx1:nt)])])

% Aceleración y: Trayectoria completa
figure(8);
subplot(1,2,1);
p13 = plot(bin_log(1,lpx1:nt),xk_log(6,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(6,lpx1:nt),'-k');
title('Trayectoria completa del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p13(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(6,lpx1:nt) xhk_log(6,lpx1:nt)]) max([xk_log(6,lpx1:nt)
xhk_log(6,lpx1:nt)])])

% Aceleración y: Maniobra terminal
subplot(1,2,2);
p14 = plot(bin_log(1,:),xk_log(6,:),'-g',bin_log(1,:),xhk_log(6,:),'-k');
title('Maniobra terminal del misil: aceleración y','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p14(1),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(6,:) xhk_log(6,:)]) max([xk_log(6,:) xhk_log(6,:)])])

%% Coordenada z
% Posición z: Trayectoria completa
figure(9);
subplot(1,2,1);
p15 = plot(bin_log(1,:),yk_log(5,:)./0.00054,'-r',bin_log(1,:),xk_log(7,:)./0.00054,'-
g',bin_log(1,:),xhk_log(7,:)./0.00054,'-m');
title('Trayectoria completa del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mts]','FontSize',22)
legend({'medición','real','RED filtrado
estándar'},'Location','north','NumColumns',3,'FontSize',18)
set(p15(1),'LineWidth', 2.0);
set(p15(2),'LineWidth', 1.8);
set(p15(3),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(7,:)./0.00054 yk_log(5,:)./0.00054]) max([xk_log(7,:)./0.00054
yk_log(5,:)./0.00054])])

% Posición z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p16 = plot(bin_log(1,lpx1:nt),yk_log(5,lpx1:nt)./0.00054,'-
r',bin_log(1,lpx1:nt),xk_log(7,lpx1:nt)./0.00054,'-
g',bin_log(1,lpx1:nt),xhk_log(7,lpx1:nt)./0.00054,'-k');
title('Maniobra terminal del misil: posición z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('posición en [mts]','FontSize',22)
set(p16(1),'LineWidth', 2.0);
set(p16(2),'LineWidth', 1.8);
```

```
set(p16(3),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054]) max([xk_log(7,lpx1:nt)./0.00054 yk_log(5,lpx1:nt)./0.00054
xhk_log(7,lpx1:nt)./0.00054])])
%%
% Velocidad z: Trayectoria completa
figure(10);
subplot(1,2,1);
p17 = plot(bin_log(1,:),xk_log(8,:),'-g',bin_log(1,:),yk_log(6,:),'-
r',bin_log(1,:),xhk_log(8,:),'-k');
title('Trayectoria completa del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','medición','RED'},'Location','northeast','NumColumns',3,'FontSize',18)
set(p17(1),'LineWidth', 1.5);
grid on
axis([1 nt min([xk_log(8,:) yk_log(6,:)]) max([xk_log(8,:) yk_log(6,:)])])
%axis([1 nt -4 4])
% Velocidad z: Maniobra terminal
lpx1 = 4500;
subplot(1,2,2);
p18 = plot(bin_log(1,lpx1:nt),xk_log(8,lpx1:nt),'-
g',bin_log(1,lpx1:nt),yk_log(6,lpx1:nt),'-r',bin_log(1,lpx1:nt),xhk_log(8,lpx1:nt),'-
k');
title('Maniobra terminal del misil: velocidad z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('velocidad en [mach]','FontSize',22)
legend({'real','medición','RED'},'Location','north','NumColumns',3,'FontSize',18)
set(p18(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(8,lpx1:nt) yk_log(6,lpx1:nt)]) max([xk_log(8,lpx1:nt)
yk_log(6,lpx1:nt)])])

% Aceleración z: Trayectoria completa
figure(11);
subplot(1,2,1);
p19 = plot(bin_log(1,:),xk_log(9,:),'-g',bin_log(1,:),xhk_log(9,:),'-k');
title('Trayectoria completa del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p19(1),'LineWidth', 1.5);
grid on
axis([1 nt min(xk_log(9,:)) max(xk_log(9,:))])
%axis([1 nt min([xk_log(9,:) xhk_log(9,:)]) max([xk_log(9,:) xhk_log(9,:)])])

% Aceleración z: Maniobra terminal
subplot(1,2,2);
p20 = plot(bin_log(1,lpx1:nt),xk_log(9,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhk_log(9,lpx1:nt),'-k');
title('Maniobra terminal del misil: aceleración z','FontSize',22);
xlabel('muestra','FontSize',22)
ylabel('aceleración en [g]','FontSize',22)
legend({'real','RED'},'Location','north','NumColumns',2,'FontSize',18)
set(p20(1),'LineWidth', 1.5);
grid on
axis([lpx1 nt min([xk_log(9,lpx1:nt) xhk_log(9,lpx1:nt)]) max([xk_log(9,lpx1:nt)
xhk_log(9,lpx1:nt)])])

%% Plots señales de error cartesianas: Coordenada x
figure(12);
lpx1 = 4500;
subplot(3,2,1);
p21 = plot(bin_log(1,:),errorc_log(1,:),'-g');
title('Error cartesiano posición x','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m]','FontSize',16)
set(p21,'LineWidth',1.5);
grid on
axis([1 nt min(errorc_log(1,:)) max(errorc_log(1,:))])

subplot(3,2,2);
p22 = plot(bin_log(1,lpx1:nt),errorc_log(1,lpx1:nt),'-g');
title('Error cartesiano posición x','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
```

```
set(p22,'LineWidth',1.5);
grid on
axis([lpx1 nt min(errorc_log(1,lpx1:nt)) max(errorc_log(1,lpx1:nt))])

subplot(3,2,3);
p23 = plot(bin_log(1,:),errorc_log(2,:),'-g');
title('Error cartesiano velocidad x','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s]','FontSize',16)
set(p23,'LineWidth',1.5);
grid on
axis([1 nt min(errorc_log(2,:)) max(errorc_log(2,:))])

subplot(3,2,4);
p24 = plot(bin_log(1,lpx1:nt),errorc_log(2,lpx1:nt),'-g');
title('Error cartesiano velocidad x','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s]','FontSize',16);
set(p24,'LineWidth',1.5);
grid on
axis([lpx1 nt min(errorc_log(2,lpx1:nt)) max(errorc_log(2,lpx1:nt))])

subplot(3,2,5);
p24 = plot(bin_log(1,:),errorc_log(3,:),'-g');
title('Error cartesiano aceleración x','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s2]','FontSize',16)
set(p24,'LineWidth',1.5);
grid on
axis([1 nt min(errorc_log(3,:)) max(errorc_log(3,:))])

subplot(3,2,6);
p25 = plot(bin_log(1,lpx1:nt),errorc_log(3,lpx1:nt),'-g');
title('Error cartesiano aceleración x','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s2]','FontSize',16);
set(p25,'LineWidth',1.5);
grid on
axis([lpx1 nt min(errorc_log(3,lpx1:nt)) max(errorc_log(3,lpx1:nt))])

% Plots señales de error cartesianas: Coordenada y
figure(13);
subplot(3,2,1);
p26 = plot(bin_log(1,:),errorc_log(4,:),'-g');
title('Error cartesiano posición y','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m]','FontSize',16)
set(p26,'LineWidth',1.5);
grid on
axis([1 nt min(errorc_log(4,:)) max(errorc_log(4,:))])

subplot(3,2,2);
p27 = plot(bin_log(1,lpx1:nt),errorc_log(4,lpx1:nt),'-g');
title('Error cartesiano posición y','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p27,'LineWidth',1.5);
grid on
axis([lpx1 nt min(errorc_log(4,lpx1:nt)) max(errorc_log(4,lpx1:nt))])

subplot(3,2,3);
p28 = plot(bin_log(1,:),errorc_log(5,:),'-g');
title('Error cartesiano velocidad y','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s]','FontSize',16)
set(p28,'LineWidth',1.5);
grid on
axis([1 nt min(errorc_log(5,:)) max(errorc_log(5,:))])

subplot(3,2,4);
p29 = plot(bin_log(1,lpx1:nt),errorc_log(5,lpx1:nt),'-g');
title('Error cartesiano velocidad y','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s]','FontSize',16);
set(p29,'LineWidth',1.5);
grid on
axis([lpx1 nt min(errorc_log(5,lpx1:nt)) max(errorc_log(5,lpx1:nt))])
```

```
subplot(3,2,5);
p30 = plot(bin_log(1,:),errorc_log(6,:),'-g');
title('Error cartesiano aceleración y','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s2]','FontSize',16)
set(p30,'LineWidth',1.5);
grid on
axis([1 nt min(errorc_log(6,:)) max(errorc_log(6,:))])

subplot(3,2,6);
p31 = plot(bin_log(1,lpx1:nt),errorc_log(6,lpx1:nt),'-g');
title('Error cartesiano aceleración y','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s2]','FontSize',16);
set(p31,'LineWidth',1.5);
grid on
axis([lpx1 nt min(errorc_log(6,lpx1:nt)) max(errorc_log(6,lpx1:nt))])

% Plots señales de error cartesianas: Coordenada z
figure(14);
lpx1 = 4500;
subplot(3,2,1);
p32 = plot(bin_log(1,:),errorc_log(7,:),'-g');
title('Error cartesiano posición z','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m]','FontSize',16)
set(p32,'LineWidth',1.5);
grid on
axis([1 nt min(errorc_log(7,:)) max(errorc_log(7,:))])

subplot(3,2,2);
p33 = plot(bin_log(1,lpx1:nt),errorc_log(7,lpx1:nt),'-g');
title('Error cartesiano posición z','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p33,'LineWidth',1.5);
grid on
axis([lpx1 nt min(errorc_log(7,lpx1:nt)) max(errorc_log(7,lpx1:nt))])

subplot(3,2,3);
p34 = plot(bin_log(1,:),errorc_log(8,:),'-g');
title('Error cartesiano velocidad z','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s]','FontSize',16)
set(p34,'LineWidth',1.5);
grid on
axis([1 nt min(errorc_log(8,:)) max(errorc_log(8,:))])

subplot(3,2,4);
p35 = plot(bin_log(1,lpx1:nt),errorc_log(8,lpx1:nt),'-g');
title('Error cartesiano velocidad z','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s]','FontSize',16);
set(p35,'LineWidth',1.5);
grid on
axis([lpx1 nt min(errorc_log(8,lpx1:nt)) max(errorc_log(8,lpx1:nt))])

subplot(3,2,5);
p36 = plot(bin_log(1,:),errorc_log(9,:),'-g');
title('Error cartesiano aceleración z','FontSize',16);
xlabel('muestra','FontSize',16)
ylabel('Error en [m/s2]','FontSize',16)
set(p36,'LineWidth',1.5);
grid on
axis([1 nt min(errorc_log(9,:)) max(errorc_log(9,:))])

subplot(3,2,6);
p37 = plot(bin_log(1,lpx1:nt),errorc_log(9,lpx1:nt),'-g');
title('Error cartesiano aceleración z','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m/s2]','FontSize',16);
set(p37,'LineWidth',1.5);
grid on
axis([lpx1 nt min(errorc_log(9,lpx1:nt)) max(errorc_log(9,lpx1:nt))])
%% Plots señales de error polares: Distancia Ad
figure(15);
```

```
maxAd = 15*ones(1,nt);
minAd = -15*ones(1,nt);

subplot(3,2,1);
laz1 = 4500;
p38 = plot(bin_log(1,:),errorp_log(1,:),'-g',bin_log(1,:),maxAd(1,:),'--
r',bin_log(1,:),minAd(1,:),'--r');
title('Error polar: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
legend({'Ad','Límite +15/-15 mts'},'NumColumns',2,'Location','North','FontSize',12);
set(p38(1),'LineWidth',1.5);
set(p38(2),'LineWidth',1.0);
set(p38(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(1,:) minAd(1,:)]) max([errorp_log(1,:) maxAd(1,:)])])

subplot(3,2,2);
p39 = plot(bin_log(1,laz1:nt),errorp_log(1,laz1:nt),'-
g',bin_log(1,laz1:nt),maxAd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minAd(1,laz1:nt),'--r');
title('Error polar: Alcance','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [m]','FontSize',16);
set(p39(1),'LineWidth',1.5);
set(p39(2),'LineWidth',1.0);
set(p39(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(1,laz1:nt) minAd(1,laz1:nt)]) max([errorp_log(1,laz1:nt)
maxAd(1,laz1:nt)])])
% Error polar en acimut
maxBdn = 2*ones(1,nt);
minBdn = -2*ones(1,nt);

subplot(3,2,3);
laz1 = 4500;
p40 = plot(bin_log(1,:),errorp_log(2,:),'-g',bin_log(1,:),maxBdn(1,:),'--
r',bin_log(1,:),minBdn(1,:),'--r');
title('Error polar: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Bdn','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p40(1),'LineWidth',1.5);
set(p40(2),'LineWidth',1.0);
set(p40(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(2,:) minBdn(1,:)]) max([errorp_log(2,:) maxBdn(1,:)])])

subplot(3,2,4);
p41 = plot(bin_log(1,laz1:nt),errorp_log(2,laz1:nt),'-
g',bin_log(1,laz1:nt),maxBdn(1,laz1:nt),'--r',bin_log(1,laz1:nt),minBdn(1,laz1:nt),'--
r');
title('Error polar: Acimut','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p41(1),'LineWidth',1.5);
set(p41(2),'LineWidth',1.0);
set(p41(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(2,laz1:nt) minBdn(1,laz1:nt)]) max([errorp_log(2,laz1:nt)
maxBdn(1,laz1:nt)])])

maxEd = 2*ones(1,nt);
minEd = -2*ones(1,nt);

subplot(3,2,5);
laz1 = 4500;
p42 = plot(bin_log(1,:),errorp_log(3,:),'-g',bin_log(1,:),maxEd(1,:),'--
r',bin_log(1,:),minEd(1,:),'--r');
title('Error polar: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
legend({'Ed','Límite +2/-2 mrad'},'NumColumns',2,'Location','North','FontSize',12);
set(p42(1),'LineWidth',1.5);
set(p42(2),'LineWidth',1.0);
set(p42(3),'LineWidth',1.0);
grid on
axis([1 nt min([errorp_log(3,:) minEd(1,:)]) max([errorp_log(3,:) maxEd(1,:)])])
```

```
subplot(3,2,6);
p43 = plot(bin_log(1,laz1:nt),errorp_log(3,laz1:nt),'-
g',bin_log(1,laz1:nt),maxEd(1,laz1:nt),'--r',bin_log(1,laz1:nt),minEd(1,laz1:nt),'--r');
title('Error polar: Elevación','FontSize',16);
xlabel('muestra','FontSize',16);
ylabel('Error en [mrad]','FontSize',16);
set(p43(1),'LineWidth',1.5);
set(p43(2),'LineWidth',1.0);
set(p43(3),'LineWidth',1.0);
grid on
axis([laz1 nt  min([errorp_log(3,laz1:nt) minEd(1,laz1:nt)]) max([errorp_log(3,laz1:nt)
maxEd(1,laz1:nt)])])
%% Estimación de vector de entrada de control
figure(16);
p44 = plot(bin_log(1,:),uk_log(1,:),'-g',bin_log(1,:),uhk_log(1,:),'-
k',bin_log(1,:),pk_log(1,:),'-m');
title('Trayectoria del misil: sobre aceleración x','FontSize',16);
legend({'real uk','RED uhatk','real
pk'},'Location','North','NumColumns',3,'FontSize',14);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p44(1),'LineWidth',1.5);
set(p44(2),'LineWidth',1.5);
set(p44(3),'LineWidth',1.5);
grid on
axis([1 nt min([uk_log(1,:) uhk_log(1,:)]) max([uk_log(1,:) uhk_log(1,:)])])

figure(17);
p45 = plot(bin_log(1,:),uk_log(2,:),'-g',bin_log(1,:),uhk_log(2,:),'-
k',bin_log(1,:),pk_log(2,:),'-m');
title('Trayectoria del misil: sobre aceleración y','FontSize',16);
legend({'real uk','RED uhatk','real
pk'},'Location','North','NumColumns',3,'FontSize',14);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p45(1),'LineWidth',1.5);
set(p45(2),'LineWidth',1.5);
set(p45(3),'LineWidth',1.5);
grid on
axis([1 nt min([uk_log(2,:) uhk_log(2,:)]) max([uk_log(2,:) uhk_log(2,:)])])

figure(18);
p46 = plot(bin_log(1,:),uk_log(3,:),'-g',bin_log(1,:),uhk_log(3,:),'-
k',bin_log(1,:),pk_log(3,:),'-m');
title('Trayectoria del misil: sobre aceleración z','FontSize',16);
legend({'real uk','RED uhatk','real
pk'},'Location','North','NumColumns',3,'FontSize',14);
xlabel('muestra','FontSize',16);
ylabel('[g/s]','FontSize',16);
set(p46(1),'LineWidth',1.5);
set(p46(2),'LineWidth',1.5);
set(p46(3),'LineWidth',1.5);
grid on
axis([1 nt min([uk_log(3,:) uhk_log(3,:)]) max([uk_log(3,:) uhk_log(3,:)])])
%% Gráfica de plano de estados (colector deslizante en el origen)
figure(19);

subplot(3,2,1);
laz1 = 500;
p25 = plot(sigk_log(1,1:laz1),sigk_log(4,1:laz1),'-b');
title('Plano de estados (muestra = [1,500])','FontSize',16);
xlabel('ex','FontSize',16);
ylabel('dex/dt','FontSize',16);
set(p25(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(1,1:laz1)) max(sigk_log(1,1:laz1)) -max(sigk_log(4,1:laz1))
max(sigk_log(4,1:laz1))])

subplot(3,2,2);
p26 = plot(sigk_log(1,laz1:nt),sigk_log(4,laz1:nt),'-b');
title('Plano de estados (muestra = [500,5226])','FontSize',16);
xlabel('ex','FontSize',16);
ylabel('dex/dt','FontSize',16);
set(p26(1),'LineWidth',1.5);
grid on
```

