UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Repositorio Digital USM **https://repositorio.usm.cl**

Departamento de Arquitectura Arq_paso

2022-04

A NOVEL FILTERING METHOD FOR HAMMERSTEIN-WIENER STATE-SPACE MODELS

ANGEL LEONEL CEDEÑO NIETO, ANGEL LEONEL

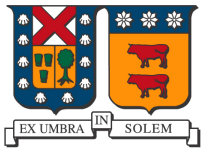
https://hdl.handle.net/11673/53465

Repositorio Digital USM, UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

Valparaíso, Chile



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

A novel Filtering Method for Hammerstein-Wiener State-Space Models

Tesis de Grado presentada por

Angel Leonel Cedeño Nieto

como requisito parcial para optar al título de

Magíster en Ciencias de la Ingeniería Electrónica

Supervisor

Juan Carlos Agüero Vásquez

Co-Supervisor

Rodrigo Carvajal Guerra

Abril, 2022

Título de Tesis:

A novel Filtering Method for Hammerstein-Wiener State-Space Models

Autor:

Angel Leonel Cedeño Nieto

Trabajo de Tesis, presentado en cumplimiento parcial de los requisitos para el grado de Magíster en Ciencias de la Ingeniería Electrónica de la Universidad Técnica Federico Santa María.

Supervisor
Dr. Juan C. Agüero _____

Co-Supervisor
Dr. Rodrigo Carvajal _____

Revisor Interno y Presidente de la Comisión
Dr. Francisco Vargas _____

Revisor Externo
Dr. Arturo Padilla _____

Al Todopoderoso

A mis padres,
a mi esposa,
y amigos

Agradecimientos

En está lineas quiero expresar mi gratitud a Dios quien me ha guiado en cada decisión y me ha dado fuerzas para continuar avante en esta etapa de mi vida. A mi familia, quienes son una fuente de inspiración y ejemplo a seguir, especialmente a mi madre, quien con sus consejos y oraciones me ha guiado por el camino correcto. A mi esposa, quien me ha acompañado en las horas más difíciles llenándome de su amor y ternura, un oasis refrescante en medio de un desierto. A todos mis amigos y compañeros, que más que amigos son como hermanos, y que se han convertido en una parte muy importante de mi vida. Gracias por su ayuda incondicional y por todos los momentos compartidos.

Agradezco a los profesores Juan Carlos Agüero y Rodrigo Carvajal, quienes me han brindado su experiencia y me han guiado en medio de este inmenso mundo de la ciencia, tecnología e investigación. Les agradezco de todo corazón por el tiempo que me han brindado, especialmente por tantas horas de conversaciones y descubrimientos en que nos vimos envueltos hasta hallar la forma de abordar los problemas que nos planteamos. Muchas gracias por preocuparse y estar pendientes de nosotros. Para mí más que profesores guía los considero mis amigos, a quienes respeto y admiro mucho. Muchas gracias por tanto.

Finalmente, quiero agradecer a la Universidad Técnica Federico Santa María, por abrirme sus puertas y darme la oportunidad de cursar mi estudios. Además, extiendo mi agradecimiento a la Agencia Nacional de Investigación y Desarrollo (ANID) Programa de becas/Doctorado Nacional/2020-21202410, al Centro Avanzado de Ingeniería Eléctrica y Electrónica (AC3E) proyecto Basal ANID FB0008, al proyecto Fondecyt ANID 1211630 del profesor Juan C. Agüero, a la Dirección de Postgrado y Programas (DPP) y a la Oficina de Asuntos Internacionales (OAI).

Resumen

La estimación de estados en sistemas no lineales ha ganado una atención significativa en los últimos años debido al creciente número de aplicaciones que involucran no linealidades tales como (i) cuantización introducida por sensores de bajo costo y baja resolución, (ii) saturación debido a la capacidad limitada de los actuadores y a restricciones físicas y/o mecánicas inherentes al sistema, o (iii) no linealidades comunes como: histéresis, contragolpe y zona muerta. En estos escenarios, obtener estimaciones de los estados de un sistema requiere estrategias y técnicas adecuadas para tratar tales no linealidades.

Por esta razón, el principal objetivo de esta Tesis es desarrollar un nuevo método de filtraje y estimación de estados para una clase de modelos no lineales denominados Hammerstein-Wiener, donde no linealidades estáticas están presentes antes y después de un sistema dinámico lineal. La solución que se presenta en esta Tesis se basa en un modelo explícito para la función de probabilidad de la salida no lineal condicionada al estado del sistema. Este modelo probabilístico se obtiene aproximando una ecuación integral mediante cuadratura gaussiana, lo que produce una estructura denominada modelo de suma de gaussianas. Este modelo permite desarrollar un algoritmo que es capaz de obtener, con alta precisión, la distribución de filtraje y la estimación de los estados en un modelo Hammerstein-Wiener, considerando una variedad de funciones no lineales típicas.

Se analiza el desempeño del algoritmo de filtraje propuesto en términos de la precisión de las estimaciones y del costo computacional asociado. Para esto se implementan simulaciones numéricas y se compara con algunas técnicas clásicas existentes en la literatura.

Abstract

The state estimation of nonlinear dynamical systems has gained significant attention due to the growing number of applications that involve nonlinearities such as (i) quantization introduced by low-cost and low-resolution sensors, (ii) saturation due to limited actuator's capacity, and inherent physical or mechanical constraints, or (iii) common nonlinearities: Hysteresis, Backlash, and Dead-zone. In these scenarios, obtaining state estimates require advanced strategies and techniques that can deal with these nonlinearities.

The main goal of this thesis is to develop a new method of filtering and state estimation for a class of nonlinear models in state-space called the Hammerstein-Wiener model, in which a linear dynamic system states between two static nonlinearities. The filtering approach developed in this work is carried out by introducing an explicit model for the probability function of the nonlinear output conditioned to the system state. This probabilistic model is obtained by approximating an integral equation using Gaussian quadrature, which produces a structure called Gaussian sum model. Thus, a Gaussian sum filtering algorithm can be utilized to obtain the filtering distribution and state estimation for a variety of nonlinear functions.

To analyze the performance of the proposed filtering algorithm in terms of the accuracy of the estimation and computational cost, numerical simulations are carried out and the results are compared with classical techniques.

Índice general

1. Introducción **1**
- 1.1. Definición del Problema . . . 5
- 1.2. Consideraciones Generales . . . 6
  - 1.2.1. No linealidad en la entrada . . . 6
  - 1.2.2. No linealidad en la salida . . . 7
- 1.3. Principales Contribuciones de la Tesis . . . 8
- 1.4. Publicaciones asociadas . . . 8
- 1.5. Estructura del Documento . . . 9

2. Conceptos y definiciones **10**
- 2.1. Estimación de estados . . . 10
  - 2.1.1. Media condicional . . . 11
- 2.2. Modelo de suma de gaussianas . . . 11
- 2.3. Cuadratura gaussiana . . . 12
  - 2.3.1. Cuadratura de Gauss-Legendre . . . 13
  - 2.3.2. Cuadratura de Gauss-Hermite . . . 15

3. El problema de filtraje bayesiano **16**
- 3.1. Introducción . . . 16
- 3.2. Filtraje bayesiano . . . 17
- 3.3. Modelo de la función de probabilidad $p(y_t|x_t)$ . . . 19
  - 3.3.1. Función estrictamente monótona a trozos . . . 19
  - 3.3.2. Función Afín . . . 22
  - 3.3.3. Función cuantización binaria . . . 23
  - 3.3.4. Función saturación . . . 25
  - 3.3.5. Función zona muerta . . . 27