```
axis([-max(sigk_log(1,laz1:nt)) max(sigk_log(1,laz1:nt)) -max(sigk_log(4,laz1:nt))
max(sigk_log(4,laz1:nt))])

subplot(3,2,3);
laz1 = 2000;
p27 = plot(sigk_log(2,1:laz1),sigk_log(5,1:laz1),'-g');
title('Plano de estados (muestra = [1,2000])','FontSize',16);
xlabel('ey','FontSize',16);
ylabel('dey/dt','FontSize',16);
set(p27(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(2,1:laz1)) max(sigk_log(2,1:laz1)) -max(sigk_log(5,1:laz1))
max(sigk_log(5,1:laz1))])

subplot(3,2,4);
p28 = plot(sigk_log(2,laz1:nt),sigk_log(5,laz1:nt),'-g');
title('Plano de estados (muestra = [2000,5226])','FontSize',16);
xlabel('ey','FontSize',16);
ylabel('dey/dt','FontSize',16);
set(p28(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(2,laz1:nt)) max(sigk_log(2,laz1:nt)) -max(sigk_log(5,laz1:nt))
max(sigk_log(5,laz1:nt))])

subplot(3,2,5);
laz1 = 100;
p29 = plot(sigk_log(3,1:laz1),sigk_log(6,1:laz1),'-r');
title('Plano de estados (muestra = [1,100])','FontSize',16);
xlabel('ez','FontSize',16);
ylabel('dez/dt','FontSize',16);
set(p29(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(3,1:laz1)) max(sigk_log(3,1:laz1)) -max(sigk_log(6,1:laz1))
max(sigk_log(6,1:laz1))])

subplot(3,2,6);
p30 = plot(sigk_log(3,laz1:nt),sigk_log(6,laz1:nt),'-r');
title('Plano de estados (muestra = [100,5226])','FontSize',16);
xlabel('ez','FontSize',16);
ylabel('dez/dt','FontSize',16);
set(p30(1),'LineWidth',1.5);
grid on
axis([-max(sigk_log(3,laz1:nt)) max(sigk_log(3,laz1:nt)) -max(sigk_log(6,laz1:nt))
max(sigk_log(6,laz1:nt))])
```

Apéndice 2.15: Diferenciador Robusto Exacto y Uniforme filtrado (UREDF)

```
%==========================================================================
%
% Título            : URED_F.m
% Propósito         : Función del diferenciador robusto exacto estándar
%                     y uniforme filtrado
%
% Descripción       : Este programa permite filtrar el ruido de la variable
%                     de estado de posición medida de un blanco aéreo
%                     de alta maniobrabilidad en coordenadas cartesianas
%                     obtenidas por un radar de seguimiento automático.
%                     Para tal efecto, el diferenciador cuenta con una
%                     parte de filtrado estándar y una parte de
%                     diferenciación uniforme, las cuales son conmutadas
%                     mediante una función sign que depende del valor de
%                     la variable deslizante filtrada. Cuando el diferen-
%                     ciador conmuta al algoritmo uniforme utiliza un
%                     filtrado de ventana deslizante adaptable de primer
%                     orden (FOAW), el cual aumenta progresivamente el
%                     tamaño de la ventana de filtrado si es que las
%                     mediciones se encuentran dentro de la capa límite
%                     de error máximo emax. Después de obtener la ventana
%                     de filtrado óptimo "nn", computa la media de las
%                     últimas "nn" mediciones.
%
% Fecha de creación : 10/03/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
```

```
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
function [xhFREUD]  = URED_F(yk,T,yk_log,batch_ykx,batch_yky,batch_ykz,intra,k)

persistent z0x z1x w1x w2x w3x z0y z1y w1y w2y w3y z0z z1z w1z w2z w3z...
           w3x_dot w3y_dot w3z_dot mx my mz

if isempty (z0x)
   z0x = yk(1,1);
   z1x = 0;
   w1x = 0;
   w2x = 0;
   w3x = 0;
   z0y = 0.98*yk(3,1);
   z1y = 0;
   w1y = 0;
   w2y = 0;
   w3y = 0;
   z0z = 0;%yk(5,1);
   z1z = 0;
   w1z = 0;
   w2z = 0;
   w3z = 0;
   w3x_dot = 0;
   w3y_dot = 0;
   w3z_dot = 0;
   mx = 1;
   my = 1;
   mz = 1;
end
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar filtrado y uniforme: coordenada x
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0x           posición
% z1x           velocidad
% z2x           aceleración
% z3x           sobre aceleración
% z4x           snap (derivada de la sobre aceleración)
% Lx            Constante de Lipschitz (constante)
% yk(1,1)       posición medida en la coordenada "x"
% T             Tiempo de muestreo
% nx            Orden de diferenciación
% nfx           Orden de filtrado
% epsx          constante de cambio de variable deslizante
% k0x,k1x,...   ganancias del diferenciador
% thetax        función de transición del diferenciador
% sigx          variable deslizante
% sigdx         derivada de la variable deslizante
% mux1,mux2     pesos del diferenciador uniforme
% nmax          máxima ventana deslizante (sliding window)
% emax          error máximo permisible
% Párametros
%
k0x   = 1.1; k1x = 4.57;k2x = 9.30;k3x = 10.03;k4x = 5.5;
nx    = 1; nfx  = 3;
numx  = nx + nfx; denx = nx + nfx + 1;
Lx    = 1;
mux1  = 5.0138/1.05;
mux2  = 2.0466/1.05;
epsx  = 1;
%-------------------------------------------------------------------------
% Filtro de mediana
%-------------------------------------------------------------------------
switch intra
    case 0
        if  (k > 4*mx)
            if mod(k,2) == 0
                n = k - (2*mx + 1);
            else
                n = k - (2*mx);
            end
            ykx_filt      = median(yk_log(1,n-mx:n+mx));
```

```
        else
            ykx_filt      = yk(1,1);
        end
    case 1
            ykx_filt      = median(batch_ykx(k,:));
end
%-------------------------------------------------------------------------
% Función de conmutación filtrado a uniforme
%-------------------------------------------------------------------------
if ((abs(w3x_dot) > epsx))
        thetax = 0;
        mx = 1;
        sigx   = -(ykx_filt - z0x);
else
        thetax = 1;
        mx = 2;
        sigx   = -(yk(1,1) - z0x);
end
%-------------------------------------------------------------------------
% Diferenciador filtrado y uniforme coordenada x
%-------------------------------------------------------------------------
    w1x_dot = -thetax*k4x*(Lx^((numx-3)/(denx)))*((abs(w1x)^((numx-
0)/(denx)))*sign(w1x)) + thetax*w2x;
    w2x_dot = -thetax*k3x*(Lx^((numx-2)/(denx)))*((abs(w1x)^((numx-
1)/(denx)))*sign(w1x)) + thetax*w3x;
    w3x_dot = -thetax*k2x*(Lx^((numx-1)/(denx)))*((abs(w1x)^((numx-
2)/(denx)))*sign(w1x)) + sigx;
    z0x_dot = -thetax*k1x*(Lx^((numx-0)/(denx)))*((abs(w1x)^((numx-
3)/(denx)))*sign(w1x)) - k1x*(1-thetax)*(Lx^((numx-
0)/(denx)))*mux1*((abs(w3x_dot))^((denx)/(denx-1)))*sign(w3x_dot) + z1x;
    z1x_dot = -thetax*k0x*(Lx^((numx+1)/(denx)))*((abs(w1x)^((numx-
4)/(denx)))*sign(w1x)) - k0x*(1-
thetax)*(Lx^((numx+1)/(denx)))*mux2*((abs(w3x_dot))^((denx+1)/(denx-1)))*sign(w3x_dot);
    % Integración de términos
    w1x_1   = w1x + w1x_dot*T;
    w2x_1   = w2x + w2x_dot*T;
    w3x_1   = w3x + w3x_dot*T;
    % Integración del diferenciador
    z0x_1   = z0x + z0x_dot*T;
    z1x_1   = z1x + z1x_dot*T;
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar filtrado y uniforme: coordenada y
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0y           posición
% z1y           velocidad
% z2y           aceleración
% z3y           sobre aceleración
% z4y           snap (derivada de la sobre aceleración)
% Ly            Constante de Lipschitz (constante)
% yk(1,1)       posición medida en la coordenada "y"
% T             Tiempo de muestreo
% ny            Orden de diferenciación
% nfy           Orden de filtrado
% epsy          constante de cambio de variable deslizante
% k0y,k1y,...   ganancias del diferenciador
% thetay        función de transición del diferenciador
% sigy          variable deslizante
% sigdy         derivada de la variable deslizante
% muy1,muy2     pesos del diferenciador uniforme
% nmax          máxima ventana deslizante (sliding window)
% emax          error máximo permisible
% Párametros
%
k0y   = 1.1; k1y = 4.57;k2y = 9.30;k3y = 10.03;k4y = 5.5;
ny    = 1; nfy  = 3;
numy  = ny + nfy; deny = ny + nfy + 1;
Ly    = 1;
epsy  = 1;
muy1  = 5.0138/1.1;
muy2  = 2.0466/0.1;
%-------------------------------------------------------------------------
% Filtro de mediana
%-------------------------------------------------------------------------
switch intra
```

```
    case 0
        if ((k > 4*my) && (my>0))
            if mod(k,2) == 0
                n = k - (2*my + 1);
            else
                n = k - (2*my);
            end
            Sy            = yk_log(3,n-my:n+my);
            yky_filt      = median(Sy);
        else
            yky_filt      = yk(3,1);
        end
    case 1
            Sy            = batch_yky(k,:);
            yky_filt      = median(Sy);
end
%-------------------------------------------------------------------------
% Función de conmutación filtrado a uniforme
%-------------------------------------------------------------------------
    if ((abs(w3y_dot) > epsy))
        thetay = 0;
        my = 1;
        sigy   = -(yky_filt - z0y);
    else
        thetay = 1;
        my = 2;
        sigy   = -(yk(3,1) - z0y);
    end
%-------------------------------------------------------------------------
% Diferenciador filtrado y uniforme coordenada y
%-------------------------------------------------------------------------
    w1y_dot = -thetay*k4y*(Ly^((numy-3)/(deny)))*((abs(w1y)^((numy-
0)/(deny)))*sign(w1y)) + thetay*w2y;
    w2y_dot = -thetay*k3y*(Ly^((numy-2)/(deny)))*((abs(w1y)^((numy-
1)/(deny)))*sign(w1y)) + thetay*w3y;
    w3y_dot = -thetay*k2y*(Ly^((numy-1)/(deny)))*((abs(w1y)^((numy-
2)/(deny)))*sign(w1y)) + sigy;
    z0y_dot = -thetay*k1y*(Ly^((numy-0)/(deny)))*((abs(w1y)^((numy-
3)/(deny)))*sign(w1y)) - k1y*(1-thetay)*(Ly^((numy-
0)/(deny)))*muy1*((abs(w3y_dot))^((deny)/(deny-1)))*sign(w3y_dot) + z1y;
    z1y_dot = -thetay*k0y*(Ly^((numy+1)/(deny)))*((abs(w1y)^((numy-
4)/(deny)))*sign(w1y)) - k0y*(1-
thetay)*(Ly^((numy+1)/(deny)))*muy2*((abs(w3y_dot))^((deny+1)/(deny-1)))*sign(w3y_dot);
    % Integración de términos
    w1y_1   = w1y + w1y_dot*T;
    w2y_1   = w2y + w2y_dot*T;
    w3y_1   = w3y + w3y_dot*T;
    % Integración del diferenciador
    z0y_1   = z0y + z0y_dot*T;
    z1y_1   = z1y + z1y_dot*T;
%-------------------------------------------------------------------------
% Diferenciador robusto exacto estándar filtrado y uniforme: coordenada z
%-------------------------------------------------------------------------
%
% Variables del diferenciador
%
% z0z           posición
% z1z           velocidad
% z2z           aceleración
% z3z           sobre aceleración
% z4z           snap (derivada de la sobre aceleración)
% Lz            Constante de Lipschitz (constante)
% yk(4,1)       posición medida en la coordenada "x"
% T             Tiempo de muestreo
% nz            Orden de diferenciación
% nfz           Orden de filtrado
% epsz          constante de cambio de variable deslizante
% k0z,k1z,...   ganancias del diferenciador
% thetaz        función de transición del diferenciador
% sigz          variable deslizante
% sigdz         derivada de la variable deslizante
% muz           peso del diferenciador uniforme
% nmax          máxima ventana deslizante (sliding window)
% emax          error máximo permisible
% Párametros
%
k0z = 1.1; k1z = 4.57;k2z = 9.30;k3z = 10.03;k4z = 5.5;
```

```
nz   = 1; nfz  = 3;
numz = nz + nfz; denz = nz + nfz + 1;
Lz   = 1;
epsz = 0.5;
muz1 = 5.0138/1.1;
muz2 = 2.0466/0.03;
%-------------------------------------------------------------------------
% Filtro de mediana
%-------------------------------------------------------------------------
switch intra
    case 0
        if ((k > 4*mz) && (mz>0))
            if mod(k,2) == 0
                n = k - (2*mz + 1);
            else
                n = k - (2*mz);
            end
            Sz            = yk_log(5,n-mz:n+mz);
            ykz_filt      = median(Sz);
        else
            ykz_filt      = yk(5,1);
        end
    case 1
            Sz            = batch_ykz(k,:);
            ykz_filt      = median(Sz);
end
%-------------------------------------------------------------------------
% Función de conmutación filtrado a uniforme
%-------------------------------------------------------------------------
    if ((abs(w3z_dot) > epsz))
        thetaz = 0;
        mz = 1;
        sigz   = -(ykz_filt - z0z);
    else
        thetaz = 1;
        mz = 2;
        sigz   = -(yk(5,1) - z0z);
    end
%-------------------------------------------------------------------------
% Diferenciador filtrado y uniforme coordenada z
%-------------------------------------------------------------------------
    w1z_dot = -thetaz*k4z*(Lz^((numz-3)/(denz)))*((abs(w1z)^((numz-
0)/(denz)))*sign(w1z)) + thetaz*w2z;
    w2z_dot = -thetaz*k3z*(Lz^((numz-2)/(denz)))*((abs(w1z)^((numz-
1)/(denz)))*sign(w1z)) + thetaz*w3z;
    w3z_dot = -thetaz*k2z*(Lz^((numz-1)/(denz)))*((abs(w1z)^((numz-
2)/(denz)))*sign(w1z)) + sigz;
    z0z_dot = -thetaz*k1z*(Lz^((numz-0)/(denz)))*((abs(w1z)^((numz-
3)/(denz)))*sign(w1z)) - k1z*(1-thetaz)*(Lz^((numz-
0)/(denz)))*muz1*((abs(w3z_dot))^((denz)/(denz-1)))*sign(w3z_dot) + z1z;
    z1z_dot = -thetaz*k0z*(Lz^((numz+1)/(denz)))*((abs(w1z)^((numz-
4)/(denz)))*sign(w1z)) - k0z*(1-
thetaz)*(Lz^((numz+1)/(denz)))*muz2*((abs(w3z_dot))^((denz+1)/(denz-1)))*sign(w3z_dot);
    % Integración de términos
    w1z_1   = w1z + w1z_dot*T;
    w2z_1   = w2z + w2z_dot*T;
    w3z_1   = w3z + w3z_dot*T;
    % Integración del diferenciador
    z0z_1   = z0z + z0z_dot*T;
    z1z_1   = z1z + z1z_dot*T;
%-------------------------------------------------------------------------
% Guardar variables para siguiente iteración
%-------------------------------------------------------------------------
z0x = z0x_1;
z1x = z1x_1;
w1x = w1x_1;
w2x = w2x_1;
w3x = w3x_1;
z0y = z0y_1;
z1y = z1y_1;
w1y = w1y_1;
w2y = w2y_1;
w3y = w3y_1;
z0z = z0z_1;
z1z = z1z_1;
w1z = w1z_1;
w2z = w2z_1;
```

```
w3z = w3z_1;
xhFREUD   = [z0x;z1x;z0y;z1y;z0z;z1z];
end
```

Apéndice 2.16: Procesamiento intra pulso de posición