4. Algoritmo de filtraje para sistemas Hammerstein-Wiener **30**
- 4.1. Introducción . . . 30
- 4.2. Filtro suma de gaussianas (GSF) . . . 30
  - 4.2.1. Reducción de gaussianas . . . 33
  - 4.2.2. Estimador de estados y error de estimación . . . 36

5. Ejemplos Numéricos **37**
5.1. Ejemplo 1: función cuadrática . . . 38
5.2. Ejemplo 2: función definida a trozos . . . 40
5.3. Ejemplo 3: función cúbica . . . 42
5.4. Ejemplo 4: función cuantizador binario . . . 44
5.5. Ejemplo 5: función saturador . . . 46
5.6. Ejemplo 6: función zona muerta . . . 48

6. Conclusiones **51**
6.1. Trabajo futuro . . . 52

A. Apéndices **53**
A.1. Algoritmos de filtraje en la literatura . . . 53
A.1.1. Sistema extendido . . . 53
A.1.2. Filtro de Kalman estándar . . . 54
A.1.3. Filtro de Kalman extendido . . . 54
A.1.4. Filtro de Kalman cuantizado . . . 55
A.1.5. Filtro de Kalman en cuadratura . . . 56
A.1.6. Filtro de partículas . . . 57
A.2. Lemas y Teoremas usados en está Tesis . . . 58
A.3. Equivalencia con el filtro de Kalman cuando $g(\cdot)$ es una función afín . . . 59
A.4. Problemas numéricos en el cálculo de los pesos en el GSF. . . . 61
A.5. Código de las simulaciones . . . 61

Índice de figuras

1.1. Representación de un proceso real. 2
1.2. Modelos no lineales BONL. 3
1.3. Diagrama de un sistema Hammerstein-Wiener en espacio de estados. 5
1.4. Resumen de las funciones no lineales consideradas. 7

2.1. Modelo de una suma de gaussianas. 12

3.1. Representación de las etapas del filtraje bayesiano. 18
3.2. Gráfica de una función definida a trozos. 20
3.3. Gráfica de una función afín. 22
3.4. Gráfica de la función cuantización binaria 23
3.5. Gráfica de la función saturación. 25
3.6. Gráfica de la función zona muerta. 28

4.1. Ejemplo del crecimiento exponencial del número de componentes gaussianas en la etapa de corrección. 33

5.1. Ejemplo 1. Gráfica de la salida lineal r_t y salida no lineal y_t. 38
5.2. Ejemplo 1. PDF filtrada $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 10 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF. 39
5.3. Ejemplo 1. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo. El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo. 39
5.4. Ejemplo 1. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t. 40
5.5. Ejemplo 2. Gráfica de la salida lineal r_t y salida no lineal y_t. 40
5.6. Ejemplo 2. PDF filtrada (marginal) $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 10 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF. 41

5.7. Ejemplo 2. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo (Gráfica del primer estado). El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo. 41
5.8. Ejemplo 2. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t. 42
5.9. Ejemplo 3. Gráfica de la salida lineal r_t y salida no lineal y_t. 42
5.10. Ejemplo 3. PDF filtrada (marginal) $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 10 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF. 43
5.11. Ejemplo 3. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo (Gráfica del primer estado). El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo. 43
5.12. Ejemplo 3. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t. 44
5.13. Ejemplo 4. Gráfica de la salida lineal r_t y salida no lineal y_t. 44
5.14. Ejemplo 4. PDF filtrada $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 100 componentes gaussianas en el filtro GSF y 300 partículas en el filtro PF. . . 45
5.15. Ejemplo 4. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo. El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo. 45
5.16. Ejemplo 4. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t. 46
5.17. Ejemplo 5. Gráfica de la salida lineal r_t y salida no lineal y_t. 46
5.18. Ejemplo 5. PDF filtrada (marginal) $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 50 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF. 47
5.19. Ejemplo 5. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo (Gráfica del primer estado). El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo. 47
5.20. Ejemplo 5. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t. 48
5.21. Ejemplo 6. Gráfica de la salida lineal r_t y salida no lineal y_t. 48

5.22. Ejemplo 6. PDF filtrada (marginal) $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 10 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF. 49
5.23. Ejemplo 6. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo (Gráfica del primer estado). El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo. 49
5.24. Ejemplo 6. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t. 50

A.1. Mapeo de los índices del producto de dos sumas. 59

1

Introducción

La idea de filtraje ha estado presente en muchas actividades del ser humano a lo largo de la historia. En un principio, la idea de filtrar se refería a librar de impurezas algunos líquidos como el agua, pasándolos a través de un mecanismo o proceso para eliminar sustancias o "elementos" no deseados. Con el desarrollo de la ciencia, este concepto fue evolucionando a uno mucho más amplio que abarca diversas áreas del conocimiento. En particular, en el área de la tecnología electrónica, este concepto se refiere a circuitos y/o algoritmos para filtrar señales de posibles ruidos y de otras señales no deseadas (Anderson and Moore, 1979) o para atenuar y/o amplificar señales en un cierto rango de frecuencias (Oppenheim, 1969). Por otro lado, con la llegada del control moderno, se introdujo el concepto de *espacio de estados* de un sistema dinámico (Kalman, 1960), así como un modelo matemático probabilístico que permite describir el comportamiento de dicho sistema con base en la evolución temporal de sus estados. En este contexto, el filtraje se refiere a caracterizar los estados (que son variables aleatorias) de un sistema dinámico con base en mediciones ruidosas de la salida (Bucy and Joseph, 1968).

Los métodos y algoritmos de filtraje para sistemas en espacio de estados juegan un papel crítico en muchas áreas del conocimiento. Son usados en un sinnúmero de aplicaciones en áreas de diseño de sistemas de control (Anderson and Moore, 2007; Goodwin et al., 2001; Leong et al., 2018), identificación de parámetros (Agüero et al., 2012; Gibson and Ninness, 2005; Kaltiokallio et al., 2021; Yuz et al., 2011), diagnóstico de fallas (Bonvini et al., 2014; Huang et al., 2021; Jeong et al., 2019; Li et al., 2011; Nemati et al., 2019; Zhang et al., 2021), sistemas de potencia (Ji et al., 2021; Zhao et al., 2021), análisis de datos financieros y económicos (Gençay et al., 2001), redes de sensores (Curiac, 2016; Han et al., 2020; Ju et al., 2019; Liu et al., 2019; Msechu and Giannakis, 2012; Noshad et al., 2019; Rana and Li, 2015; Schizas et al., 2008), pronóstico (Ahwiadi and Wang, 2020; Corbetta et al., 2018), sistemas cyber-físicos (Ding et al., 2021), asimilación de datos hídricos y geofísicos (Cosme et al., 2012; McLaughlin, 2002), seguimiento marítimo (Stone et al., 2013), sistemas de navegación (Chiang et al., 2012), transporte y gestión del tráfico en carreteras (Ahmed et al., 2019; Rostami Shahrbabaki et al., 2018; Schreiter et al., 2010), entre otras.

Una de las formas más usadas para desarrollar algoritmos de filtraje es el enfoque bayesiano, con el cual se obtienen ecuaciones de filtraje generales basadas en la representación probabilística del sistema (Kamen and Su, 1999). En este enfoque, una de las mayores dificultades es tratar con algunas integrales que son difíciles o casi imposibles de calcular, ya que el cómputo de la función a posteriori del estado crece en complejidad a medida que nueva información de la salida está disponible. Una alternativa es usar la propiedad

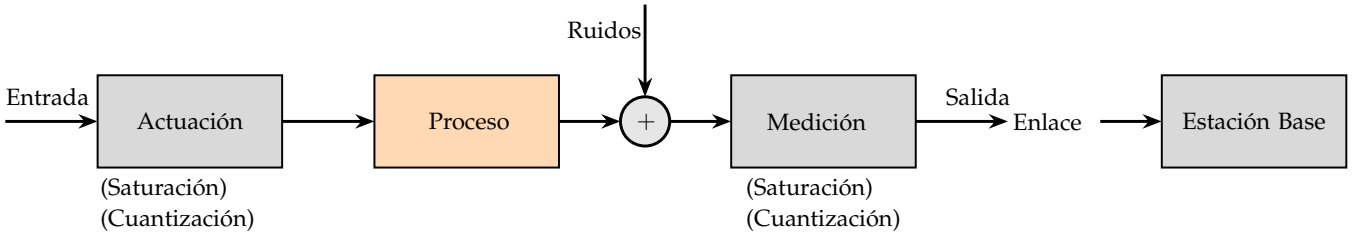
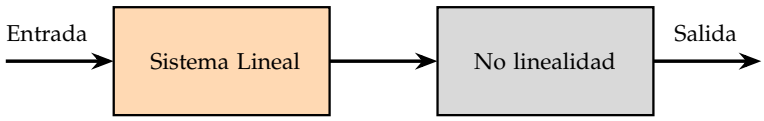


Figura 1.1: Representación de un proceso real.

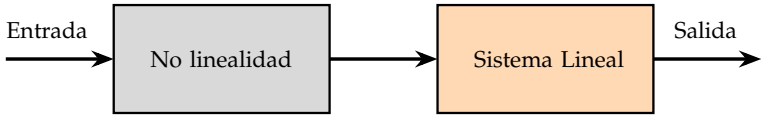
markoviana del sistema descrito en espacio de estados para obtener estimadores recursivos que permiten incorporar las nuevas mediciones sin tener que usar toda la información pasada de la salida y de los estados. En este contexto, para sistemas lineales en espacio de estados y ruidos con distribución gaussiana, el estimador *óptimo* de los estados del sistema, que minimiza el error cuadrático medio, es el filtro de Kalman (Kalman, 1960). Sin embargo para el caso general, es decir, sistemas no lineales y ruidos no gaussianos, no es posible obtener un estimador óptimo de manera cerrada ya que, en estas condiciones, algunas integrales que requieren ser resueltas en cada iteración del algoritmo de filtraje beyesiano son computacionalmente intratables. Es por esto que existen muchas soluciones sub-óptimas que permiten resolver el problema de filtraje de manera aproximada para ciertos tipos de sistemas (Särkkä, 2013).

En general, los sistemas físicos presentan dinámicas subyacentes no lineales, las cuales son representadas por modelos matemáticos obtenidos de las leyes físicas que los gobiernan (Cessenat, 2018). Además, en muchas aplicaciones se incorporan, por diseño, no linealidades a los sistemas para mejorar la eficiencia y seguridad de los procesos, y en algunos casos las no linealidades aparecen por la naturaleza de los sensores, actuadores y otros dispositivos usados en la práctica. Por ejemplo, en algunos sistemas de control se introduce saturación en la señal de actuación para salvaguardar la vida útil de actuadores y demás dispositivos que conforman el sistema, para garantizar la seguridad de las personas que manipulan o supervisan dicho proceso y para tratar con la limitada capacidad en amplitud y velocidad de los actuadores. (Hu and Lin, 2001; Kapila and Grigoriadis, 2002; Li and Lin, 2018; Yang et al., 2019). Por otra parte, para almacenar y/o enviar información a través de un canal de comunicaciones se cuantiza y codifica dicha información con el fin de reducir el consumo de recursos de almacenamiento y/o ancho de banda. La cuantización es un tipo de no linealidad que produce pérdida de información respecto a la señal original. Un tipo común de cuantización es asignar valores binarios a la señal que se desea transmitir (por ejemplo 1 y 0), en función de si es mayor o no a cierto umbral (Gersho and Gray, 2012; Wang et al., 2010). Este tipo de sistemas con dinámicas no lineales se representan con una estructura similar a la mostrada en la Figura 1.1 ya que describe muchos procesos físicos y permite un mejor entendimiento del sistema, ver por ejemplo (Chang and Liu, 2020; Curiac, 2016; Jeong et al., 2019; Li et al., 2011; Rana and Li, 2015). En esta figura se muestra un esquema de un sistema real, con una entrada que típicamente proviene de un sistema de control local o en red, actuadores que ejecutan la señal de control en el proceso (generalmente no lineal), un sensor que mide con errores y/o con poca resolución los valores de las variables de salida, un enlace de comunicaciones y una estación base donde se realizan tareas de control, identificación de parámetros, supervisión, entre otras.

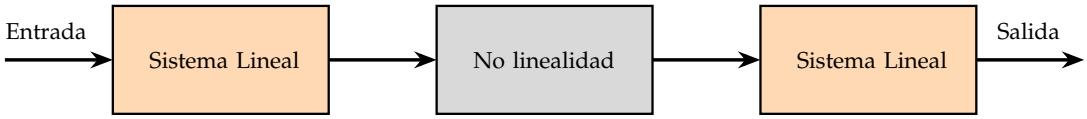
En general, un modelo matemático que tenga en cuenta todas las no linealidades que puede tener un proceso real es muy poco práctico. Por simplicidad, para algunas tareas de control, detección de fallas, supervisión e identificación de parámetros, se usan versiones lineales de estos modelos que son mucho más fáciles de tratar y estudiar, y para los cuales existe una teoría muy madura en lo que concierne a control, estimación de estados, algoritmos de filtraje, identificación de parámetros, entre otras (Gruyitch, 2019; Hendricks et al., 2008; Williams II and Lawrence, 2007). Sin embargo, a pesar de las dificultades y falta de generalidad que tiene el análisis de sistemas no lineales, hay un creciente desarrollo en técnicas de control, filtraje e identificación, como se puede observar en la literatura actual (Guo and Han, 2018; Schön et al.,



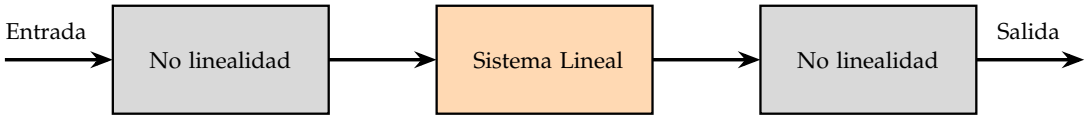
(a) Modelo Wiener



(b) Modelo Hammerstein



(c) Modelo Wiener-Hammerstein



(d) Modelo Hammerstein-Wiener

Figura 1.2: Modelos no lineales BONL.

2011; Simon, 2006; Wills et al., 2013). Una de las formas de abordar este análisis, es mediante la elección de una representación apropiada de estos sistemas no lineales. Por ejemplo, modelos NARMAX (por sus siglas en inglés Nonlinear Autoregressive Moving Average model with Exogenous inputs) (Billings, 2013), BONL (por sus siglas en inglés Block-Oriented Nonlinear model) (Giri and Bai, 2010; Schoukens and Tiels, 2017), neuro-difusos (Brown and Harris, 1994), Volterra series (Ogunfunmi, 2007), no paramétricos, entre otros.