```
function [batch_ykx,batch_yky,batch_ykz] =
batch_intra_pulso(xk_log,yk_log,polar_log,Xgm12t,Xgm34t,nro_intra_pulsos)
%=========================================================================
%
% Título            : batch_intra_pulso.m
% Propósito         : Generar mediciones ruidosas de radar intra pulso en
%                     ceptas o batches
%
% Descripción       : Este programa permite la generación de mediciones
%                     ruidosas de posición en coordenadas cartesianas
%                     en secuencias de 36 pulsos, intra pulso para el
%                     filtrado por medio de un filtro de mediana. Se asume
%                     que el PRF del radar de control de tiro es de 1800
%                     Hz, lo que permite obtener 36 pulsos o retornos del
%                     blanco en un período de 0.02 seg después de haberse
%                     procesado el pulso y[k-1](Este período es el
%                     período de muestreo del algoritmo de estimación de
%                     variables de estado, el cual solo procesa un pulso
%                     cada 0.02 segs). A la secuencia de nro_intra_pulsos+1 pulsos,
contando
%                     el pulso anterior y[k-1] se le
%                     aplica la operación de mediana y el resultado de esta
%                     operación ingresa al algoritmo de estimación
%
% Fecha de creación : 19/03/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
%-------------------------------------------------------------------------
% Constantes
%-------------------------------------------------------------------------
% número de puntos
nt = 5226;
% longitud de subsecuencias utilizadas para la generación del modelo mixto
% gaussiano
nt_set  = 52;
nt_set2 = 26;
a  = 0;
j  = 1;
%-------------------------------------------------------------------------
% Procesamiento en batches de Alcance (Ad), Acimut verdadero (Bdn) y
% elevación (Ed)
%-------------------------------------------------------------------------
for k = 2:nt
    % Es el pulso anterior y[k-1] procesado por el observador
    last_paint_Ad     = polar_log(1,k-1);
    % Es el pulso actual y[k] procesado por el observador
    actual_paint_Ad   = polar_log(1,k);
    % Generar un espacio lineal entre el pulso anterior y[k-1] y el pulso
    % actual y[k] a fin de generar una secuencia de la siguiente manera:
    % S = {y[k-1] p1 p2 p3 .... p35 y[k]}, donde p1,p2,... son los intra
    % pulsos no procesados por el algoritmo de estimación. Estos pulsos,
    % por el PRF del radar, son obtenidos pero nunca son procesados dado que
    % cada 0.02 segs el observador solo puede procesar 1 pulso.
    batch_Ad        = linspace(last_paint_Ad,actual_paint_Ad,nro_intra_pulsos+1);

    last_paint_Bdn   = polar_log(2,k-1);
    actual_paint_Bdn = polar_log(2,k);
    batch_Bdn        = linspace(last_paint_Bdn,actual_paint_Bdn,nro_intra_pulsos+1);
```

```
    last_paint_Ed    = polar_log(3,k-1);
    actual_paint_Ed  = polar_log(3,k);
    batch_Ed         = linspace(last_paint_Ed,actual_paint_Ed,nro_intra_pulsos+1);
%-------------------------------------------------------------------------
% Agregar ruido a los batches para simular los 36 pulsos de radar
%-------------------------------------------------------------------------
    if (xk_log(7,k)/0.00054 >= 5)

        % la distribución gaussiana mixta se encuentra dividida en 100
        % grupos de 52 y 1 grupo de 26. Si el índice de la muestra actual k
        % es menor que 52, agrega la distribución gaussiana mixta
        % correspondiente a las primeras 52 muestras. De lo contrario,
        % incrementa el índice j para utilizar la siguiente distribución
        % gaussiana mixta. Así, si los batches mantienen las propiedades de
        % ruido gaussiana mixto por secuencia.

        if (k <= nt_set*j)
            Ad_noise  = Xgm12t(3,a+1:a+length(batch_Ad));                          % ruido
en alcance (mts)
            Bdn_noise = Xgm12t(2,a+1:a+length(batch_Bdn))*0.0572958;               % ruido
en acimut (grados)
            Ed_noise  = Xgm12t(1,a+1:a+length(batch_Ed))*0.0572958;                % ruido
en elevación (grados)
            if k == nt_set*j
               if k == 5200
                  a = a + 26;
                  nt_set = nt_set2;
               else
                  a = a + 52;
               end
               j = j + 1;
            end
        end
        Adh = (batch_Ad + Ad_noise).*cosd(batch_Ed + Ed_noise);
        batch_ykx(k,:) = Adh.*cosd(batch_Bdn + Bdn_noise);
        batch_yky(k,:) = Adh.*sind(batch_Bdn + Bdn_noise);
        batch_ykz(k,:) = (batch_Ad + 0.05*Ad_noise).*sind(batch_Ed + 0.05*Ed_noise);
    else
        if (k <= nt_set*j)
            Ad_noise  = Xgm34t(3,a+1:a+length(batch_Ad));      % ruido en alcance (mts)
            Bdn_noise = Xgm34t(2,a+1:a+length(batch_Bdn))*0.0572958; % ruido en acimut
(grados)
            Ed_noise  = Xgm34t(1,a+1:a+length(batch_Ed))*0.0572958;                % ruido
en elevación (grados)
            if k == nt_set*j
               if k == 5200
                  a = a + 26;
                  nt_set = nt_set2;
               else
                  a = a + 52;
               end
            end
            j = j + 1;
        end
        Adh = (batch_Ad + Ad_noise).*cosd(batch_Ed + Ed_noise);
        batch_ykx(k,:) = Adh.*cosd(batch_Bdn + Bdn_noise);
        batch_yky(k,:) = Adh.*sind(batch_Bdn + Bdn_noise);
        batch_ykz(k,:) = (batch_Ad + 0.05*Ad_noise).*sind(batch_Ed + 0.05*Ed_noise);
    end
end
batch_ykx(1,:) = yk_log(1,1).*ones(1,nro_intra_pulsos+1);
batch_yky(1,:) = yk_log(3,1).*ones(1,nro_intra_pulsos+1);
batch_ykz(1,:) = yk_log(5,1).*ones(1,nro_intra_pulsos+1);
```

Apéndice 2.17: Simulación de la trayectoria del misil (UREDF)

```
%=========================================================================
%
% Título            : sim_URED_F.m
% Propósito         : Simulación de diferenciador robusto exacto
%
% Descripción       : Este programa permite la diferenciación robusta exacta
%                     y uniforme y filtración de la medición ruidosa de
%                     posición de un blanco aéreo de alta maniobrabilidad
%                     en coordenadas cartesianas. Para tal efecto, se compa
```

```
%                       ra el desempeño de filtración de ruido angular o glint
%                       noise y robustez ante perturbaciones con el de los
%                       filtros Kalman lineal (KF), Cubature Kalman Filter
%                       (CKF) y Particle Filter (PF).
%
% Observaciones:        Para poder ejecutar este programa es necesario contar
%                       con el toolbox de Sensor Fusion and Tracking toolbox
%                       (kalman filter y cubature kalman filter)
%                       y el control systems toolbox (particle filter)
% Fecha de creación : 15/03/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
clear all;
clc;
close all;
%-------------------------------------------------------------------------
% Inicialización de variables
%-------------------------------------------------------------------------
RMSE_URED_Fc_old = 0;
RMSE_KFc_old = 0;
RMSE_PFc_old = 0;
RMSE_CKFc_old = 0;
RMSE_URED_Fp_old = 0;
RMSE_KFp_old = 0;
RMSE_PFp_old = 0;
RMSE_CKFp_old = 0;
yk_log = [0;0;0;0;0;0];
%-------------------------------------------------------------------------
% Parámetros iniciales de radar (sensor)
%-------------------------------------------------------------------------
T      = 1/50;                      % Período de muestreo del observador (seg)
PRF    = 1800;                      % Pulsos/seg transmitidos
Batch  = PRF*T;
%-------------------------------------------------------------------------
% Definición de constantes físicas
%-------------------------------------------------------------------------
g     = 9.81;                      % gravedad (m/s2)
M     = 343;                       % velocidad del sonido (m/s)
load_factor = 100*g;               % máxima sobre aceleración (g/s)
%-------------------------------------------------------------------------
% Definición de vectores de tiempo
%-------------------------------------------------------------------------
ti    = 0;                         % Tiempo inicial de simulación
tff   = 104.5;                     % Tiempo final de simulación
tt    = ti:T:tff;                  % Vector de tiempo
tt    = tt';                       % Transpuesta del vector de tiempo
nt    = length(tt);                % Número de muestras
%-------------------------------------------------------------------------
% Definición de posición y velocidad inicial del blanco en
% coordenadas esféricas
%-------------------------------------------------------------------------
Ad_init  = 35000;                      % Distancia al blanco (m)
Ed_init  = 0.01;                       % Elevación al blanco (°)
Bdn_init = 30;                         % Marcación al blanco (°)
Vm_init  = 0.9*M;                      % Velocidad del blanco (1 Mach = 343 m/s)
Rv_init  = 210;                        % Rumbo inicial del blanco (°)
%-------------------------------------------------------------------------
% Conversión de coordenadas esféricas a cartesianas (North-Sky-East) de la
% posición y velocidad inicial del blanco
%-------------------------------------------------------------------------
[Adx_init, Ady_init, Adz_init] = p2c(Ad_init,Ed_init,Bdn_init);
[Vmx_init, Vmy_init, Vmz_init] = p2c(Vm_init,0,Rv_init);
%-------------------------------------------------------------------------
% Inicialización de vector de estados
%-------------------------------------------------------------------------
xk = [Adx_init Vmx_init 0*g Ady_init Vmy_init 0*g Adz_init Vmz_init 0*g]';
%-------------------------------------------------------------------------
% Definición de matrices del modelo de espacio de estados del sistema
%-------------------------------------------------------------------------
% Matriz de medición (Pulse radar)
Ck     = [1 0 0 0 0 0 0 0 0;        % pos x
          0 0 0 0 0 0 0 0 0;        % vel x
```

```
          0 0 0 1 0 0 0 0 0;        % pos y
          0 0 0 0 0 0 0 0 0;        % vel y
          0 0 0 0 0 0 1 0 0;        % pos z
          0 0 0 0 0 0 0 0 0];       % vel z

% Cálculo de matriz de transición Fk
Fk = [1 T (T^2)/2 0 0       0 0 0       0;
      0 1       T 0 0       0 0 0       0;
      0 0       1 0 0       0 0 0       0;
      0 0       0 1 T (T^2)/2 0 0       0;
      0 0       0 0 1       T 0 0       0;
      0 0       0 0 0       1 0 0       0;
      0 0       0 0 0       0 1 T (T^2)/2;
      0 0       0 0 0       0 0 1       T;
      0 0       0 0 0       0 0 0       1];

% Cálculo de la matriz de entrada Gk
Gk = [(T^3)/6       0        0;
      (T^2)/2       0        0;
       T            0        0;
       0       (T^3)/6       0;
       0       (T^2)/2       0;
       0            T        0;
       0            0   (T^3)/6;
       0            0   (T^2)/2;
       0            0        T];

% Matriz de distribución de perturbaciones (aceleraciones)
Dk = [75e4*(T^3)/6      0                0;
      5e3*(T^2)/2       0                0;
       1e2*T            0                0;
       0       75e4*(T^3)/6              0;
       0        5e3*(T^2)/2              0;
       0             1e2*T               0;
       0                 0     75e4*(T^3)/6;
       0                 0      5e3*(T^2)/2;
       0                 0           1e2*T];
%-------------------------------------------------------------------------
% Generación del vector de sobre aceleraciones de entrada uk
%-------------------------------------------------------------------------
uk = gen_jerk_sea_skimming3(nt,load_factor,g,T);
%-------------------------------------------------------------------------
% Generación del vector de perturbaciones ?k
%-------------------------------------------------------------------------
pk = perturb_1(g,ti,T,tff);
%-------------------------------------------------------------------------
% Cargar ruido glint
%-------------------------------------------------------------------------
load glint_puntos.mat
%-------------------------------------------------------------------------
% Cargar secuencias intra pulso
%-------------------------------------------------------------------------
load xk_log; load polar_log; load yk_log;
nro_intra_pulsos = (PRF/2)*T;
[batch_ykx,batch_yky,batch_ykz] =
batch_intra_pulso(xk_log,yk_log,polar_log,Xgm12t,Xgm34t,nro_intra_pulsos);
%-------------------------------------------------------------------------
% Inicialización Filtro Kalman lineal (KF)
%-------------------------------------------------------------------------
initialState = 0.5*xk;
KF = trackingKF('MotionModel','3D Constant Acceleration','State',initialState);
%-------------------------------------------------------------------------
% Inicialización Filtro Kalman Cubature (CKF)
%-------------------------------------------------------------------------
CKF = trackingCKF(@constacc,@cameas,initialState);
%-------------------------------------------------------------------------
% Inicialización Filtro de Partículas (PF)
%-------------------------------------------------------------------------
stateCov = diag([0.5 0.001 0.001 0.0001 0.001 0.001 0.7 0.001 0.001]);
myPF = particleFilter(@contaccParticleFilterStateFcn,@contaccMeasurementLikelihoodFcn);
initialize(myPF,2000,xk,stateCov);
myPF.StateEstimationMethod = 'mean';
myPF.ResamplingMethod = 'residual';
%-------------------------------------------------------------------------
% Simulación del modelo
%-------------------------------------------------------------------------
disp('Diferenciador Robusto Exacto Estándar y Uniforme Filtrado intra pulso');
```

```
gl     = 1;
prompt = 'Con filtro de mediana intrapulso [1], Sin filtro de mediana intrapulso [0]:';
intra  = input(prompt);
for k = 1:nt

    % Modelo dinámico o de transición de estados
    xk       = Fk*xk + Gk*uk(:,k) + Dk*pk(:,k);
    % Simulación de adquisición de datos por el radar
    % Datos de posición adquiridos en coordenadas esféricas
    [Ad,Bdn,Ed] = c2p(xk);
    % Modelo de medición polar a cartesiano
    yk = yk_noise(Ad,Bdn,Ed,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,gl,k);
    % Diferenciador Robusto Exacto y Uniforme Filtrado
    tic
    [xhURED_F]  = URED_F(yk,T,yk_log,batch_ykx,batch_yky,batch_ykz,intra,k);
    TCPU_URED_F = toc;
    % Kalman filter lineal (KF)
    tic
    [xhKF,phKF] = predict(KF,T); xhKF        = xhKF';
    [xcKF,pcKF] = correct(KF,[yk(1,1),yk(3,1),yk(5,1)]');
    TCPU_KF = toc;
    % Kalman cubature filter (CKF)
    tic
    [xhCKF,phCKF] = predict(CKF,T);
    [xcCKF,pcCKF] = correct(CKF,[yk(1,1),yk(3,1),yk(5,1)]');
    TCPU_CKF = toc;
    % Particle filter (PF)
    tic
    [xhPF,phPF] = correct(myPF,[yk(1,1), yk(3,1), yk(5,1)]');
    [xcPF,pcPF] = predict(myPF,uk,pk,Fk,Gk,Dk,k);
    TCPU_PF = toc;
    % Conversión de coordenada estimadas cartesianas a coordenadas polares
    [AdhURED_F,BdnhURED_F,EdhURED_F] = c2p_filt(xhURED_F);
    [AdhKF,BdnhKF,EdhKF]             = c2p(xhKF);
    [AdhPF,BdnhPF,EdhPF]             = c2p(xhPF);
    [AdhCKF,BdnhCKF,EdhCKF]          = c2p(xhCKF);
    % Cálculo de errores en coordenadas cartesianas
    errorc_URED_F(1,1)      = xhURED_F(1,1)  - xk(1,1);
    errorc_URED_F(2,1)      = xhURED_F(2,1)  - xk(2,1);
    errorc_URED_F(3,1)      = xhURED_F(3,1)  - xk(4,1);
    errorc_URED_F(4,1)      = xhURED_F(4,1)  - xk(5,1);
    errorc_URED_F(5,1)      = xhURED_F(5,1)  - xk(7,1);
    errorc_URED_F(6,1)      = xhURED_F(6,1)  - xk(8,1);
    errorc_KF              = xhKF  - xk;
    errorc_PF              = xhPF  - xk;
    errorc_CKF             = xhCKF - xk;
    % Cálculo de errores en coordenadas esféricas
    errorAdhURED_F     = AdhURED_F - Ad;                                    %mts
    errorBdnhURED_F    = BdnhURED_F*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEdhURED_F     = EdhURED_F*(pi*1000/180) - Ed*(pi*1000/180);
    errorp_URED_F      = [errorAdhURED_F; errorBdnhURED_F; errorEdhURED_F];
    errorAdhKF        = AdhKF - Ad;                                        %mts
    errorBdnhKF       = BdnhKF*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEdhKF        = EdhKF*(pi*1000/180) - Ed*(pi*1000/180);
    errorp_KF         = [errorAdhKF; errorBdnhKF; errorEdhKF];
    errorAdhPF        = AdhPF - Ad;                                        %mts
    errorBdnhPF       = BdnhPF*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEdhPF        = EdhPF*(pi*1000/180) - Ed*(pi*1000/180);
    errorp_PF         = [errorAdhPF; errorBdnhPF; errorEdhPF];
    errorAdhCKF       = AdhCKF - Ad;                                        %mts
    errorBdnhCKF      = BdnhCKF*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEdhCKF       = EdhCKF*(pi*1000/180) - Ed*(pi*1000/180);
    errorp_CKF        = [errorAdhCKF; errorBdnhCKF; errorEdhCKF];
    % Cálculo de RMSE
    if (k >= 4650)
        RMSE_URED_Fc        = RMSE_URED_Fc_old + ((errorc_URED_F).^2);
        RMSE_KFc            = RMSE_KFc_old + ((errorc_KF).^2);
        RMSE_PFc            = RMSE_PFc_old + ((errorc_PF).^2);
        RMSE_CKFc           = RMSE_CKFc_old + ((errorc_CKF).^2);
        RMSE_URED_Fp        = RMSE_URED_Fp_old + ((errorp_URED_F).^2);
        RMSE_KFp            = RMSE_KFp_old + ((errorp_KF).^2);
        RMSE_PFp            = RMSE_PFp_old + ((errorp_PF).^2);
        RMSE_CKFp           = RMSE_CKFp_old + ((errorp_CKF).^2);
    else
        RMSE_URED_Fc        = 0;
        RMSE_KFc            = 0;
        RMSE_PFc            = 0;
```