Los modelos orientados a bloques BONL, son una combinación de sub-sistemas lineales e invariantes en el tiempo (LTI) y sub-sistemas estáticos no lineales (Grimble and Majecki, 2020; Schön et al., 2011; Schoukens and Tiels, 2017; Slotine and Li, 1991). En la Figura 1.2 se muestran algunos ejemplos de sistemas BONL muy usados en la literatura, llamados modelos Hammerstein y modelos Wiener, así como algunas combinaciones de ellos. Estos modelos ofrecen algunas ventajas respecto a otras representaciones de sistemas no lineales, tales como bajo costo computacional en tareas de identificación y facilidad de uso en sistemas de control (Zhu, 2002). Particularmente en la Figura 1.2 (d), se presenta el modelo Hammerstein–Wiener que consiste en un bloque no lineal y estático seguido de un bloque dinámico lineal, seguido de otro bloque no lineal y estático. Los modelos Hammerstein-Wiener han sido usados en muchas aplicaciones, como por ejemplo, para mejorar el modelo de transistores de electrón único (Pês et al., 2019), modelar procesos industriales (Xu et al., 2019), modelar procesos insulina-glucosa en pacientes con diabetes mellitus insulino-dependientes (Bhattacharjee et al., 2010), predecir variaciones de temperatura en pacas de ensilaje apiladas en la producción ganadera (Nadimi et al., 2012), por mencionar algunas.

Existe una gran literatura concerniente al control e identificación de sistemas Hammerstein-Wiener (Giri and Bai, 2010; Ławryńczuk, 2015; Luo and Song, 2018; Salhi and Kamoun, 2016; Wills et al., 2013). Sin embargo, cuando el esquema de identificación está basado en la representación en espacio de estados, típicamente se requieren algoritmos de filtraje y suavizado (Agüero et al., 2012; Gibson and Ninness, 2005; Yuz et al., 2011). Adicionalmente, en muchas aplicaciones de control, redes de sensores, detección de fallas, sistemas de navegación y gestión del tráfico de carreteras, es clave poder realizar estimación de los estados del sistema usando mediciones de la salida ruidosa que típicamente pasa por algún tipo de no linealidad como saturación, cuantización, histéresis, zona muerta, entre otros (Mellodge, 2015). Por esta razón, es necesario desarrollar algoritmos de filtraje para este tipo de sistemas no lineales que permitan caracterizar los estados de un sistema Hammerstein-Wiener en cada instante, es decir, obtener la función de densidad de probabilidad (PDF) de cada estado condicionada a las mediciones y sus respectivas estadísticas como media y varianza. Estas estadísticas tienen un rol muy importante en lo que se denomina estimación de estados, ya que la media condicional es un estimador óptimo del estado y la varianza es una medida del error de estimación.

Un enfoque muy usado para diseñar algoritmos de filtraje para estimar los estados del sistema son los Métodos de Monte Carlo. Particularmente, el método conocido como muestreo secuencial de importancia (SIS), llamado también filtro de partículas, ver por ejemplo (Doucet et al., 2000a; Gordon et al., 1993a). En el filtro de partículas se usa un conjunto de muestras y pesos de una distribución, las mismas que al propagarse a través de las funciones no lineales del sistema permiten obtener estimaciones del estado directamente, sin conocer explícitamente la distribución a posteriori del mismo. La ventaja de estos métodos es su relativa facilidad de implementación, aunque pueden presentar un alto costo computacional cuando se usa un número elevado de muestras y/o cuando el orden del sistema crece. Además del filtro de partículas, existen otros enfoques de filtraje para sistemas no lineales en espacio de estados, los cuales bajo ciertas consideraciones permiten obtener una estimación de las PDFs de filtraje. Una de las técnicas más usadas es el filtro de Kalman extendido (Gelb et al., 1974; Grewal and Andrews, 2014), en el cual se usa una aproximación de Taylor del sistema no lineal en torno a una estimación del estado, para luego aplicar las ecuaciones del filtro de Kalman estándar. Otra técnica basada en el filtro de Kalman es el filtro de Kalman en cuadratura, (Arasaratnam et al., 2007; Singh and Bhaumik, 2015), en el cual se utilizan puntos de la cuadratura de Gauss (Hermite o Laguerre) para linealizar las funciones no lineales de estado y salida mediante una regresión lineal y así obtener soluciones cerradas de las PDFs de filtraje. Por otra parte, en el Unscented Kalman Filter (Julier and Uhlmann, 1997; Wan and Van Der Merwe, 2000) se busca aproximar directamente la media y varianza de la distribución del estado en lugar de aproximar funciones no lineales. Este filtro usa un número fijo de puntos sigma que se propagan a través de las funciones no lineales y así obtener una estimación del estado. La desventaja de estos métodos es que para ciertas no linealidades producen estimaciones poco precisas y pueden llegar a ser inestables (Wang et al., 2020). Existen además versiones de estos algoritmos de filtraje que explotan algunas características particulares del sistema y sus no linealidades tales como saturación en los estados o cuantización en la salida (Wen et al., 2018). Para más detalles sobre algoritmos de filtraje el lector puede referirse por ejemplo a (Huang and Wang, 2006; Särkkä, 2013).

Particularmente para sistemas Hammerstein-Wiener, se ha usado el filtro de Kalman extendido para la identificación de parámetros, como se muestra en (Mansouri et al., 2011). En (Wang et al., 2020) se presenta una modificación del filtro de Kalman extendido, en donde se realiza una re-linealización del sistema no lineal en torno a una predicción del estado, con el objetivo de mejorar el rendimiento y estabilidad del filtro. En (Wills et al., 2013) se presenta el filtro de partículas para sistemas Hammerstein-Wiener con el propósito de calcular los momentos requeridos para el cómputo de la función auxiliar en el algoritmo de identificación de parámetros Expectation-Maximization (EM). En (Andrieu and Doucet, 2002) se diseña el filtro de partículas marginalizado llamado también filtro de partículas Rao–Blackwellizado, para sistemas Wiener. Este enfoque consiste en usar el filtro de partículas estándar para estimar la parte no lineal del

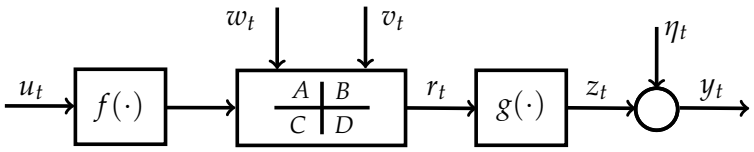


Figura 1.3: Diagrama de un sistema Hammerstein-Wiener en espacio de estados.

sistema y después, condicionados a esta estimación, usar el filtro de Kalman estándar para estimar la parte lineal del sistema. En (Albornoz et al., 2019) se propone una idea para filtraje de sistemas con datos cuantizados, donde un sistema LTI en espacio de estados es seguido de un bloque no lineal estático que corresponde a un cuantizador con infinitos niveles de cuantización. En dicho trabajo, los autores proponen usar una aproximación de la función de masa de probabilidad (PMF) de la salida no lineal dada una realización de estado mediante sumas de Riemann. Esta aproximación produce en esta función de probabilidad una estructura de suma de gaussianas, la misma que es usada para diseñar un filtro suma de gaussianas.

Este trabajo de tesis se puede considerar como una extensión y mejora a (Albornoz et al., 2019). Es una extensión, porque generalizamos el algoritmo de sumas de gaussianas a sistemas Hammerstein-Wiener, tal que el sistema posee no linealidades en la entrada y en la salida (la no linealidad en la salida no está restringida a ser un cuantizador). Es una mejora, porque usamos cuadratura gaussiana, que es un método más preciso que las sumas de Riemann para resolver integrales numéricamente. A continuación se define el problema a resolver.

1.1 Definición del Problema

Considere el modelo LTI MISO (por sus siglas en inglés Linear Time-Invariant Multiple-Input Single-output) en tiempo discreto y en espacio de estados que representa un sistema Hammerstein-Wiener dado por (ver Figura 1.3):

$$x_{t+1} = Ax_t + Bf(u_t) + w_t, \tag{1.1}$$
$$r_t = Cx_t + Df(u_t) + v_t, \tag{1.2}$$
$$z_t = g(r_t), \tag{1.3}$$
$$y_t = z_t + \eta_t, \tag{1.4}$$

donde $x_t \in \mathbb{R}^n$, $r_t \in \mathbb{R}$, $z_t \in \mathbb{R}$, $y_t \in \mathbb{R}$, y $u_t \in \mathbb{R}^m$ son: el vector de estados, la salida lineal (no medible), la salida no lineal (medible), la salida no lineal (disponible), y la entrada (determinística) del sistema, respectivamente. $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{1\times n}$ y $D \in \mathbb{R}^{1\times m}$ son las matrices del sistema. El ruido de estado $w_t \in \mathbb{R}^n$, el ruido de la salida $v_t \in \mathbb{R}$ y el ruido $\eta_t \in \mathbb{R}$ son variables aleatorias gaussianas mutuamente independientes de media cero y matrices de covarianzas Q, R, y P respectivamente. La condición inicial x_1 es una variable aleatoria independiente de w_t, v_t, y η_t que satisface $x_1 \sim \mathcal{N}(x_1; \mu_1, P_1)$, donde $\mathcal{N}(x; \mu, \Sigma)$ representa una PDF gaussiana de media μ y matriz de covarianza Σ correspondiente a la variable aleatoria x. Las funciones $f(\cdot)$ y $g(\cdot)$ son mapeos conocidos

$$f(\cdot) : \mathbb{R}^m \to \mathbb{R}^m, \tag{1.5}$$
$$g(\cdot) : \mathbb{R} \to \mathbb{R}. \tag{1.6}$$

Luego, el problema de interés es como sigue:

Problema 1. *Dado el conjunto de mediciones $y_{1:t} = \{y_1, y_2, \ldots, y_t\}$ de la salida no lineal y_t y el conjunto de datos de la entrada $u_{1:t} = \{u_1, u_2, \ldots, u_t\}$ correspondiente a la entrada u_t, para cada instante de tiempo t*

1. *Obtener la PDF condicional del estado x_t, $p(x_t|y_{1:t})$ y*

2. *Obtener el estimador de estados y la matriz de covarianza del error de estimación:*

$$\hat{x}_{t|t} = \mathbb{E}\left\{x_t | y_{1:t}\right\}, \tag{1.7}$$

$$\Sigma_{t|t} = \mathbb{E}\left\{(x_t - \hat{x}_{t|t})(x_t - \hat{x}_{t|t})^T | y_{1:t}\right\}, \tag{1.8}$$

donde $\mathbb{E}\{x|y\}$ *es la esperanza condicional de* x *dado* y.

1.2 Consideraciones Generales

En este trabajo se asume que la no linealidad $g(\cdot)$ es cualquier función no monótona que puede ser dividida en un número finito de M trozos en donde $g(\cdot)$ es estrictamente monótona, es decir, su inversa existe y su derivada es diferente de cero. Esta consideración ayuda a resolver el problema de filtraje ya que permite definir la PDF de la variable aleatoria z_t dado el estado x_t. Sin embargo, existen funciones en donde un número de trozos M_c con $M_c \leq M$ pueden ser contantes, es decir, su derivada es cero. Este es el caso de la función cuantización que asigna un valor constante a todos los puntos en un cierto intervalo. Por esta razón, para generalizar el algoritmo propuesto en este trabajo, también se considera un tipo particular de cuantización, llamada cuantización binaria, en la cual el cómputo de la función de probabilidad de la variable z_t dado el estado x_t es diferente al caso en que se satisface la monotonía estricta a trozos. Afortunadamente, una vez resueltos estos dos problemas, es relativamente sencillo extender estos resultados para tratar casi todas las funciones $g(\cdot)$, por ejemplo, las funciones saturación y zona muerta, las cuales pueden verse como una combinación de funciones estrictamente monótonas a trozos y el cuantizador binario.

La salida no lineal z_t está sujeta a las perturbaciones η_t como se observa en la ecuación 1.4. Esta configuración es típica cuando la salida no lineal z_t es medida con algún instrumento con mucha resolución pero con posibles errores de medición, o con ruidos ambientales, eléctricos, etc. Sin embargo, para tratar la función del cuantizador binario, es intuitivo asumir que no existen errores en las mediciones ya que la salida puede tomar solo dos valores, por ejemplo $\{0, 1\}$. En este caso, es decir $y_t = z_t$, debido a que el cuantizador binario no cumple con la monotonía estricta a trozos, la variable aleatoria $y_t|x_t$ es discreta y por lo tanto se trata de manera diferente a las funciones estrictamente monótonas a trozos.

Para propósitos de filtraje, asumimos que las no linealidades de la entrada y salida son mapeos conocidos. Esta consideración se hace cuando se conoce el sistema que produce la no linealidad, como por ejemplo, saturación en la señal de entrada debido a las limitaciones de los actuadores o cuantización en la señal de salida debido a la baja resolución de sensores usados para medición. Sin embargo, en algunas metodologías en el área de identificación de sistemas, estas no linealidades no se asumen conocidas si no que se estiman a partir de mediciones de la salida. Lo que se hace es asumir que las no linealidades o sus inversas están parametrizadas mediante un vector de parámetros y un conjunto de funciones base conocidas, y por lo tanto, se estiman en conjunto con los parámetros del sistema.

1.2.1 No linealidad en la entrada

El problema de interés de esta tesis es obtener una estimación de las PDFs de filtraje del sistema, así como una estimación de los estados y su correspondiente varianza del error de estimación. En general, en este tipo de problemas la entrada u_t es información dada por lo que se considera una señal determinística. Al ser información dada, las no linealidades de la entrada pueden ser cualquier tipo de función que se comporte bien.

1.2.2 No linealidad en la salida

La no linealidad de la salida influye directamente en el modelo estadístico de las PDFs de filtraje del sistema, por ello es muy importante definir el tipo de funciones que serán tratadas en esta tesis:

1. Función estrictamente monótona a trozos (Lax and Shea, 2014). Caso particular: Función Afín.
2. Función cuantización binaria (Widrow and Kollár, 2008).
3. Función saturación (Hu and Lin, 2001).
4. Función zona muerta (Mellodge, 2015).

En la Figura 1.4 se muestra un resumen de las funciones no lineales consideradas en este trabajo.