```
        RMSE_CKFc           = 0;
        RMSE_URED_Fp        = 0;
        RMSE_KFp            = 0;
        RMSE_PFp            = 0;
        RMSE_CKFp           = 0;
    end
    bin_log(:,k)      = k;
    xk_log(:,k)       = xk;
    yk_log(:,k)       = yk;
    xhURED_F_log(:,k)  = xhURED_F;
    xhKF_log(:,k)     = xhKF;
    xhCKF_log(:,k)    = xhCKF;
    xhPF_log(:,k)     = xhPF;
    errorc_URED_F_log(:,k) = errorc_URED_F;
    errorc_KF_log(:,k)    = errorc_KF;
    errorc_PF_log(:,k)    = errorc_PF;
    errorc_CKF_log(:,k)   = errorc_CKF;
    errorp_URED_F_log(:,k) = errorp_URED_F;
    errorp_KF_log(:,k)    = errorp_KF;
    errorp_PF_log(:,k)    = errorp_PF;
    errorp_CKF_log(:,k)   = errorp_CKF;
    polar_log(:,k)        = [Ad;Bdn;Ed];
    polarh_URED_F_log(:,k) = [AdhURED_F;BdnhURED_F;EdhURED_F];
    polarh_KF_log(:,k)    = [AdhKF;BdnhKF;EdhKF];
    polarh_PF_log(:,k)    = [AdhPF;BdnhPF;EdhPF];
    polarh_CKF_log(:,k)   = [AdhCKF;BdnhCKF;EdhCKF];
    TCPU(:,k)             = [TCPU_KF;TCPU_CKF;TCPU_PF;TCPU_URED_F];
    % Guardar variables
    RMSE_URED_Fc_old       = RMSE_URED_Fc;
    RMSE_KFc_old           = RMSE_KFc;
    RMSE_PFc_old           = RMSE_PFc;
    RMSE_CKFc_old          = RMSE_CKFc;
    RMSE_URED_Fp_old       = RMSE_URED_Fp;
    RMSE_KFp_old           = RMSE_KFp;
    RMSE_PFp_old           = RMSE_PFp;
    RMSE_CKFp_old          = RMSE_CKFp;
end
RMSE_KFc_tot     = sqrt(RMSE_KFc)/(nt-4650);
RMSE_CKFc_tot    = sqrt(RMSE_CKFc)/(nt-4650);
RMSE_PFc_tot     = sqrt(RMSE_PFc)/(nt-4650);
RMSE_URED_Fc_tot = sqrt(RMSE_URED_Fc)/(nt-4650);
RMSE_KFp_tot     = sqrt(RMSE_KFp)/(nt-4650);
RMSE_CKFp_tot    = sqrt(RMSE_CKFp)/(nt-4650);
RMSE_PFp_tot     = sqrt(RMSE_PFp)/(nt-4650);
RMSE_URED_Fp_tot = sqrt(RMSE_URED_Fp)/(nt-4650);
%-------------------------------------------------------------------------
% Conversión de vector de estados reales y estimados a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
% 0.10197 - de mts/seg2 a g's
conv_3         = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292
0.10197]';
conv_2         = [ 0.00054 0.00292 0.00054 0.00292 0.00054 0.00292 ]';
xk_log         = xk_log.*conv_3;
xhKF_log       = xhKF_log.*conv_3;
xhCKF_log      = xhCKF_log.*conv_3;
xhPF_log       = xhPF_log.*conv_3;
xhURED_F_log    = xhURED_F_log.*conv_2;
yk_log         = yk_log.*conv_2;
%-------------------------------------------------------------------------
% Plots
%-------------------------------------------------------------------------
%% 3D trayectoria completa
        figure(1);p1 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-
g',...
                               xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-
r',...
                               xhKF_log(1,:),      xhKF_log(4,:),      xhKF_log(7,:),'-
k',...
                               xhCKF_log(1,:),     xhCKF_log(4,:),
xhCKF_log(7,:),'-m',...
                               xhPF_log(1,:),      xhPF_log(4,:),      xhPF_log(7,:),'-
c',...
                               xhURED_F_log(1,:), xhURED_F_log(3,:),
xhURED_F_log(5,:),'-b',...
```

```
                                                0,              0,              0,
'o',...
                                  xk_log(1,1),        xk_log(4,1),        xk_log(7,1),
'o',...
                                  xk_log(1,nt),       xk_log(4,nt),
xk_log(7,nt),'o',...
                              0.9*xk_log(1,nt),   0.9*xk_log(4,nt),
0.9*xk_log(7,nt),'o');
         grid on
         set(p1(1),'LineWidth',2.0);
         set(p1(2),'LineWidth',3.0);
         set(p1(3),'LineWidth',2.0);
         set(p1(4),'LineWidth',3.0);
         set(p1(5),'LineWidth',3.5);
         set(p1(6),'LineWidth',4.0);
         set(p1(7),'LineWidth',3.0);
         set(p1(8),'LineWidth',3.0);
         set(p1(9),'LineWidth',3.0);
         set(p1(10),'LineWidth',3.0);

         axis([-5 20 -5 10 0 0.04]);          % Toda la trayectoria
         axis ij
         xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
         ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
         zlabel('Altitud (eje z) en [mn]','FontSize',16)
         title('Trayectoria sea skimming del misil','FontSize',16)
         legend({'medicion','real','KF','CKF','PF','URED_F','Posición del buque
observador','Posición inicial del misil',...
                 'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
            %% 3D trayectoria completa

            figure(2);p2 = plot3(  yk_log(1,:),        yk_log(3,:),
yk_log(5,:),'-g',...
                                   xk_log(1,:),        xk_log(4,:),
xk_log(7,:),'-r',...
                                   xhKF_log(1,:),      xhKF_log(4,:),
xhKF_log(7,:),'-k',...
                                   xhCKF_log(1,:),     xhCKF_log(4,:),
xhCKF_log(7,:),'-m',...
                                   xhPF_log(1,:),      xhPF_log(4,:),
xhPF_log(7,:),'-c',...
                                   xhURED_F_log(1,:), xhURED_F_log(3,:),
xhURED_F_log(5,:),'-b',...
                                   0,              0,              0, 'o',...
                                   xk_log(1,1),        xk_log(4,1),        xk_log(7,1),
'o',...
                                   xk_log(1,nt),       xk_log(4,nt),
xk_log(7,nt),'o',...
                                   0.9*xk_log(1,nt),   0.9*xk_log(4,nt),
0.9*xk_log(7,nt),'o');
            grid on
         set(p2(1),'LineWidth',2.0);
         set(p2(2),'LineWidth',3.0);
         set(p2(3),'LineWidth',2.0);
         set(p2(4),'LineWidth',3.0);
         set(p2(5),'LineWidth',3.5);
         set(p2(6),'LineWidth',4.0);
         set(p2(7),'LineWidth',3.0);
         set(p2(8),'LineWidth',3.0);
         set(p2(9),'LineWidth',3.0);
         set(p2(10),'LineWidth',3.0);
            axis([4 8 -2 5 0 0.016]);
            axis ij
            xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
            ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
            zlabel('Altitud (eje z) en [mn]','FontSize',16)
            title('Maniobra terminal del misil','FontSize',16)
            legend({'medicion','real','KF','CKF','PF','URED_F','Posición del buque
observador','Posición inicial del misil',...
                    'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)

            %% Coordenada x
            % Posición x: Trayectoria completa
            figure(3);
            subplot(1,2,1);
```

```
            p3 = plot(bin_log(1,:),yk_log(1,:),'-r',bin_log(1,:),xk_log(1,:),'-
g',bin_log(1,:),xhKF_log(1,:),'-k',bin_log(1,:),xhCKF_log(1,:),'--
m',bin_log(1,:),xhPF_log(1,:),'-c',bin_log(1,:),xhURED_F_log(1,:),'-.b');
            title('Trayectoria completa del misil: posición x','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)

legend({'medición','real','KF','CKF','PF','URED_F'},'Location','north','NumColumns',3,'F
ontSize',18)
            set(p3(1),'LineWidth', 2.5);
            set(p3(2),'LineWidth', 1.8);
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            set(p3(5),'LineWidth', 1.6);
            set(p3(6),'LineWidth', 1.6);
            grid on
            axis([1 nt min([xk_log(1,:) yk_log(1,:)]) max([xk_log(1,:) yk_log(1,:)])])

            % Posición x: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),yk_log(1,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(1,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhKF_log(1,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhCKF_log(1,lpx1:nt),'--
m',bin_log(1,lpx1:nt),xhPF_log(1,lpx1:nt),'-
c',bin_log(1,lpx1:nt),xhURED_F_log(1,lpx1:nt),'-.b');
            set(p4(1),'LineWidth', 2.5);
            set(p4(2),'LineWidth', 1.8);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            set(p4(5),'LineWidth', 1.6);
            set(p4(6),'LineWidth', 1.6);
            title('Maniobra terminal del misil: posición x','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)
            grid on
            axis([lpx1 nt min([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)])
max([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)])])
            %% Coordenada y
            % Posición y: Trayectoria completa
            figure(4);
            subplot(1,2,1);
            p5 = plot(bin_log(1,:),yk_log(3,:),'-r',bin_log(1,:),xk_log(4,:),'-
g',bin_log(1,:),xhKF_log(4,:),'-k',bin_log(1,:),xhCKF_log(4,:),'--
m',bin_log(1,:),xhPF_log(4,:),'-c',bin_log(1,:),xhURED_F_log(3,:),'-.b');
            title('Trayectoria completa del misil: posición y','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)

legend({'medición','real','KF','CKF','PF','URED_F'},'Location','northeast','NumColumns',
3,'FontSize',18)
            set(p5(1),'LineWidth', 2.5);
            set(p5(2),'LineWidth', 1.8);
            set(p5(3),'LineWidth', 1.6);
            set(p5(4),'LineWidth', 1.6);
            set(p5(5),'LineWidth', 1.6);
            set(p5(6),'LineWidth', 1.6);
            grid on
            axis([1 nt min([xk_log(4,:) yk_log(3,:)]) max([xk_log(4,:) yk_log(3,:)])])

            % Posición y: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p6 = plot(bin_log(1,lpx1:nt),yk_log(3,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(4,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhKF_log(4,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhCKF_log(4,lpx1:nt),'--
m',bin_log(1,lpx1:nt),xhPF_log(4,lpx1:nt),'-
c',bin_log(1,lpx1:nt),xhURED_F_log(3,lpx1:nt),'-.b');
            set(p6(1),'LineWidth', 2.5);
            set(p6(2),'LineWidth', 1.8);
            set(p6(3),'LineWidth', 1.6);
            set(p6(4),'LineWidth', 1.6);
            set(p6(5),'LineWidth', 1.6);
            set(p6(6),'LineWidth', 1.6);
            title('Maniobra terminal del misil: posición y','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)
```

```
            grid on
            axis([lpx1 nt min([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt)])
max([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt)])])
            %% Coordenada z
            % Posición z: Trayectoria completa
            figure(5);
            subplot(1,2,1);
            p7 = plot(bin_log(1,:),yk_log(5,:),'-r',bin_log(1,:),xk_log(7,:),'-
g',bin_log(1,:),xhKF_log(5,:),'-k',bin_log(1,:),xhCKF_log(7,:),'--
m',bin_log(1,:),xhPF_log(7,:),'--c',bin_log(1,:),xhURED_F_log(5,:),'-.b');
            title('Trayectoria completa del misil: posición z','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)

legend({'medición','real','KF','CKF','PF','URED_F'},'Location','northeast','NumColumns',
3,'FontSize',18)
            set(p7(1),'LineWidth', 2.5);
            set(p7(2),'LineWidth', 1.8);
            set(p7(3),'LineWidth', 1.6);
            set(p7(4),'LineWidth', 1.6);
            set(p7(5),'LineWidth', 1.6);
            set(p7(6),'LineWidth', 1.6);
            grid on
            axis([1 nt min([xk_log(7,:) yk_log(5,:)]) max([xk_log(7,:) yk_log(5,:)])])

            % Posición z: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p8 = plot(bin_log(1,lpx1:nt),yk_log(5,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(7,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhKF_log(7,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhCKF_log(7,lpx1:nt),'--
m',bin_log(1,lpx1:nt),xhPF_log(7,lpx1:nt),'--
c',bin_log(1,lpx1:nt),xhURED_F_log(5,lpx1:nt),'-.b');
            set(p8(1),'LineWidth', 2.5);
            set(p8(2),'LineWidth', 1.8);
            set(p8(3),'LineWidth', 1.6);
            set(p8(4),'LineWidth', 1.6);
            set(p8(5),'LineWidth', 1.6);
            set(p8(6),'LineWidth', 1.6);
            title('Maniobra terminal del misil: posición z','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)
            grid on
            axis([lpx1 nt min([xk_log(7,lpx1:nt) yk_log(5,lpx1:nt)])
max([xk_log(7,lpx1:nt) yk_log(5,lpx1:nt)])])
            %% Plots señales de error cartesianas: Coordenada x
            figure(6);
            lpx1 = 4500;
            subplot(3,2,1);
            p9 = plot(bin_log(1,:),errorc_KF_log(1,:),'-
k',bin_log(1,:),errorc_CKF_log(1,:),'--m',bin_log(1,:),errorc_PF_log(1,:),'--
c',bin_log(1,:),errorc_URED_F_log(1,:),'-.b');
            title('Error cartesiano posición x','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [m]','FontSize',16)

legend({'KF','CKF','PF','URED_F'},'Location','northeast','NumColumns',4,'FontSize',12)
            set(p9,'LineWidth',1.5);
            grid on
            axis([1 nt min([0.01*errorc_KF_log(1,:) 0.01*errorc_CKF_log(1,:)
errorc_PF_log(1,:) errorc_URED_F_log(1,:)]) max([errorc_KF_log(1,:) errorc_CKF_log(1,:)
errorc_PF_log(1,:) errorc_URED_F_log(1,:)])])

            subplot(3,2,2);
            p10 = plot(bin_log(1,lpx1:nt),errorc_KF_log(1,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorc_CKF_log(1,lpx1:nt),'--
m',bin_log(1,lpx1:nt),errorc_PF_log(1,lpx1:nt),'--
c',bin_log(1,lpx1:nt),errorc_URED_F_log(1,lpx1:nt),'-.b');
            title('Error cartesiano posición x','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [m]','FontSize',16);
            set(p10,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorc_KF_log(1,lpx1:nt) errorc_CKF_log(1,lpx1:nt)
errorc_PF_log(1,lpx1:nt) errorc_URED_F_log(1,lpx1:nt)]) max([errorc_KF_log(1,lpx1:nt)
errorc_CKF_log(1,lpx1:nt) errorc_PF_log(1,lpx1:nt) errorc_URED_F_log(1,lpx1:nt)])])
```

```
            % Plots señales de error cartesianas: Coordenada y
            subplot(3,2,3);
            p11 = plot(bin_log(1,:),errorc_KF_log(4,:),'-
k',bin_log(1,:),errorc_CKF_log(4,:),'--m',bin_log(1,:),errorc_PF_log(4,:),'--
c',bin_log(1,:),errorc_URED_F_log(3,:),'-.b');
            title('Error cartesiano posición y','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [m]','FontSize',16)

legend({'KF','CKF','PF','URED_F'},'Location','northeast','NumColumns',4,'FontSize',12)
            set(p11,'LineWidth',1.5);
            grid on
            axis([1 nt min([0.001*errorc_KF_log(4,:) 0.001*errorc_CKF_log(4,:)
errorc_PF_log(4,:) 0.1*errorc_URED_F_log(4,:)]) max([errorc_KF_log(4,:)
errorc_CKF_log(4,:) errorc_PF_log(4,:) 0.01*errorc_URED_F_log(3,:)])])

            subplot(3,2,4);
            p12 = plot(bin_log(1,lpx1:nt),errorc_KF_log(4,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorc_CKF_log(4,lpx1:nt),'--
m',bin_log(1,lpx1:nt),errorc_PF_log(4,lpx1:nt),'--
c',bin_log(1,lpx1:nt),errorc_URED_F_log(3,lpx1:nt),'-.b');
            title('Error cartesiano posición y','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [m]','FontSize',16);
            set(p12,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorc_KF_log(4,lpx1:nt) errorc_CKF_log(4,lpx1:nt)
errorc_PF_log(4,lpx1:nt) errorc_URED_F_log(3,lpx1:nt)]) max([errorc_KF_log(4,lpx1:nt)
errorc_CKF_log(4,lpx1:nt) errorc_PF_log(4,lpx1:nt) errorc_URED_F_log(3,lpx1:nt)])])

            subplot(3,2,5);
            p11 = plot(bin_log(1,:),errorc_KF_log(7,:),'-
k',bin_log(1,:),errorc_CKF_log(7,:),'--m',bin_log(1,:),errorc_PF_log(7,:),'--
c',bin_log(1,:),errorc_URED_F_log(5,:),'-.b');
            title('Error cartesiano posición z','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [m]','FontSize',16)

legend({'KF','CKF','PF','URED_F'},'Location','northeast','NumColumns',4,'FontSize',12)
            set(p11,'LineWidth',1.5);
            grid on
            axis([1 nt min([errorc_KF_log(7,:) errorc_CKF_log(7,:) errorc_PF_log(7,:)
errorc_URED_F_log(5,:)]) max([errorc_KF_log(7,:) errorc_CKF_log(7,:) errorc_PF_log(7,:)
errorc_URED_F_log(5,:)])])

            subplot(3,2,6);
            p12 = plot(bin_log(1,lpx1:nt),errorc_KF_log(7,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorc_CKF_log(7,lpx1:nt),'--
m',bin_log(1,lpx1:nt),errorc_PF_log(7,lpx1:nt),'--
c',bin_log(1,lpx1:nt),errorc_URED_F_log(5,lpx1:nt),'-.b');
            title('Error cartesiano posición z','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [m]','FontSize',16);
            set(p12,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorc_KF_log(7,lpx1:nt) errorc_CKF_log(7,lpx1:nt)
errorc_PF_log(7,lpx1:nt) errorc_URED_F_log(5,lpx1:nt)]) max([errorc_KF_log(7,lpx1:nt)
errorc_CKF_log(7,lpx1:nt) errorc_PF_log(7,lpx1:nt) errorc_URED_F_log(5,lpx1:nt)])])

            %% Plots señales de error polares: Distancia Ad
            figure(7);
            maxAd = 37.5*ones(1,nt);
            minAd = -37.5*ones(1,nt);
            lpx1 = 4500;
            subplot(3,2,1);
            p9 = plot(bin_log(1,:),errorp_KF_log(1,:),'-
k',bin_log(1,:),errorp_CKF_log(1,:),'--m',bin_log(1,:),errorp_PF_log(1,:),'--
c',bin_log(1,:),errorp_URED_F_log(1,:),'-.b',bin_log(1,:),maxAd(1,:),'--
r',bin_log(1,:),minAd(1,:),'--r');
            title('Error polar: Alcance','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [m]','FontSize',16)
            legend({'KF','CKF','PF','URED_F','Ad','Límite +/-37.5
mts'},'Location','northeast','NumColumns',4,'FontSize',12)
            set(p9,'LineWidth',1.5);
            grid on
```

```
            axis([1 nt min([errorp_PF_log(1,:) errorp_URED_F_log(1,:) minAd(1,:)])
max([errorp_KF_log(1,:) errorp_CKF_log(1,:) errorp_PF_log(1,:) errorp_URED_F_log(1,:)
maxAd(1,:)])])

            subplot(3,2,2);
            p10 = plot(bin_log(1,lpx1:nt),errorp_KF_log(1,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorp_CKF_log(1,lpx1:nt),'--
m',bin_log(1,lpx1:nt),errorp_PF_log(1,lpx1:nt),'--
c',bin_log(1,lpx1:nt),errorp_URED_F_log(1,lpx1:nt),'-.b',bin_log(1,:),maxAd(1,:),'--
r',bin_log(1,:),minAd(1,:),'--r');
            title('Error polar: Alcance','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [m]','FontSize',16);
            set(p10,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorp_KF_log(1,lpx1:nt) errorp_CKF_log(1,lpx1:nt)
errorp_PF_log(1,lpx1:nt) errorp_URED_F_log(1,lpx1:nt) minAd(1,:)])
max([errorp_KF_log(1,lpx1:nt) errorp_CKF_log(1,lpx1:nt) errorp_PF_log(1,lpx1:nt)
errorp_URED_F_log(1,lpx1:nt) maxAd(1,:)])])

            % Plots señales de error cartesianas: Coordenada y
            subplot(3,2,3);
            maxBdn = 2*ones(1,nt);
            minBdn = -2*ones(1,nt);
            p11 = plot(bin_log(1,:),errorp_KF_log(2,:),'-
k',bin_log(1,:),errorp_CKF_log(2,:),'--m',bin_log(1,:),errorp_PF_log(2,:),'--
c',bin_log(1,:),errorp_URED_F_log(2,:),'-.b',bin_log(1,:),maxBdn(1,:),'--
r',bin_log(1,:),minBdn(1,:),'--r');
            title('Error polar: Acimut','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [mrad]','FontSize',16);
            legend({'Límite +/-2 mrad'},'Location','north','NumColumns',1,'FontSize',12)
            set(p11,'LineWidth',1.5);
            grid on
            axis([1 nt min([errorp_KF_log(2,:) errorp_CKF_log(2,:) errorp_PF_log(2,:)
errorp_URED_F_log(2,:) minBdn(1,:)]) max([errorp_KF_log(2,:) errorp_CKF_log(2,:)
errorp_PF_log(2,:) errorp_URED_F_log(2,:) maxBdn(1,:)])])

            subplot(3,2,4);
            p12 = plot(bin_log(1,lpx1:nt),errorp_KF_log(2,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorp_CKF_log(2,lpx1:nt),'--
m',bin_log(1,lpx1:nt),errorp_PF_log(2,lpx1:nt),'--
c',bin_log(1,lpx1:nt),errorp_URED_F_log(2,lpx1:nt),'-.b',bin_log(1,:),maxBdn(1,:),'--
r',bin_log(1,:),minBdn(1,:),'--r');
            title('Error polar: Acimut','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [mrad]','FontSize',16);
            set(p12,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorp_KF_log(2,lpx1:nt) errorp_CKF_log(2,lpx1:nt)
errorp_PF_log(2,lpx1:nt) errorp_URED_F_log(2,lpx1:nt) minBdn(1,:)])
max([errorp_KF_log(2,lpx1:nt) errorp_CKF_log(2,lpx1:nt) errorp_PF_log(2,lpx1:nt)
errorp_URED_F_log(2,lpx1:nt) maxBdn(1,:)])])

            subplot(3,2,5);
            maxEd = 2*ones(1,nt);
            minEd = -2*ones(1,nt);
            p11 = plot(bin_log(1,:),errorp_KF_log(3,:),'-
k',bin_log(1,:),errorp_CKF_log(3,:),'--m',bin_log(1,:),errorp_PF_log(3,:),'--
c',bin_log(1,:),errorp_URED_F_log(3,:),'-.b',bin_log(1,:),maxEd(1,:),'--
r',bin_log(1,:),minEd(1,:),'--r');
            title('Error polar: Elevación','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [mrad]','FontSize',16);
            legend({'Límite +2/-2
mrad'},'Location','north','NumColumns',1,'FontSize',12)
            set(p11,'LineWidth',1.5);
            grid on
            axis([1 nt min([errorp_KF_log(3,:) errorp_CKF_log(3,:) errorp_PF_log(3,:)
errorp_URED_F_log(3,:) minEd(1,:)]) max([errorp_KF_log(3,:) errorp_CKF_log(3,:)
errorp_PF_log(3,:) errorp_URED_F_log(3,:) maxEd(1,:)])])

            subplot(3,2,6);
            p12 = plot(bin_log(1,lpx1:nt),errorp_KF_log(3,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorp_CKF_log(3,lpx1:nt),'--
m',bin_log(1,lpx1:nt),errorp_PF_log(3,lpx1:nt),'--
```

```
c',bin_log(1,lpx1:nt),errorp_URED_F_log(3,lpx1:nt),'-.b',bin_log(1,:),maxEd(1,:),'--
r',bin_log(1,:),minEd(1,:),'--r');
            title('Error polar: Elevación','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [mrad]','FontSize',16);
            set(p12,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorp_KF_log(3,lpx1:nt) errorp_CKF_log(3,lpx1:nt)
errorp_PF_log(3,lpx1:nt) errorp_URED_F_log(3,lpx1:nt) minEd(1,:)])
max([errorp_KF_log(3,lpx1:nt) errorp_CKF_log(3,lpx1:nt) errorp_PF_log(3,lpx1:nt)
errorp_URED_F_log(3,lpx1:nt) maxEd(1,:)])])

save trayectorias.mat
```

Apéndice 5

Programas de Matlab del Capítulo V

Apéndice 5.1: Simulación de la trayectoria del misil (SMO1,SMO2 vs IMM)

```
%=========================================================================
%
% Título            : sim_SMOvsIMM.m
% Propósito         : Simulación de diferenciador robusto exacto
%
% Descripción       : Este programa permite la diferenciación robusta exacta
%                     y uniforme y filtración de la medición ruidosa de
%                     posición de un blanco aéreo de alta maniobrabilidad
%                     en coordenadas cartesianas. Para tal efecto, se compa
%                     ra el desempeño de filtración de ruido angular o glint
%                     noise y robustez ante perturbaciones con el de los
%                     filtros Kalman lineal (IMM), Cubature Kalman Filter
%                     (CIMM) y Particle Filter (PF).
%
% Observaciones:      Para poder ejecutar este programa es necesario contar
%                     con el toolbox de Sensor Fusion and Tracking toolbox
%                     (kalman filter y cubature kalman filter)
%                     y el control systems toolbox (particle filter)
% Fecha de creación : 15/03/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
clear all;
clc;
close all;
%-------------------------------------------------------------------------
% Inicialización de variables
%-------------------------------------------------------------------------
RMSE_SMO1c = 0;
RMSE_SMO2c = 0;
RMSE_IMMc  = 0;
yk_log = [0;0;0;0;0;0];
fiSMO2 = [0;0;0];
%-------------------------------------------------------------------------
% Parámetros iniciales de radar (sensor)
%-------------------------------------------------------------------------
T      = 1/50;                      % Período de muestreo del observador (seg)
PRF    = 1800;                      % Pulsos/seg transmitidos
Batch  = PRF*T;                     % Longitud de secuencia intra pulso (muestras)
%-------------------------------------------------------------------------
% Definición de constantes físicas
```

```
%-------------------------------------------------------------------------
g     = 9.81;                          % gravedad (m/s2)
M     = 343;                           % velocidad del sonido (m/s)
load_factor = 100*g;                   % máxima sobre aceleración (g/s)
%-------------------------------------------------------------------------
% Definición de vectores de tiempo
%-------------------------------------------------------------------------
ti    = 0;                             % Tiempo inicial de simulación
tff   = 104.5;                         % Tiempo final de simulación
tt    = ti:T:tff;                      % Vector de tiempo
tt    = tt';                           % Transpuesta del vector de tiempo
nt    = length(tt);                    % Número de muestras
%-------------------------------------------------------------------------
% Definición de posición y velocidad inicial del blanco en
% coordenadas esféricas
%-------------------------------------------------------------------------
Ad_init  = 35000;                          % Distancia al blanco (m)
Ed_init  = 0.01;                           % Elevación al blanco (°)
Bdn_init = 30;                             % Marcación al blanco (°)
Vm_init  = 0.9*M;                          % Velocidad del blanco (1 Mach = 343 m/s)
Rv_init  = 210;                            % Rumbo inicial del blanco (°)
%-------------------------------------------------------------------------
% Conversión de coordenadas esféricas a cartesianas (North-Sky-East) de la
% posición y velocidad inicial del blanco
%-------------------------------------------------------------------------
[Adx_init, Ady_init, Adz_init] = p2c(Ad_init,Ed_init,Bdn_init);
[Vmx_init, Vmy_init, Vmz_init] = p2c(Vm_init,0,Rv_init);
%-------------------------------------------------------------------------
% Inicialización de vector de estados
%-------------------------------------------------------------------------
xk = [Adx_init Vmx_init 0*g Ady_init Vmy_init 0*g Adz_init Vmz_init 0*g]';
%-------------------------------------------------------------------------
% Definición de matrices del modelo de espacio de estados del sistema
%-------------------------------------------------------------------------
% Matriz de medición (Pulse Doppler radar - Para ESSMO)
Ck     = [1 0 0 0 0 0 0 0 0;        % pos x
          0 1 0 0 0 0 0 0 0;        % vel x
          0 0 0 1 0 0 0 0 0;        % pos y
          0 0 0 0 1 0 0 0 0;        % vel y
          0 0 0 0 0 0 1 0 0;        % pos z
          0 0 0 0 0 0 0 1 0];       % vel z

% Cálculo de matriz de transición Fk
Fk = [1 T (T^2)/2 0 0       0 0 0       0;
      0 1       T 0 0       0 0 0       0;
      0 0       1 0 0       0 0 0       0;
      0 0       0 1 T (T^2)/2 0 0       0;
      0 0       0 0 1       T 0 0       0;
      0 0       0 0 0       1 0 0       0;
      0 0       0 0 0       0 1 T (T^2)/2;
      0 0       0 0 0       0 0 1       T;
      0 0       0 0 0       0 0 0       1];

% Cálculo de la matriz de entrada Gk
Gk = [(T^3)/6       0        0;
      (T^2)/2       0        0;
       T            0        0;
       0       (T^3)/6       0;
       0       (T^2)/2       0;
       0             T       0;
       0             0  (T^3)/6;
       0             0  (T^2)/2;
       0             0        T];

% Matriz de distribución de perturbaciones (aceleraciones)
Dk = [75e4*(T^3)/6       0                0;
      5e3*(T^2)/2        0                0;
       1e2*T             0                0;
       0      75e4*(T^3)/6                0;
       0       5e3*(T^2)/2                0;
       0             1e2*T                0;
       0                 0     75e4*(T^3)/6;
       0                 0      5e3*(T^2)/2;
       0                 0           1e2*T];
%-------------------------------------------------------------------------
% Generación del vector de sobre aceleraciones de entrada uk
%-------------------------------------------------------------------------
```

```
uk = gen_jerk_sea_skimming3(nt,load_factor,g,T);
%-------------------------------------------------------------------------
% Generación del vector de perturbaciones ?k
%-------------------------------------------------------------------------
pk = perturb_1(g,ti,T,tff);
%-------------------------------------------------------------------------
% Cargar ruido glint
%-------------------------------------------------------------------------
load glint_puntos.mat
%-------------------------------------------------------------------------
% Cargar secuencias intra pulso
%-------------------------------------------------------------------------
load xk_log; load polar_log; load yk_log;
nro_intra_pulsos = (PRF)*T;
[batch_ykx,batch_yky,batch_ykz] =
batch_intra_pulso(xk_log,yk_log,polar_log,Xgm12t,Xgm34t,nro_intra_pulsos);
[batch_ykvx,batch_ykvy,batch_ykvz] =
batch_intra_pulso_vel(xk_log,yk_log,Xgm56t,Xgm78t,nro_intra_pulsos);
%-------------------------------------------------------------------------
% Inicialización del IMM
%-------------------------------------------------------------------------
detection = objectDetection(1,[0.7*Adx_init 0.7*Ady_init
1.5*Adz_init]','MeasurementNoise',10, ...
    'SensorIndex',1,'ObjectAttributes',{'Example object',5});
filter =
{initcaukf(detection);initctckf(detection);initcvpf(detection)};%;initctckf(detection);}
; %;
modelConv = @switchimm;
transProb = [0.990 0.000 0.010;% 0.00 0.00;
             0.010 0.980 0.010;% 0.01 0.00;
             0.010 0.000 0.990];
initialState = 0.5*xk;
IMM = trackingIMM('State',initialState,'StateCovariance',eye(9),...
     'TransitionProbabilities',transProb,'TrackingFilters',filter,...
     'ModelConversionFcn',modelConv);
%%
%-------------------------------------------------------------------------
% Simulación
%-------------------------------------------------------------------------
disp('Observador de modo deslizante combinado (CSMO) vs Modelo Múltiple Interactivo
(IMM)');
gl     = 1;
prompt = 'Simular con filtrado intrapulso [1], Simular sin filtrado intrapulso [0]:';
intra  = input(prompt);
%[Gl,Gn,Z3,Ao,s0] = WZSMO_sintesis_SMO2(Fk,Gk,Ck,Dk,gl);
load load_WZSMO_sintesis_SMO2.mat
for k = 1:nt