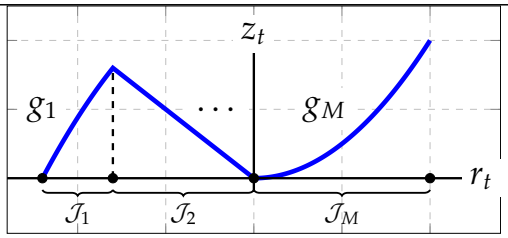
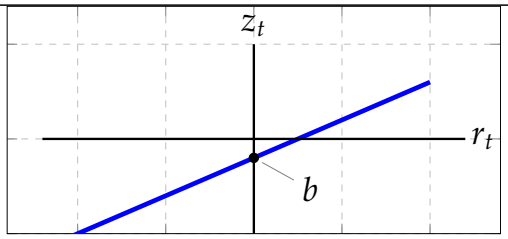
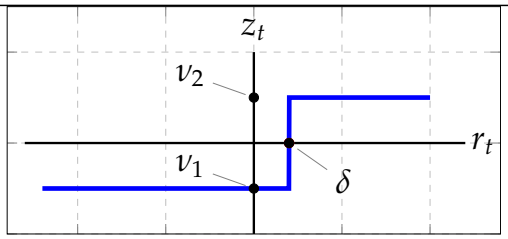
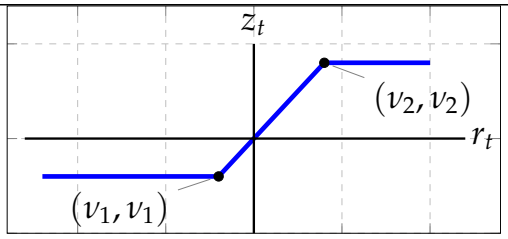
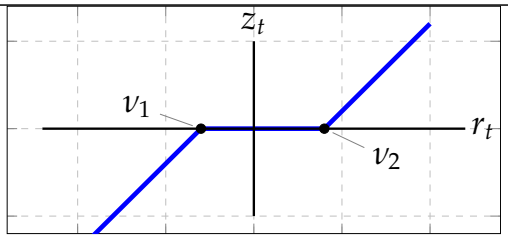
Nombre	Ecuación	Gráfica
Estrictamente Monótona a trozos	$z_t = \begin{cases} g_1(r_t) & \text{if} \quad r_t \in \mathcal{J}_1 \\ \vdots & \vdots \\ g_M(r_t) & \text{if} \quad r_t \in \mathcal{J}_M \end{cases}$	
Afín	$z_t = Kr_t + b$	
Cuantizador binario	$z_t = \begin{cases} \nu_1 & \text{if} \quad r_t < \delta \\ \nu_2 & \text{if} \quad r_t \geq \delta \end{cases}$	
Saturador	$z_t = \begin{cases} \nu_1 & \text{if} \quad r_t < \nu_1 \\ r_t & \text{if} \quad \nu_1 \leq r_t < \nu_2 \\ \nu_2 & \text{if} \quad r_t \geq \nu_2 \end{cases}$	
Zona muerta	$z_t = \begin{cases} r_t - \nu_1 & \text{if} \quad r_t < \nu_1 \\ 0 & \text{if} \quad \nu_1 \leq r_t < \nu_2 \\ r_t - \nu_2 & \text{if} \quad r_t \geq \nu_2 \end{cases}$	

Figura 1.4: Resumen de las funciones no lineales consideradas.

1.3 Principales Contribuciones de la Tesis

Las principales contribuciones de la tesis son las siguientes:

Principales Contribuciones

1. Se caracteriza la función de probabilidad de la salida no lineal condicionada al estado actual, considerando funciones no lineales que pueden ser divididas en un número finito de funciones que satisfacen monotonía estricta a trozos, incluyendo además a la función de cuantización binaria que no cumple con ser estrictamente monótona.
2. Se diseña un único algoritmo de filtraje para obtener el estimador de estados y las PDFs de filtraje de un sistema Hammerstein-Wiener, en el cual la no linealidad de la salida es cualquier función no lineal que puede ser dividida en un número finito de funciones que satisfacen monotonía estricta a trozos (incluyendo además a la función de cuantización binaria). Este algoritmo usa el modelo probabilístico de la salida no lineal dado el estado determinado en el punto anterior.
3. Se valida mediante simulaciones el correcto funcionamiento del algoritmo diseñado comparándolo con los algoritmos existentes en la literatura, evidenciado las mejoras en términos de la exactitud de las estimaciones y costo computacional.

1.4 Publicaciones asociadas

- Cedeño, A. L., Albornoz, R., Carvajal, R., Godoy, B. I., & Agüero, J. C. (2021). On Filtering Methods for State-Space Systems having Binary Output Measurements. IFAC-PapersOnLine, 54(7), 815–820.

  Abstract: In this paper we develop two filtering algorithms for state-space systems with binary outputs. We approximate the conditional probability mass function of the output signal given the state by using a Gaussian quadrature rule. This approximation naturally leads to a Gaussian Sum structure for the a posteriori density function. Our first algorithm is based on Gaussian Mixture models, and the second algorithm is based on Particle Filtering. Finally, we present numerical examples to illustrate the effectiveness of our proposal.

- Cedeño, A. L., Carvajal, R., & Agüero, J. C. (2021). A Novel Filtering Method for Hammerstein-Wiener State-Space Systems. 2021 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 1–7.

  Abstract: In this paper, we develop a novel filtering algorithm for Hammerstein-Wiener State-Space Systems. The likelihood function of the noisy nonlinear output signal given the system state is approximated by a Gaussian-Legendre quadrature rule. This approximation produces an explicit model of this likelihood function with a Gaussian Sum structure. Based on the general Bayesian filtering framework, we develop a Gaussian Sum Filter algorithm to obtain the a posteriori probability density function of the state given the current and past nonlinear output. With the characterization of the a posteriori probability function, we can obtain the associated statistics of the state. Finally, we present numerical examples to illustrate the benefits of our proposal.

1.5 Estructura del Documento

La estructura de los siguientes capítulos de esta tesis se describen a continuación.

- En el Capítulo 2 se presentan algunos conceptos y definiciones que se usarán a lo largo de este trabajo de tesis.
- En el Capítulo 3 se define el problema de filtraje bayesiano, el cual provee el marco teórico para realizar cualquier algoritmo de filtraje. Además, se caracteriza la función de probabilidad de la salida no lineal dado el estado como un modelo de suma de gaussianas. Esta función de probabilidad depende de cada función no lineal estudiada en esta tesis.
- En el Capítulo 4 se formula el algoritmo de filtraje sistemas Hammerstein-Wiener que permite obtener las PDFs de filtraje y el estimador de estados. Este algoritmo se basa en la función de probabilidad de la salida no lineal condicionada al estado que se define con un modelo de suma de gaussianas, por lo cual, el algoritmo de filtraje resultante es un filtro suma de gaussianas.
- En el Capítulo 5 se presentan ejemplos numéricos para evaluar el desempeño del algoritmo de filtraje propuesto. Se realizan comparaciones a nivel de simulación con algunos algoritmos de filtraje disponibles en la literatura.
- En el Capítulo 6 se presentan las conclusiones de la tesis y opciones de trabajo futuro.

2

Conceptos y definiciones

En este capítulo de la tesis se introducen algunos conceptos y definiciones que permitirán entender y desarrollar el algoritmo de filtraje para sistemas Hammerstein-Wiener. Se introduce el concepto de media condicional como un estimador de estados que satisface cierto criterio de optimalidad. Se explica el modelo de suma de gaussianas que permitirá describir la función de probabilidad de la salida no lineal condicionada al estado, punto clave para el desarrollo del algoritmo propuesto en esta tesis. Además, se revisarán conceptos de integración numérica, particularmente los métodos de cuadratura gaussiana, como Gauss-Legendre y Gauss-Hermite. El método de integración numérica Gauss-Legendre permitirá resolver ciertas integrales en el proceso de filtraje, y es este método el que produce en la función de probabilidad de la salida no lineal, un modelo de suma de gaussianas.

Contribución

La contribución principal de este capítulo es sentar las bases teóricas para el desarrollo del método de filtraje propuesto en esta tesis. Además, se reescribe la formulación estándar de la cuadratura de Gauss-Legendre, que está definida para el intervalo $[-1, 1]$, de tal forma de resolver integrales en intervalos finitos, semi-infinitos, e infinitos.

2.1 Estimación de estados

En muchas aplicaciones el objetivo de implementar un algoritmo de filtraje es obtener estimaciones de los estados del sistema con base en mediciones de la salida. Existen varios enfoques usados en la literatura para obtener estimadores que satisfagan algún criterio de optimalidad como, por ejemplo, el criterio de máxima verosimilitud, de máximo a posteriori y el de la media condicional (Crassidis and Junkins, 2012; Kamen and Su, 1999). En este trabajo usaremos el criterio de la media condicional para obtener estimaciones del estado del sistema que minimizan el error cuadrático medio.