    % Modelo dinámico o de transición de estados
    xk       = Fk*xk + Gk*uk(:,k) + Dk*pk(:,k);
    % Simulación de adquisición de datos por el radar
    % Datos de posición adquiridos en coordenadas esféricas
    [Ad,Bdn,Ed] = c2p(xk);
    % Modelo de medición polar a cartesiano
    yk = yk_noise(Ad,Bdn,Ed,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,gl,k);
    % Solución Nro. 1: URED_F + ARED
    tic
    [xhURED_F]  = URED_F(yk,T,yk_log,batch_ykx,batch_yky,batch_ykz,intra,k);
    [xhSMO1,uhSMO1,duhSMO,Lk,MUk,sigSMO] = ARED(xhURED_F,T);
    TCPU_SMO1 = toc;
    % Solución Nro. 2: URED_F + ARED + ESSMO
    tic
    [xhURED_FV]        = URED_FV(yk,T,yk_log,batch_ykvx,batch_ykvy,batch_ykvz,intra,k);
    xhURED_FV_log(:,k) = xhURED_FV;
    yk_SMO2            =
[xhURED_F(1,1);xhURED_FV(1,1);xhURED_F(3,1);xhURED_FV(3,1);xhURED_F(5,1);xhURED_FV(5,1)]
;
    uk_SMO2(:,k)       = uhSMO1;
    xhSMO2             = WZSMO(Gl,Gn,Z3,Ao,Gk,yk_SMO2,uk_SMO2,k,g,s0);
    TCPU_SMO2          = toc;
    %
    % Interactive multiple model (IMM)
    tic
    [xhIMM,phIMM] = predict(IMM,T);
    [xcIMM,pcIMM] = correct(IMM,[yk(1,1), yk(3,1), yk(5,1)]');
    TCPU_IMM = toc;
    % Conversión de coordenada estimadas cartesianas a coordenadas polares
```

```
    [AdhSMO1,BdnhSMO1,EdhSMO1] = c2p(xhSMO1);
    [AdhSMO2,BdnhSMO2,EdhSMO2] = c2p(xhSMO2);
    [AdhIMM,BdnhIMM,EdhIMM] = c2p(xhIMM);
    % Cálculo de errores en coordenadas cartesianas
    errorc_SMO1      = xhSMO1   - xk;
    errorc_SMO2      = xhSMO2   - xk;
    errorc_IMM       = xhIMM - xk;
    % Cálculo de errores en coordenadas esféricas
    errorAdhSMO1     = AdhSMO1 - Ad;                                    %mts
    errorBdnhSMO1    = BdnhSMO1*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEdhSMO1     = EdhSMO1*(pi*1000/180) - Ed*(pi*1000/180);
    errorp_SMO1      = [errorAdhSMO1; errorBdnhSMO1; errorEdhSMO1];
    errorAdhSMO2     = AdhSMO2 - Ad;                                    %mts
    errorBdnhSMO2    = BdnhSMO2*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEdhSMO2     = EdhSMO2*(pi*1000/180) - Ed*(pi*1000/180);
    errorp_SMO2      = [errorAdhSMO2; errorBdnhSMO2; errorEdhSMO2];
    errorAdhIMM      = AdhIMM - Ad;                                    %mts
    errorBdnhIMM     = BdnhIMM*(pi*1000/180) - Bdn*(pi*1000/180);
    errorEdhIMM      = EdhIMM*(pi*1000/180) - Ed*(pi*1000/180);
    errorp_IMM       = [errorAdhIMM; errorBdnhIMM; errorEdhIMM];
    % Cálculo de RMSE
    if (k >= 4650)
        RMSE_SMO1c        = RMSE_SMO1c + ((errorc_SMO1).^2);
        RMSE_SMO2c        = RMSE_SMO2c + ((errorc_SMO2).^2);
        RMSE_IMMc         = RMSE_IMMc + ((errorc_IMM).^2);
    else
        RMSE_SMO1c        = 0;
        RMSE_SMO2c        = 0;
        RMSE_IMMc         = 0;
    end
    bin_log(:,k)          = k;
    xk_log(:,k)           = xk;
    yk_log(:,k)           = yk;
    xhSMO1_log(:,k)       = xhSMO1;
    xhSMO2_log(:,k)       = xhSMO2;
    fiSMO2_log(:,k)       = fiSMO2;
    xhIMM_log(:,k)        = xhIMM;
    errorc_SMO1_log(:,k)  = errorc_SMO1;
    errorc_SMO2_log(:,k)  = errorc_SMO2;
    errorc_IMM_log(:,k)   = errorc_IMM;
    errorp_SMO1_log(:,k)  = errorp_SMO1;
    errorp_SMO2_log(:,k)  = errorp_SMO2;
    errorp_IMM_log(:,k)   = errorp_IMM;
    polar_log(:,k)        = [Ad;Bdn;Ed];
    polarh_SMO1_log(:,k)  = [AdhSMO1;BdnhSMO1;EdhSMO1];
    polarh_SMO2_log(:,k)  = [AdhSMO2;BdnhSMO2;EdhSMO2];
    polarh_IMM_log(:,k)   = [AdhIMM;BdnhIMM;EdhIMM];
    TCPU(:,k)             = [TCPU_IMM;TCPU_SMO1;TCPU_SMO2];
end
RMSE_SMO1c_tot     = sqrt(RMSE_SMO1c)/(nt-4650);
RMSE_SMO2c_tot     = sqrt(RMSE_SMO2c)/(nt-4650);
RMSE_IMMc_tot      = sqrt(RMSE_IMMc)/(nt-4650);
%-------------------------------------------------------------------------
% Conversión de vector de estados reales y estimados a Mn, Mach, g's
%-------------------------------------------------------------------------
% 0.00054 - de mts a Mn
% 0.00292 - de mts/seg a Mach
% 0.10197 - de mts/seg2 a g's
conv_3         = [ 0.00054 0.00292 0.10197 0.00054 0.00292 0.10197 0.00054 0.00292
0.10197]';
conv_2         = [ 0.00054 0.00292 0.00054 0.00292 0.00054 0.00292 ]';
xk_log         = xk_log.*conv_3;
xhIMM_log      = xhIMM_log.*conv_3;
xhSMO1_log     = xhSMO1_log.*conv_3;
xhSMO2_log     = xhSMO2_log.*conv_3;
yk_log         = yk_log.*conv_2;
%-------------------------------------------------------------------------
% Plots
%-------------------------------------------------------------------------
%% 3D trayectoria completa
         figure(1);p1 = plot3(  yk_log(1,:),        yk_log(3,:),        yk_log(5,:),'-
g',...
                                xk_log(1,:),        xk_log(4,:),        xk_log(7,:),'-
r',...
                                xhIMM_log(1,:),     xhIMM_log(4,:),
xhIMM_log(7,:),'-k',...
```

```
                                xhSMO1_log(1,:),     xhSMO1_log(4,:),
xhSMO1_log(7,:),'-b',...
                                xhSMO2_log(1,:),     xhSMO2_log(4,:),
xhSMO2_log(7,:),'-m',...
                                              0,              0,              0, 'o',...
                                xk_log(1,1),        xk_log(4,1),        xk_log(7,1),
'o',...
                                xk_log(1,nt),       xk_log(4,nt),
xk_log(7,nt),'o',...
                           0.9*xk_log(1,nt),   0.9*xk_log(4,nt),
0.9*xk_log(7,nt),'o');
        grid on
        set(p1(1),'LineWidth',2.0);
        set(p1(2),'LineWidth',3.0);
        set(p1(3),'LineWidth',2.5);
        set(p1(4),'LineWidth',3.0);
        set(p1(5),'LineWidth',3.0);
        set(p1(6),'LineWidth',3.0);
        set(p1(7),'LineWidth',3.0);
        set(p1(8),'LineWidth',3.0);
        set(p1(9),'LineWidth',3.0);

        axis([-5 20 -5 10 0 0.04]);          % Toda la trayectoria
        axis ij
        xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
        ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
        zlabel('Altitud (eje z) en [mn]','FontSize',16)
        title('Trayectoria sea skimming del misil','FontSize',16)
        legend({'medicion','real','IMM','SMO','SMO2','Posición del buque
observador','Posición inicial del misil',...
                'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)
           %% 3D trayectoria completa

           figure(2);p2 = plot3(  yk_log(1,:),        yk_log(3,:),
yk_log(5,:),'-g',...
                                  xk_log(1,:),        xk_log(4,:),
xk_log(7,:),'-r',...
                                  xhIMM_log(1,:),     xhIMM_log(4,:),
xhIMM_log(7,:),'-k',...
                                  xhSMO1_log(1,:),     xhSMO1_log(4,:),
xhSMO1_log(7,:),'-b',...
                                  xhSMO2_log(1,:),    xhSMO2_log(4,:),
xhSMO2_log(7,:),'-m',...
                                  0,                  0,                  0, 'o',...
                                  xk_log(1,1),        xk_log(4,1),        xk_log(7,1),
'o',...
                                  xk_log(1,nt),       xk_log(4,nt),
xk_log(7,nt),'o',...
                                  0.9*xk_log(1,nt),   0.9*xk_log(4,nt),
0.9*xk_log(7,nt),'o');
        grid on
        set(p2(1),'LineWidth',2.0);
        set(p2(2),'LineWidth',3.0);
        set(p2(3),'LineWidth',2.5);
        set(p2(4),'LineWidth',3.0);
        set(p2(5),'LineWidth',3.0);
        set(p2(6),'LineWidth',3.0);
        set(p2(7),'LineWidth',3.0);
        set(p2(8),'LineWidth',3.0);
        set(p2(9),'LineWidth',3.0);
        axis([4 8 -2 5 0 0.016]);
        axis ij
        xlabel('Dirección Norte  (eje x) en [mn]','FontSize',16)
        ylabel('Dirección Este (eje y) en [mn]','FontSize',16)
        zlabel('Altitud (eje z) en [mn]','FontSize',16)
        title('Maniobra terminal del misil','FontSize',16)
        legend({'medicion','real','IMM','SMO','SMO2','Posición del buque
observador','Posición inicial del misil',...
                'Posición final del misil','Posición del buque
blanco'},'Location','northeast','NumColumns',1,'FontSize',14)

           %% Coordenada x
           % Posición x: Trayectoria completa
           figure(3);
           subplot(1,2,1);
```

```
            p3 = plot(bin_log(1,:),yk_log(1,:),'-r',bin_log(1,:),xk_log(1,:),'-
g',bin_log(1,:),xhIMM_log(1,:),'-k',bin_log(1,:),xhSMO1_log(1,:),'-
.b',bin_log(1,:),xhSMO2_log(1,:),'-m');
            title('Trayectoria completa del misil: posición x','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)

legend({'medición','real','IMM','SMO1','SMO2'},'Location','north','NumColumns',3,'FontSi
ze',18)
            set(p3(1),'LineWidth', 2.5);
            set(p3(2),'LineWidth', 1.8);
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            set(p3(5),'LineWidth', 1.6);
            grid on
            axis([1 nt min([xk_log(1,:) yk_log(1,:)]) max([xk_log(1,:) yk_log(1,:)])])

            % Posición x: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),yk_log(1,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(1,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhIMM_log(1,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhSMO1_log(1,lpx1:nt),'-.b',bin_log(1,:),xhSMO2_log(1,:),'-m');
            set(p4(1),'LineWidth', 2.5);
            set(p4(2),'LineWidth', 1.8);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            set(p4(5),'LineWidth', 1.6);
            title('Maniobra terminal del misil: posición x','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)
            grid on
            axis([lpx1 nt min([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)])
max([xk_log(1,lpx1:nt) yk_log(1,lpx1:nt)])])

            % Velocidad x: Trayectoria completa
            figure(4);
            subplot(1,2,1);
            p3 = plot(bin_log(1,:),yk_log(2,:),'-r',bin_log(1,:),xk_log(2,:),'-
g',bin_log(1,:),xhIMM_log(2,:),'-k',bin_log(1,:),xhSMO1_log(2,:),'-
.b',bin_log(1,:),xhSMO2_log(2,:),'-m');
            title('Trayectoria completa del misil: velocidad x','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('velocidad en [mach]','FontSize',22)

legend({'medición','real','IMM','SMO1','SMO2'},'Location','north','NumColumns',3,'FontSi
ze',18)
            set(p3(1),'LineWidth', 2.5);
            set(p3(2),'LineWidth', 1.8);
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            set(p3(5),'LineWidth', 1.6);
            grid on
            axis([1 nt min([xk_log(2,:) yk_log(2,:)]) max([xk_log(2,:) yk_log(2,:)])])

            % Velocidad x: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),yk_log(2,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(2,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhIMM_log(2,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhSMO1_log(2,lpx1:nt),'-.b',bin_log(1,:),xhSMO2_log(2,:),'-m');
            set(p4(1),'LineWidth', 2.5);
            set(p4(2),'LineWidth', 1.8);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            set(p4(5),'LineWidth', 1.6);
            title('Maniobra terminal del misil: velocidad x','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('velocidad en [mach]','FontSize',22)
            grid on
            axis([lpx1 nt min([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt)])
max([xk_log(2,lpx1:nt) yk_log(2,lpx1:nt)])])

            % Aceleración x: Trayectoria completa
            figure(5);
            subplot(1,2,1);
```

```
            p3 = plot(bin_log(1,:),xk_log(3,:),'-g',bin_log(1,:),xhIMM_log(3,:),'-
k',bin_log(1,:),xhSMO1_log(3,:),'-.b',bin_log(1,:),xhSMO2_log(3,:),'-m');
            title('Trayectoria completa del misil: aceleración x','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('aceleración en [g/s]','FontSize',22)

legend({'real','IMM','SMO1','SMO2'},'Location','north','NumColumns',3,'FontSize',18)
            set(p3(1),'LineWidth', 1.8);
            set(p3(2),'LineWidth', 1.6);
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            grid on
            axis([1 nt min(xk_log(3,:)) max(xk_log(3,:))])

            % Aceleración x: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),xk_log(3,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhIMM_log(3,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhSMO1_log(3,lpx1:nt),'-.b',bin_log(1,:),xhSMO2_log(3,:),'-m');
            set(p4(1),'LineWidth', 1.8);
            set(p4(2),'LineWidth', 1.6);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            title('Maniobra terminal del misil: aceleración x','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('aceleración en [g/s]','FontSize',22)
            grid on
            axis([lpx1 nt min(xk_log(3,lpx1:nt)) max(xk_log(3,lpx1:nt))])

            %% Coordenada y
            % Posición y: Trayectoria completa
            figure(6);
            subplot(1,2,1);
            p3 = plot(bin_log(1,:),yk_log(3,:),'-r',bin_log(1,:),xk_log(4,:),'-
g',bin_log(1,:),xhIMM_log(4,:),'-k',bin_log(1,:),xhSMO1_log(4,:),'-
.b',bin_log(1,:),xhSMO2_log(4,:),'-m');
            title('Trayectoria completa del misil: posición y','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)

legend({'medición','real','IMM','SMO1','SMO2'},'Location','north','NumColumns',3,'FontSi
ze',18)
            set(p3(1),'LineWidth', 2.5);
            set(p3(2),'LineWidth', 1.8);
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            set(p3(5),'LineWidth', 1.6);
            grid on
            axis([1 nt min([xk_log(4,:) yk_log(3,:)]) max([xk_log(4,:) yk_log(3,:)])])

            % Posición y: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),yk_log(3,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(4,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhIMM_log(4,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhSMO1_log(4,lpx1:nt),'-.b',bin_log(1,:),xhSMO2_log(4,:),'-m');
            set(p4(1),'LineWidth', 2.5);
            set(p4(2),'LineWidth', 1.8);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            set(p4(5),'LineWidth', 1.6);
            title('Maniobra terminal del misil: posición y','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)
            grid on
            axis([lpx1 nt min([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt)])
max([xk_log(4,lpx1:nt) yk_log(3,lpx1:nt)])])

            % Velocidad y: Trayectoria completa
            figure(7);
            subplot(1,2,1);
            p3 = plot(bin_log(1,:),yk_log(4,:),'-r',bin_log(1,:),xk_log(5,:),'-
g',bin_log(1,:),xhIMM_log(5,:),'-k',bin_log(1,:),xhSMO1_log(5,:),'-
.b',bin_log(1,:),xhSMO2_log(5,:),'-m');
            title('Trayectoria completa del misil: velocidad y','FontSize',22);
            xlabel('muestra','FontSize',22)
```

```
            ylabel('velocidad en [mach]','FontSize',22)

legend({'medición','real','IMM','SMO1','SMO2'},'Location','north','NumColumns',3,'FontSi
ze',18)
            set(p3(1),'LineWidth', 2.5);
            set(p3(2),'LineWidth', 1.8);
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            set(p3(5),'LineWidth', 1.6);
            grid on
            axis([1 nt min([xk_log(5,:) yk_log(4,:)]) max([xk_log(5,:) yk_log(4,:)])])