2.1.1 Media condicional

Considere que se desea resolver el problema de estimar un vector x cuando sólo se tiene disponible mediciones de un vector y que está correlacionado con x. Si definimos como $\hat{x}$ un estimador de x entonces el error de estimación se define como

$$e = x - \hat{x}. \tag{2.1}$$

Un estimador se dice óptimo cuando el error de estimación tiene varianza mínima, es decir, $\mathbb{E}\left\{ee^T|y\right\}$ es la mínima posible. Para formular el problema de optimización de manera genérica, consideramos el siguiente funcional de costo matricial

$$\mathcal{P}\left[h(y)\right] = \mathbb{E}\left\{\left[x - h(y)\right]\left[x - h(y)\right]^T|y\right\}, \tag{2.2}$$

donde $h(y)$ es una estimación arbitraria de x. Luego, el estimador óptimo que minimiza $\mathcal{P}$ es $\hat{x} = \mathbb{E}\left\{x|y\right\}$, como se formaliza en el siguiente Lema (Anderson and Moore, 1979; Söderström, 2002):

Lema 2. *El estimador óptimo de x que minimiza cualquier función escalar monótonamente creciente de $\mathcal{P}\left[h(y)\right]$ definida en (2.2) es la media condicional $\hat{x} = \mathbb{E}\left\{x|y\right\}$.*

Demostración. *Considere*

$$\begin{aligned}
\mathcal{P}\left[h(y)\right] &= \mathbb{E}\left\{\left[x - h(y)\right]\left[x - h(y)\right]^T|y\right\}, && (2.3)\\
&= \mathbb{E}\left\{\left[x - \hat{x} + \hat{x} - h(y)\right]\left[x - \hat{x} + \hat{x} - h(y)\right]^T|y\right\}, && (2.4)\\
&= \mathbb{E}\left\{(x - \hat{x})(x - \hat{x})^T|y\right\} + \mathbb{E}\left\{(\hat{x} - h(y))(\hat{x} - h(y))^T|y\right\}, && (2.5)\\
&+ \mathbb{E}\left\{(x - \hat{x})(\hat{x} - h(y))^T|y\right\} + \mathbb{E}\left\{(\hat{x} - h(y))(x - \hat{x})^T|y\right\}, && (2.6)
\end{aligned}$$

luego, notando que $\mathbb{E}\left\{\hat{x}|y\right\} = \hat{x}$ y que $\mathbb{E}\left\{h(y)|y\right\} = h(y)$ entonces

$$\begin{aligned}
\mathcal{P}\left[h(y)\right] &= \mathbb{E}\left\{(x - \hat{x})(x - \hat{x})^T|y\right\} + \mathbb{E}\left\{(\hat{x} - h(y))(\hat{x} - h(y))^T|y\right\}, && (2.7)\\
&= \mathcal{P}\left[\hat{x}\right] + \mathbb{E}\left\{(\hat{x} - h(y))(\hat{x} - h(y))^T|y\right\}, && (2.8)\\
&\geq \mathcal{P}\left[\hat{x}\right], && (2.9)
\end{aligned}$$

con igualdad si $h(y) = \hat{x}$, es decir, $\mathcal{P}$ es mínimo cuando $h(y) = \hat{x}$, lo que completa la prueba. □

Observación 3. *Note que la desigualdad en (2.9) es en sentido matricial. Si $A, B \in \mathbb{R}^{n\times n}$ son matrices simétricas, entonces $A \geq B$ significa que $A - B \geq 0$, es decir, $x^T(A - B)x \geq 0$, $\forall x \neq 0$. Por otra parte,* $\det\{\mathcal{P}\}$ *y* $tr\{\mathcal{P}\}$ *son ejemplos de funciones escalares monótonamente creciente.*

2.2 Modelo de suma de gaussianas

El modelo de suma de gaussianas (GMM, por sus siglas en inglés Gaussian Mixture Models) (Bishop, 2006; Frühwirth, 2006; Frühwirth et al., 2019; McLachlan and Peel, 2004; Mengersen et al., 2011; Titterington et al., 1985), llamado también mezcla finita de gaussianas, es una superposición de PDFs gaussianas de la forma:

$$p(x) = \sum_{i=1}^{K} \rho_i \mathcal{N}\left(x; \mu_i, \Gamma_i\right), \qquad \text{s.t.} \qquad 0 \leq \rho_i \leq 1, \qquad \sum_{i=1}^{K} \rho_i = 1, \tag{2.10}$$

donde ρ_i es el i-ésimo peso, μ_i es la i-ésima media y Γ_i es la i-ésima matriz de covarianza. La PDF gaussiana viene dada por la siguiente expresión

$$\mathcal{N}(x;\mu_i,\Gamma_i) = \frac{1}{\sqrt{\det\{2\pi\Gamma_i\}}} \exp\left\{-\frac{1}{2}(x-\mu_i)^T \Gamma_i^{-1}(x-\mu_i)\right\}. \tag{2.11}$$

En la Figura 2.1 se puede observar un ejemplo de una suma de gaussianas con los siguientes parámetros: $\rho_1 = 0.25$, $\rho_2 = 0.4$, $\rho_3 = 0.35$, $\mu_1 = -8$, $\mu_2 = 0$, $\mu_3 = 8$, $\Gamma_1 = 3$, $\Gamma_2 = 10$, y $\Gamma_1 = 2$.

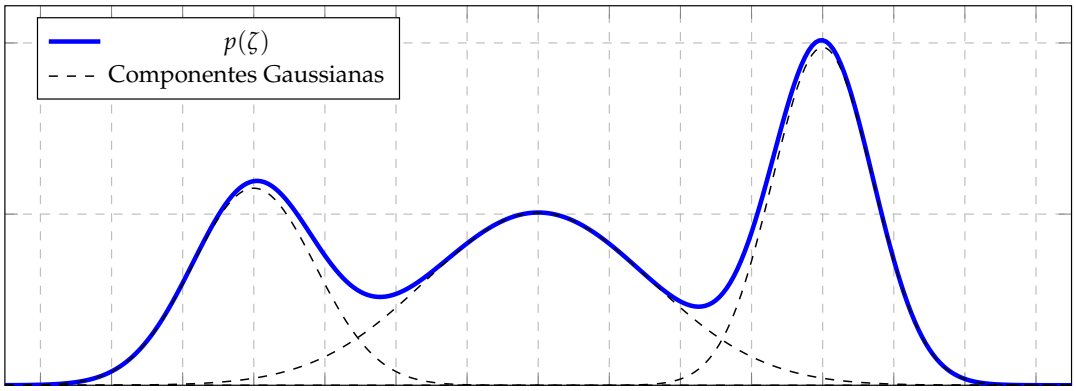


Figura 2.1: Modelo de una suma de gaussianas.

Los modelos GMM han sido utilizados en una variedad de aplicaciones. Particularmente se usan para aproximar otras distribuciones ya que son un conjunto aproximador universal (Lo, 1972). Las aplicaciones en donde se usan los modelos GMM incluyen: reconocimiento de voz (Povey et al., 2010), identificación de sistemas (Bittner et al., 2019; Cedeño et al., 2020; Orellana et al., 2020, 2021), filtraje bayesiano (Alspach and Sorenson, 1972; Balenzuela et al., 2019; Cedeño et al., 2021; Kitagawa, 1994a; Rahmathullah et al., 2014) segmentación de imágenes médicas (Song et al., 2010), por mencionar algunas.