            % Velocidad y: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),yk_log(4,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(5,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhIMM_log(5,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhSMO1_log(5,lpx1:nt),'-.b',bin_log(1,:),xhSMO2_log(5,:),'-m');
            set(p4(1),'LineWidth', 2.5);
            set(p4(2),'LineWidth', 1.8);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            set(p4(5),'LineWidth', 1.6);
            title('Maniobra terminal del misil: velocidad y','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('velocidad en [mach]','FontSize',22)
            grid on
            axis([lpx1 nt min([xk_log(5,lpx1:nt) yk_log(4,lpx1:nt)])
max([xk_log(5,lpx1:nt) yk_log(4,lpx1:nt)])])

            % Aceleración y: Trayectoria completa
            figure(8);
            subplot(1,2,1);
            p3 = plot(bin_log(1,:),xk_log(6,:),'-g',bin_log(1,:),xhIMM_log(6,:),'-
k',bin_log(1,:),xhSMO1_log(6,:),'-.b',bin_log(1,:),xhSMO2_log(6,:),'-m');
            title('Trayectoria completa del misil: aceleración y','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('aceleración en [g/s]','FontSize',22)

legend({'real','IMM','SMO1','SMO2'},'Location','north','NumColumns',3,'FontSize',18)
            set(p3(1),'LineWidth', 1.8);
            set(p3(2),'LineWidth', 1.6);
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            grid on
            axis([1 nt min(xk_log(6,:)) max(xk_log(6,:))])

            % Aceleración y: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),xk_log(6,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhIMM_log(6,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhSMO1_log(6,lpx1:nt),'-.b',bin_log(1,:),xhSMO2_log(6,:),'-m');
            set(p4(1),'LineWidth', 1.8);
            set(p4(2),'LineWidth', 1.6);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            title('Maniobra terminal del misil: aceleración y','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('aceleración en [g/s]','FontSize',22)
            grid on
            axis([lpx1 nt min(xk_log(6,lpx1:nt)) max(xk_log(6,lpx1:nt))])
            %% Coordenada z
            % Posición : Trayectoria completa
            figure(9);
            subplot(1,2,1);
            p3 = plot(bin_log(1,:),yk_log(5,:),'-r',bin_log(1,:),xk_log(7,:),'-
g',bin_log(1,:),xhIMM_log(7,:),'-k',bin_log(1,:),xhSMO1_log(7,:),'-
.b',bin_log(1,:),xhSMO2_log(7,:),'-m');
            title('Trayectoria completa del misil: posición z','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)

legend({'medición','real','IMM','SMO1','SMO2'},'Location','north','NumColumns',3,'FontSi
ze',18)
            set(p3(1),'LineWidth', 2.5);
            set(p3(2),'LineWidth', 1.8);
```

```
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            set(p3(5),'LineWidth', 1.6);
            grid on
            axis([1 nt min([xk_log(7,:) yk_log(5,:)]) max([xk_log(7,:) yk_log(5,:)])])

            % Posición z: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),yk_log(5,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(7,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhIMM_log(7,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhSMO1_log(7,lpx1:nt),'-.b',bin_log(1,:),xhSMO2_log(7,:),'-m');
            set(p4(1),'LineWidth', 2.5);
            set(p4(2),'LineWidth', 1.8);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            set(p4(5),'LineWidth', 1.6);
            title('Maniobra terminal del misil: posición z','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('posición en [mn]','FontSize',22)
            grid on
            axis([lpx1 nt min([xk_log(7,lpx1:nt) yk_log(5,lpx1:nt)])
max([xk_log(7,lpx1:nt) yk_log(5,lpx1:nt)])])

            % Velocidad z: Trayectoria completa
            figure(10);
            subplot(1,2,1);
            p3 = plot(bin_log(1,:),yk_log(6,:),'-r',bin_log(1,:),xk_log(8,:),'-
g',bin_log(1,:),xhIMM_log(8,:),'-k',bin_log(1,:),xhSMO1_log(8,:),'-
.b',bin_log(1,:),xhSMO2_log(8,:),'-m');
            title('Trayectoria completa del misil: velocidad z','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('velocidad en [mach]','FontSize',22)

legend({'medición','real','IMM','SMO1','SMO2'},'Location','north','NumColumns',3,'FontSi
ze',18)
            set(p3(1),'LineWidth', 2.5);
            set(p3(2),'LineWidth', 1.8);
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            set(p3(5),'LineWidth', 1.6);
            grid on
            axis([1 nt min([xk_log(8,:) yk_log(6,:)]) max([xk_log(8,:) yk_log(6,:)])])

            % Velocidad z: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),yk_log(6,lpx1:nt),'-
r',bin_log(1,lpx1:nt),xk_log(8,lpx1:nt),'-g',bin_log(1,lpx1:nt),xhIMM_log(8,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhSMO1_log(8,lpx1:nt),'-.b',bin_log(1,:),xhSMO2_log(8,:),'-m');
            set(p4(1),'LineWidth', 2.5);
            set(p4(2),'LineWidth', 1.8);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            set(p4(5),'LineWidth', 1.6);
            title('Maniobra terminal del misil: velocidad z','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('velocidad en [mach]','FontSize',22)
            grid on
            axis([lpx1 nt min([xk_log(8,lpx1:nt) yk_log(6,lpx1:nt)])
max([xk_log(8,lpx1:nt) yk_log(6,lpx1:nt)])])

            % Aceleración z: Trayectoria completa
            figure(11);
            subplot(1,2,1);
            p3 = plot(bin_log(1,:),xk_log(9,:),'-g',bin_log(1,:),xhIMM_log(9,:),'-
k',bin_log(1,:),xhSMO1_log(9,:),'-.b',bin_log(1,:),xhSMO2_log(9,:),'-m');
            title('Trayectoria completa del misil: aceleración z','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('aceleración en [g/s]','FontSize',22)

legend({'real','IMM','SMO1','SMO2'},'Location','north','NumColumns',3,'FontSize',18)
            set(p3(1),'LineWidth', 1.8);
            set(p3(2),'LineWidth', 1.6);
            set(p3(3),'LineWidth', 1.6);
            set(p3(4),'LineWidth', 1.6);
            grid on
```

```
            axis([1 nt min(xk_log(9,:)) max(xk_log(9,:))])

            % Aceleración z: Maniobra terminal
            lpx1 = 4500;
            subplot(1,2,2);
            p4 = plot(bin_log(1,lpx1:nt),xk_log(9,lpx1:nt),'-
g',bin_log(1,lpx1:nt),xhIMM_log(9,lpx1:nt),'-
k',bin_log(1,lpx1:nt),xhSMO1_log(9,lpx1:nt),'-.b',bin_log(1,:),xhSMO2_log(9,:),'-m');
            set(p4(1),'LineWidth', 1.8);
            set(p4(2),'LineWidth', 1.6);
            set(p4(3),'LineWidth', 1.6);
            set(p4(4),'LineWidth', 1.6);
            title('Maniobra terminal del misil: aceleración z','FontSize',22);
            xlabel('muestra','FontSize',22)
            ylabel('aceleración en [g/s]','FontSize',22)
            grid on
            axis([lpx1 nt min(xk_log(9,lpx1:nt)) max(xk_log(9,lpx1:nt))])

            %% Plots señales de error cartesianas: Coordenada x
            figure(12);
            sgtitle('Errores cartesianos','FontSize',18)
            lpx1 = 4500;
            subplot(3,2,1);
            p9 = plot(bin_log(1,:),errorc_IMM_log(1,:),'-
k',bin_log(1,:),errorc_SMO1_log(1,:),'-.b',bin_log(1,:),errorc_SMO2_log(1,:),'-m');
            title('Trayectoria completa del misil: posición x','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [m]','FontSize',16)

legend({'IMM','SMO1','SMO2'},'Location','northeast','NumColumns',4,'FontSize',12)
            set(p9,'LineWidth',1.5);
            grid on
            axis([1 nt min([0.01*errorc_IMM_log(1,:) 0.01*errorc_SMO1_log(1,:)
0.01*errorc_SMO2_log(1,:)]) max([0.1*errorc_IMM_log(1,:) 0.1*errorc_SMO1_log(1,:)
0.1*errorc_SMO2_log(1,:)])])

            subplot(3,2,2);
            p10 = plot(bin_log(1,lpx1:nt),errorc_IMM_log(1,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorc_SMO1_log(1,lpx1:nt),'-
.b',bin_log(1,lpx1:nt),errorc_SMO2_log(1,lpx1:nt),'-m');
            title('Maniobra terminal del misil: posición x','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [m]','FontSize',16);
            set(p10,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorc_IMM_log(1,lpx1:nt) errorc_SMO1_log(1,lpx1:nt)
errorc_SMO2_log(1,lpx1:nt)]) max([errorc_IMM_log(1,lpx1:nt) errorc_SMO1_log(1,lpx1:nt)
errorc_SMO2_log(1,lpx1:nt)])])

            % Plots señales de error cartesianas: Coordenada y
            subplot(3,2,3);
            p11 = plot(bin_log(1,:),errorc_IMM_log(4,:),'-
k',bin_log(1,:),errorc_SMO1_log(4,:),'-.b',bin_log(1,:),errorc_SMO2_log(4,:),'-m');
            title('Trayectoria completa del misil: posición y','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [m]','FontSize',16)

legend({'IMM','SMO1','SMO2'},'Location','northeast','NumColumns',4,'FontSize',12)
            set(p11,'LineWidth',1.5);
            grid on
            axis([1 nt min([0.001*errorc_IMM_log(4,:) 0.1*errorc_SMO1_log(4,:)
0.1*errorc_SMO2_log(4,:)]) max([0.001*errorc_IMM_log(4,:) 0.01*errorc_SMO1_log(4,:)
0.01*errorc_SMO2_log(4,:)])])

            subplot(3,2,4);
            p12 = plot(bin_log(1,lpx1:nt),errorc_IMM_log(4,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorc_SMO1_log(4,lpx1:nt),'-
.b',bin_log(1,lpx1:nt),errorc_SMO2_log(4,lpx1:nt),'-m');
            title('Maniobra terminal del misil: posición y','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [m]','FontSize',16);
            set(p12,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorc_IMM_log(4,lpx1:nt) errorc_SMO1_log(4,lpx1:nt)
errorc_SMO2_log(4,lpx1:nt)]) max([errorc_IMM_log(4,lpx1:nt) errorc_SMO1_log(4,lpx1:nt)
errorc_SMO2_log(4,lpx1:nt)])])
```

```
            subplot(3,2,5);
            p11 = plot(bin_log(1,:),errorc_IMM_log(7,:),'-
k',bin_log(1,:),errorc_SMO1_log(7,:),'-.b',bin_log(1,:),errorc_SMO2_log(7,:),'-m');
            title('Trayectoria completa del misil: posición z','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [m]','FontSize',16)

legend({'IMM','SMO1','SMO2'},'Location','northeast','NumColumns',4,'FontSize',12)
            set(p11,'LineWidth',1.5);
            grid on
            axis([1 nt min([0.01*errorc_IMM_log(7,:) errorc_SMO1_log(7,:)
errorc_SMO2_log(7,:)]) max([0.01*errorc_IMM_log(7,:) errorc_SMO1_log(7,:)
errorc_SMO2_log(7,:)])])

            subplot(3,2,6);
            p12 = plot(bin_log(1,lpx1:nt),errorc_IMM_log(7,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorc_SMO1_log(7,lpx1:nt),'-
.b',bin_log(1,lpx1:nt),errorc_SMO2_log(7,lpx1:nt),'-m');
            title('Maniobra terminal del misil: posición z','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [m]','FontSize',16);
            set(p12,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorc_IMM_log(7,lpx1:nt) errorc_SMO1_log(7,lpx1:nt)
errorc_SMO1_log(7,lpx1:nt)]) max([errorc_IMM_log(7,lpx1:nt) errorc_SMO1_log(7,lpx1:nt)
errorc_SMO1_log(7,lpx1:nt)])])

            %% Plots señales de error polares: Distancia Ad
            figure(7);
            sgtitle('Errores polares','FontSize',18)
            maxAd = 37.5*ones(1,nt);
            minAd = -37.5*ones(1,nt);
            lpx1 = 4500;
            subplot(3,2,1);
            p9 = plot(bin_log(1,:),errorp_IMM_log(1,:),'-
k',bin_log(1,:),errorp_SMO1_log(1,:),'-.b',bin_log(1,:),errorp_SMO2_log(1,:),'-
m',bin_log(1,:),maxAd(1,:),'--r',bin_log(1,:),minAd(1,:),'--r');
            title('Trayectoria final del misil: Alcance','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [m]','FontSize',16)
            legend({'IMM','SMO1','SMO2','Límite +/-37.5
mts'},'Location','northeast','NumColumns',6,'FontSize',12)
            set(p9,'LineWidth',1.5);
            grid on
            axis([1 nt min([0.01*errorp_IMM_log(1,:) 0.01*errorp_SMO1_log(1,:)
0.01*errorp_SMO2_log(1,:) minAd(1,:)]) max([0.05*errorp_IMM_log(1,:)
0.05*errorp_SMO1_log(1,:) 0.05*errorp_SMO2_log(1,:) maxAd(1,:)])])

            subplot(3,2,2);
            p10 = plot(bin_log(1,lpx1:nt),errorp_IMM_log(1,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorp_SMO1_log(1,lpx1:nt),'-
.b',bin_log(1,lpx1:nt),errorp_SMO2_log(1,lpx1:nt),'-m',bin_log(1,:),maxAd(1,:),'--
r',bin_log(1,:),minAd(1,:),'--r');
            title('Maniobra terminal del misil: Alcance','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [m]','FontSize',16);
            set(p10,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorp_IMM_log(1,lpx1:nt) errorp_SMO1_log(1,lpx1:nt)
errorp_SMO2_log(1,lpx1:nt) minAd(1,:)]) max([errorp_IMM_log(1,lpx1:nt)
errorp_SMO1_log(1,lpx1:nt) errorp_SMO2_log(1,lpx1:nt) maxAd(1,:)])])

            % Plots señales de error polar: Bdn
            subplot(3,2,3);
            maxBdn = 2*ones(1,nt);
            minBdn = -2*ones(1,nt);
            p11 = plot(bin_log(1,:),errorp_IMM_log(2,:),'-
k',bin_log(1,:),errorp_SMO1_log(2,:),'-.b',bin_log(1,:),errorp_SMO2_log(2,:),'-
m',bin_log(1,:),maxBdn(1,:),'--r',bin_log(1,:),minBdn(1,:),'--r');
            title('Trayectoria completa del misil: Acimut','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [mrad]','FontSize',16);
            legend({'Límite +/-2 mrad'},'Location','north','NumColumns',1,'FontSize',12)
            set(p11,'LineWidth',1.5);
            grid on
```

```
            axis([1 nt min([0.5*errorp_IMM_log(2,:) 0.5*errorp_SMO1_log(2,:)
0.5*errorp_SMO2_log(2,:)]) max([0.05*errorp_IMM_log(2,:) 0.05*errorp_SMO1_log(2,:)
0.05*errorp_SMO2_log(2,:) maxBdn(1,:)])])

            subplot(3,2,4);
            p12 = plot(bin_log(1,lpx1:nt),errorp_IMM_log(2,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorp_SMO1_log(2,lpx1:nt),'-
.b',bin_log(1,lpx1:nt),errorp_SMO2_log(2,lpx1:nt),'-m',bin_log(1,:),maxBdn(1,:),'--
r',bin_log(1,:),minBdn(1,:),'--r');
            title('Maniobra terminal del misil: Acimut','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [mrad]','FontSize',16);
            set(p12,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorp_IMM_log(2,lpx1:nt) errorp_SMO1_log(2,lpx1:nt)
errorp_SMO2_log(2,lpx1:nt) minBdn(1,:)]) max([errorp_IMM_log(2,lpx1:nt)
errorp_SMO1_log(2,lpx1:nt) errorp_SMO2_log(2,lpx1:nt) maxBdn(1,:)])])

            subplot(3,2,5);
            maxEd = 2*ones(1,nt);
            minEd = -2*ones(1,nt);
            p11 = plot(bin_log(1,:),errorp_IMM_log(3,:),'-
k',bin_log(1,:),errorp_SMO1_log(3,:),'-.b',bin_log(1,:),errorp_SMO2_log(3,:),'-
m',bin_log(1,:),maxEd(1,:),'--r',bin_log(1,:),minEd(1,:),'--r');
            title('Trayectoria completa del misil: Elevación','FontSize',16);
            xlabel('muestra','FontSize',16)
            ylabel('Error en [mrad]','FontSize',16);
            legend({'Límite +2/-2
mrad'},'Location','north','NumColumns',1,'FontSize',12)
            set(p11,'LineWidth',1.5);
            grid on
            axis([1 nt min([errorp_IMM_log(3,:) errorp_SMO1_log(3,:)
errorp_SMO2_log(3,:) minEd(1,:)]) max([errorp_IMM_log(3,:) errorp_SMO1_log(3,:)
errorp_SMO2_log(3,:) maxEd(1,:)])])

            subplot(3,2,6);
            p12 = plot(bin_log(1,lpx1:nt),errorp_IMM_log(3,lpx1:nt),'-
k',bin_log(1,lpx1:nt),errorp_SMO1_log(3,lpx1:nt),'-
.b',bin_log(1,lpx1:nt),errorp_SMO2_log(3,lpx1:nt),'-m',bin_log(1,:),maxEd(1,:),'--
r',bin_log(1,:),minEd(1,:),'--r');
            title('Maniobra terminal del misil: Elevación','FontSize',16);
            xlabel('muestra','FontSize',16);
            ylabel('Error en [mrad]','FontSize',16);
            set(p12,'LineWidth',1.5);
            grid on
            axis([lpx1 nt min([errorp_IMM_log(3,lpx1:nt) errorp_SMO1_log(3,lpx1:nt)
errorp_SMO2_log(3,lpx1:nt) minEd(1,:)]) max([errorp_IMM_log(3,lpx1:nt)
errorp_SMO1_log(3,lpx1:nt) errorp_SMO2_log(3,lpx1:nt) maxEd(1,:)])])

save trayectorias.mat
```

Apéndice 5.2: Procesamiento intrapulso de velocidad doppler

```
function [batch_ykvx,batch_ykvy,batch_ykvz] =
batch_intra_pulso_vel(xk_log,yk_log,Xgm56t,Xgm78t,nro_intra_pulsos)
%=========================================================================
%
% Título            : batch_intra_pulso_vel.m
% Propósito         : Generar mediciones ruidosas de radar intra pulso en
%                     ceptas o batches
%
% Descripción       : Este programa permite la generación de mediciones
%                     ruidosas de velocidad doppler en coordenadas cartesianas
%                     en secuencias de 36 pulsos, intra pulso para el
%                     filtrado por medio de un filtro de mediana. Se asume
%                     que el PRF del radar de control de tiro es de 1800
%                     Hz, lo que permite obtener 36 pulsos o retornos del
%                     blanco en un período de 0.02 seg después de haberse
%                     procesado el pulso y[k-1](Este período es el
%                     período de muestreo del algoritmo de estimación de
%                     variables de estado, el cual solo procesa un pulso
%                     cada 0.02 segs). A la secuencia de nro_intra_pulsos+1 pulsos,
contando
%                     el pulso anterior y[k-1] se le
%                     aplica la operación de mediana y el resultado de esta
%                     operación ingresa al algoritmo de estimación
```

```
%
% Fecha de creación : 19/03/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
%-------------------------------------------------------------------------
% Constantes
%-------------------------------------------------------------------------
% número de puntos
nt = 5226;
% longitud de subsecuencias utilizadas para la generación del modelo mixto
% gaussiano
nt_set  = 52;
nt_set2 = 26;
a  = 0;
j  = 1;
%-------------------------------------------------------------------------
% Procesamiento en batches de Alcance (Ad), Acimut verdadero (Bdn) y
% elevación (Ed)
%-------------------------------------------------------------------------
for k = 2:nt
    % Es el pulso anterior y[k-1] procesado por el observador
    last_paint_vx     = xk_log(2,k-1)./0.00292;
    % Es el pulso actual y[k] procesado por el observador
    actual_paint_vx   = xk_log(2,k)./0.00292;
    % Generar un espacio lineal entre el pulso anterior y[k-1] y el pulso
    % actual y[k] a fin de generar una secuencia de la siguiente manera:
    % S = {y[k-1] p1 p2 p3 .... p35 y[k]}, donde p1,p2,... son los intra
    % pulsos no procesados por el algoritmo de estimación. Estos pulsos,
    % por el PRF del radar, son obtenidos pero nunca son procesados dado que
    % cada 0.02 segs el observador solo puede procesar 1 pulso.
    batch_vx        = linspace(last_paint_vx,actual_paint_vx,nro_intra_pulsos+1);

    last_paint_vy   = xk_log(5,k-1)./0.00292;
    actual_paint_vy = xk_log(5,k)./0.00292;
    batch_vy        = linspace(last_paint_vy,actual_paint_vy,nro_intra_pulsos+1);

    last_paint_vz    = xk_log(8,k-1)./0.00292;
    actual_paint_vz  = xk_log(8,k)./0.00292;
    batch_vz         = linspace(last_paint_vz,actual_paint_vz,nro_intra_pulsos+1);
%-------------------------------------------------------------------------
% Agregar ruido a los batches para simular los 36 pulsos de radar
%-------------------------------------------------------------------------
    if (xk_log(7,k)/0.00054 >= 5)

        % la distribución gaussiana mixta se encuentra dividida en 100
        % grupos de 52 y 1 grupo de 26. Si el índice de la muestra actual k
        % es menor que 52, agrega la distribución gaussiana mixta
        % correspondiente a las primeras 52 muestras. De lo contrario,
        % incrementa el índice j para utilizar la siguiente distribución
        % gaussiana mixta. Así, si los batches mantienen las propiedades de
        % ruido gaussiana mixto por secuencia.

        if (k <= nt_set*j)
            V_noise   = Xgm56t(1,a+1:a+length(batch_vx));                    % ruido
en alcance (mts)
            if k == nt_set*j
               if k == 5200
                  a = a + 26;
                  nt_set = nt_set2;
               else
                  a = a + 52;
               end
               j = j + 1;
            end
        end
        batch_ykvx(k,:) = batch_vx + V_noise;
        batch_ykvy(k,:) = batch_vy + V_noise;
        batch_ykvz(k,:) = batch_vz + V_noise;
    else
        if (k <= nt_set*j)
            V_noise   = Xgm78t(1,a+1:a+length(batch_vx));
            if k == nt_set*j
```

```
                if k == 5200
                    a = a + 26;
                    nt_set = nt_set2;
                else
                    a = a + 52;
                end
            end
            j = j + 1;
        end
        batch_ykvx(k,:) = batch_vx + V_noise;
        batch_ykvy(k,:) = batch_vy + V_noise;
        batch_ykvz(k,:) = batch_vz + V_noise;
    end
end
batch_ykvx(1,:) = yk_log(2,1).*ones(1,nro_intra_pulsos+1);
batch_ykvy(1,:) = yk_log(4,1).*ones(1,nro_intra_pulsos+1);
batch_ykvz(1,:) = yk_log(6,1).*ones(1,nro_intra_pulsos+1);
```

Apéndice 6

Programas de Matlab varios

```
function [Ad,Bdn,Ed] = c2p(xk)
%=========================================================================
%
% Título           : c2p.m
% Propósito        : Conversión cartesiana a polar
%
% Descripción      : Este programa permite la conversión de coordenadas
%                    cartesianas tridimensionales en coordenadas polares
%                    de distancia, acimut y elevación.
%
% Fecha de creación : 15/06/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
% Variables
% xk(1,1): posición x
% xk(4,1): posición y
% xk(7,1): posición z
Adx = xk(1,1);
Ady = xk(4,1);
Adz = xk(7,1);
% El buque traqueador se encuentra en el origen del plano cartesiano.
% Primer cuadrante: Norte es el eje x, Este es el eje y
%
if (Adx>0 && Ady>0)
    Bdn = 90-(180/pi)*atan2(Adx,Ady);
end
% Segundo cuadrante
if (Adx<0 && Ady>0)
    Bdn = 90+(180/pi)*atan2(Adx,Ady);
end
% Tercer cuadrante
if (Adx<0 && Ady<0)
    Bdn = 180+(180/pi)*atan2(Ady,Adx);
end
% Cuarto cuadrante
if (Adx>=0 && Ady<0)
    Bdn =(180/pi)*atan2(Ady,Adx);
end
Ed  = (180/pi)*atan(Adz/sqrt(Adx^2 + Ady^2));
Ad  = sqrt(Adx^2 + Ady^2 + Adz^2);
end
```

```
function[Adx, Ady, Adz]= p2c(Ad,Ed,Bdn)
Adh = Ad.*cosd(Ed);
Adx = Adh.*cosd(Bdn);
Ady = Adh.*sind(Bdn);
Adz = Ad.*sind(Ed);
end

function[yk] = yk_noise(Ad,Bdn,Ed,xk,Xgm12t,Xgm34t,Xgm56t,Xgm78t,gl,k)
%=========================================================================
%
% Título            : yk_noise.m
% Propósito         : Generar mediciones ruidosas de radar
%
% Descripción       : Este programa permite la generación de mediciones
%                     ruidosas de posición y velocidad en coordenadas
%                     cartesianas a partir de la conversión del modelo de
%                     medición no lineal (coordenadas esféricas a
%                     cartesianas).
%
% Fecha de creación : 15/06/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
%
% Variables
%
% Xgm12(k,1),Xgm34(k,1): Vector de ruido glint en distancia (mts)
% Xgm12(k,2),Xgm34(k,2): Vector de ruido glint en acimut (mrad)
% Xgm12(k,3),Xgm34(k,3): Vector de ruido glint en elevación (mrad)
% Xgm56(k,3),Xgm78(k,3): Vector de ruido doppler (mts/s)
% gl: constante que activa el ruido
%
% Vector de mediciones yk: 6x1
% Al ejecutar este algoritmo es como hacer la operación yk = ck*xk + ruido
% donde ck es la matriz de medición, xk es el vector de estados reales.
%
% Conversiones
% 1 mrad = 0.0572958
%
% Saturación ascendente
% A medida que el misil se acerca al blanco el ruido glint aumenta
%
%x = 1:nt;
%y = 0:0.5/(length(x)-1):0.5;
%
% Cálculo de coordenadas cartesianas con ruido de medición: posición
%
if (xk(7,1) >= 5)
    Ad_noise  = gl*Xgm12t(3,k);                 % ruido en alcance (mts)
    Bdn_noise = gl*Xgm12t(2,k)*0.0572958;       % ruido en acimut (grados)
    Ed_noise  = gl*Xgm12t(1,k)*0.0572958;       % ruido en elevación (grados)
    Adh = (Ad + Ad_noise).*cosd(Ed + Ed_noise);
    yk(1,1) = Adh.*cosd(Bdn + Bdn_noise);
    yk(3,1) = Adh.*sind(Bdn + Bdn_noise);
    yk(5,1) = (Ad + 0.05*Ad_noise).*sind(Ed + 0.05*Ed_noise);
else
    Ad_noise  = gl*Xgm34t(3,k);                 % ruido en alcance (mts)
    Bdn_noise = gl*Xgm34t(2,k)*0.0572958;       % ruido en acimut (grados)
    Ed_noise  = gl*Xgm34t(1,k)*0.0572958;       % ruido en elevación (grados)
    Adh = (Ad + Ad_noise).*cosd(Ed + Ed_noise);
    yk(1,1) = Adh.*cosd(Bdn + Bdn_noise);
    yk(3,1) = Adh.*sind(Bdn + Bdn_noise);
    yk(5,1) = (Ad + 0.05*Ad_noise).*sind(Ed + 0.05*Ed_noise);
end
%
% Cálculo de coordenadas cartesianas con ruido de medición: velocidad
%
if (xk(7,1) >= 5)
    V_noise  = gl*Xgm56t(1,k);                % ruido doppler (mts/seg)
    yk(2,1)  = xk(2,1) + V_noise;
    yk(4,1)  = xk(5,1) + V_noise;
    yk(6,1)  = xk(8,1) + V_noise;
else
```

```
    V_noise  = gl*Xgm78t(1,k);                % ruido doppler (mts/seg)
    yk(2,1)  = xk(2,1) + V_noise;
    yk(4,1)  = xk(5,1) + V_noise;
    yk(6,1)  = xk(8,1) + V_noise;
end
end

function [Ad,Bdn,Ed] = c2p_filt(xk)
%=========================================================================
%
% Título            : c2p_filt.m
% Propósito         : Conversión cartesiana a polar
%
% Descripción       : Este programa permite la conversión de coordenadas
%                     cartesianas tridimensionales en coordenadas polares
%                     de distancia, acimut y elevación.
%
% Fecha de creación : 15/06/2020
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%
%=========================================================================
% Variables
% xk(1,1): posición x
% xk(3,1): posición y
% xk(5,1): posición z
Adx = xk(1,1);
Ady = xk(3,1);
Adz = xk(5,1);
% El buque traqueador se encuentra en el origen del plano cartesiano.
% Primer cuadrante: Norte es el eje x, Este es el eje y
%
if (Adx>0 && Ady>0)
    Bdn = 90-(180/pi)*atan2(Adx,Ady);
end
% Segundo cuadrante
if (Adx<0 && Ady>0)
    Bdn = 90+(180/pi)*atan2(Adx,Ady);
end
% Tercer cuadrante
if (Adx<0 && Ady<0)
    Bdn = 180+(180/pi)*atan2(Ady,Adx);
end
% Cuarto cuadrante
if (Adx>=0 && Ady<0)
    Bdn =(180/pi)*atan2(Ady,Adx);
end
Ed  = (180/pi)*atan(Adz/sqrt(Adx^2 + Ady^2));
Ad  = sqrt(Adx^2 + Ady^2 + Adz^2);
end

function likelihood = contaccMeasurementLikelihoodFcn(predictedParticles,measurement)
%=========================================================================
%
% Título            : contaccMeasurementLikelihoodFcn.m
% Propósito         : Modelo de probabilidad de medición del filtro de
%                     partículas para un blanco aéreo a aceleración
%                     constante.
%
% Descripción       : Función de probabilidad de las mediciones para el
%                     filtro de partículas. Para el desarrollo de esta
%                     función se tomó como referencia
%                     vdpMeasurementLikelihoodFcn.m de MathWorks.
% Entradas          : predictedParticles - Matriz de dimensión NumberOfStates-by-
NumberOfParticles que posee las partículas predichas
% Salidas           : likelihood - Vector de "NumberOfParticles" elementos que su
elemento número "n" es la probabilidad de la particula número "n"
% Fecha de creación : 19/03/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
```

```
%
%========================================================================

numberOfMeasurements = 3; % Número de variables de estado medidas
%------------------------------------------------------------------------
% Validar las mediciones
%------------------------------------------------------------------------
validateattributes(measurement, {'double'}, {'vector', 'numel', numberOfMeasurements},
...
    'contaccMeasurementLikelihoodFcn', 'measurement');
%------------------------------------------------------------------------
% Ruido de las mediciones
%------------------------------------------------------------------------
% varianza = 100
% Matriz de covarianza del ruido
measurementNoise = 100 * eye(numberOfMeasurements);
%------------------------------------------------------------------------
% Obtener predicción del estado de todas las partículas
%------------------------------------------------------------------------
predictedMeasurement(1,:) = predictedParticles(1,:);
predictedMeasurement(2,:) = predictedParticles(4,:);
predictedMeasurement(3,:) = predictedParticles(7,:);
%------------------------------------------------------------------------
% Calculo de errores entre predicción y medición
%------------------------------------------------------------------------
measurementError = bsxfun(@minus, predictedMeasurement, measurement(:));
% Usar el ruido de medición y efectuar el producto punto
measurementErrorProd = dot(measurementError, measurementNoise \ measurementError, 1);
%------------------------------------------------------------------------
% Convertir norm del error en una medición probabilística
%------------------------------------------------------------------------
% Evaluar la PDF de la distribución normal multivariable. Una medición del error
% igual a 0 brinda la mayor probabilidad posible.
likelihood = 1/sqrt((2*pi).^numberOfMeasurements * det(measurementNoise)) * exp(-0.5 *
measurementErrorProd);
end

function particles = contaccParticleFilterStateFcn(particles,uk,pk,Fk,Gk,Dk,k)
%========================================================================
%
% Título            : contaccParticleFilterStateFcn.m
% Propósito         : Matriz de transición de estados de un blanco aéreo
%                     a aceleración constante.
%
% Descripción       : Efectúa la transición de estados de las partículas.
%                     Para el desarrollo de esta función se tomó como referencia
%                     vdpParticleFilterStateFcn.m de MathWorks.
% Entradas          : Particles - Matriz de dimensión NumberOfStates-by-
NumberOfParticles que posee las partículas predichas
% Salidas           : PredictedParticles - Matriz de partículas predichas para la
siguiente iteración
% Fecha de creación : 19/03/2021
% Autor             : Italo Aranda Cetraro (a20194480@pucp.edu.pe)
% Institución       : Pontificia Universidad Católica del Perú
% Programa          : Maestría en Ingeniería de Control y Automatización
%
% Derechos de autor : Todos los derechos reservados.
%                     Copyright 2017 The MathWorks, Inc.
%========================================================================

[numberOfStates, numberOfParticles] = size(particles);
%------------------------------------------------------------------------
% Propagación de estados
%------------------------------------------------------------------------
for kk=1:numberOfParticles
    particles(:,kk) = contaccStateFcnContinuous(particles(:,kk),uk,pk,Fk,Gk,Dk,k);
end

% Agregar ruido de proceso para las partículas
processNoise = 0.025*eye(numberOfStates);
particles = particles + processNoise * randn(size(particles));
end
%------------------------------------------------------------------------
% Modelo del blanco
%------------------------------------------------------------------------
function dxdt = contaccStateFcnContinuous(x,u,p,Fk,Gk,Dk,k)
```

```
dxdt = Fk*[x(1);x(2);x(3);x(4);x(5);x(6);x(7);x(8);x(9)] + Gk*[u(1,k);u(2,k);u(3,k)] +
Dk*[p(1,k);p(2,k);p(3,k)];
end
```