2.3 Cuadratura gaussiana

Cuadratura gaussiana (Cohen, 2011; Davis and Rabinowitz, 1984; Krommer and Ueberhuber, 1998; Stoer and Bulirsch, 2002; Stroud and Secrest, 1966) engloba un conjunto de métodos dentro de la familia de integración numérica que permiten calcular integrales definidas en un intervalo $[a,b]$ que puede ser finito: $a < \infty$ y $b < \infty$, semi-infinito: $a = -\infty$ ó $b = \infty$, o infinito $a = -\infty$ y $b = \infty$, con la siguiente forma

$$\int_a^b \varrho(s)h(s)ds, \tag{2.12}$$

donde $\varrho(s)$ es la función de pesos, la cual debe satisfacer lo siguiente

- $\varrho(s) \geq 0$ es medible en el intervalo finito o infinito $[a,b]$.
- Todos los momentos (o integrales monómicas) c_k existen y son finitos, con

$$c_k = \int_a^b s^k \varrho(s)ds, \tag{2.13}$$

para $k = 1,2,\ldots$.

- Para polinomios $P(s)$ que son no negativos en $[a,b]$

$$\int_a^b \varrho(s)P(s)ds = 0 \rightarrow P(s) = 0. \tag{2.14}$$

En general, la regla de cuadratura establece que si $h(s)$ es un polinomio de algún orden, entonces existen pesos ω_τ y puntos ψ_τ tal que la integral en (2.12) es calculada de forma exacta mediante

$$\int_a^b \varrho(s)h(s)ds = \sum_{\tau=1}^{L} \omega_\tau h\left(\psi_\tau\right). \tag{2.15}$$

Los polinomios que satisfacen (2.15) son polinomios ortogonales para los cuales la función $\varrho(s)$ es única. Dependiendo del conjunto de polinomios usados, la regla de cuadratura gaussiana toma distintos nombres. Por ejemplo, usando los polinomios de Legendre se tiene la regla de cuadratura de Gauss-Legendre. De manera similar para las reglas de Gauss-Laguerre y Gauss-Hermite. En este trabajo se usa la regla de cuadratura de Gauss-Legendre para aproximar la función de probabilidad de la salida no lineal condicionada al estado del sistema. Además, en la literatura se usa la regla de cuadratura de Gauss-Hermite para una versión del filtro de Kalman (que se explica en apéndice A.1) llamado filtro de Kalman en cuadratura.

Para la regla de cuadratura de Gauss-Legendre la función de pesos se escoge como $\varrho(s) = 1$ con $a = -1$ y $b = 1$. Para la regla de cuadratura de Gauss-Hermite la función de pesos se escoge como $\varrho(s) = \exp\left\{-s^2\right\}$ con $a = -\infty$ y $b = \infty$.

2.3.1 Cuadratura de Gauss-Legendre

La cuadratura de Gauss-Legendre es una forma de integración numérica que consiste en aproximar una integral definida en el intervalo finito $[-1,1]$ de la siguiente forma

$$\int_{-1}^{1} h(s)ds \approx \sum_{\tau=1}^{L} \omega_\tau h\left(\psi_\tau\right), \tag{2.16}$$

donde ω_τ son los pesos y ψ_τ son las raíces de los polinomios de Legendre de grado L, los cuales vienen dados en su representación de sumatoria de la siguiente forma (Cohen, 2011):

$$P_L(s) = \frac{1}{2^L}\sum_{k=0}^{U}(-1)^k \frac{(2L-2k)!}{k!(L-k)!(L-2k)!}s^{L-2k}, \tag{2.17}$$

donde $U = (L-1)/2$ si L es impar y $U = L/2$ si L es par. Los ceros del polinomio $P_L(s)$, ψ_τ, permiten encontrar los pesos ω_τ de la siguiente manera:

$$\omega_\tau = \frac{1}{P'_L(\psi_\tau)}\int_{-1}^{1}\frac{P_L(s)}{s-\psi_\tau}ds = \frac{2}{\left(1-\psi_\tau^2\right)\left[P'_L(\psi_\tau)\right]^2}, \tag{2.18}$$

donde $P'_L(\psi_\tau)$ es la derivada de $P_L(s)$ con respecto a s y evaluada en los ceros ψ_τ. La última igualdad en (2.18) viene dada en (Abramowitz et al., 1988). Por ejemplo, la regla de cuadratura de Gauss-Legendre de 3 puntos usa el polinomio $P_3(s) = 0.5(5s^3 - 3s)$ con lo que se obtienen los siguientes pesos y puntos: $\omega_\tau = \left\{-\sqrt{(3/5)}, 0, \sqrt{(3/5)}\right\}$ y $\psi_\tau = \{5/9, 8/9, 5/9\}$ con $\tau = 1,2,3$. Note que la forma de calcular los pesos ω_τ difiere según la representación de los polinomios $P_L(s)$ en (2.17). Por ejemplo, la representación en forma recursiva de los polinomios de Legendre (Davis and Rabinowitz, 1984) produce otra ecuación para calcular ω_τ basada en la recursión de $P_L(s)$. Ademas, existen maneras computacionalmente rápidas para calcular los pesos como el método presentado en (Hale and Townsend, 2013). Para nuestro caso, los pesos se calculan de manera off-line y una sola vez cuando se ejecuta el correspondiente algoritmo de filtraje.

Con un apropiado cambio de variable, se puede usar la cuadratura de Gauss-Legendre para resolver integrales en cualquier intervalo finito, semi-infinito e infinito, como se muestra a continuación:

Integral sobre el intervalo $[a, b]$

Para resolver la integral definida en el intervalo $[a, b]$ se realiza el siguiente cambio de variable

$$s = \frac{b-a}{2}r + \frac{b+a}{2}, \qquad ds = \frac{b-a}{2}dr. \tag{2.19}$$

Este cambio de variable permite mapear la integral que se desea resolver al intervalo $[-1, 1]$, con lo que se obtiene

$$\int_a^b h(s)ds = \int_{-1}^{1} h\left(\frac{b-a}{2}r + \frac{b+a}{2}\right)\frac{b-a}{2}dr. \tag{2.20}$$

Una vez en esta forma, se puede usar la definición en (2.16) para obtener

$$\int_a^b h(s)ds \approx \sum_{\tau=1}^{L} \omega_\tau h\left(\frac{b-a}{2}\psi_\tau + \frac{b+a}{2}\right)\frac{b-a}{2}. \tag{2.21}$$

Integral sobre el intervalo $(-\infty, b]$

Esta integral puede ser mapeada al intervalo $(0, 1]$ usando el siguiente cambio de variable

$$s = b - \frac{1-r}{r}, \qquad ds = \frac{dr}{r^2}, \tag{2.22}$$

con lo que se obtiene

$$\int_{-\infty}^b h(s)ds = \int_0^1 h\left(b - \frac{1-r}{r}\right)\frac{dr}{r^2}. \tag{2.23}$$

Un nuevo cambio de variable $q = 2r - 1$ con $dq = 2dr$ produce

$$\frac{1-r}{r} = \frac{1-q}{1+q}. \tag{2.24}$$

Con este resultado se puede reescribir la integral deseada en el intervalo $(-1, 1]$, lo cual permite obtener:

$$\int_{-\infty}^b h(s)ds = \int_{-1}^1 h\left(b - \frac{1-q}{1+q}\right)\frac{2dq}{(1+q)^2}. \tag{2.25}$$

Usando la regla de cuadratura de Gauss-Legendre para resolver la integral anterior se tiene

$$\int_{-\infty}^b h(s)ds \approx \sum_{\tau=1}^{L} \omega_\tau h\left(b - \frac{1-\psi_\tau}{1+\psi_\tau}\right)\frac{2}{(1+\psi_\tau)^2}. \tag{2.26}$$

Integral sobre el intervalo $[a, \infty)$

Esta integral puede ser mapeada al intervalo $(0, 1]$ usando el siguiente cambio de variable

$$s = a + \frac{1-r}{r}, \qquad ds = -\frac{dr}{r^2}, \tag{2.27}$$

con lo que se obtiene

$$\int_a^\infty h(s)ds = \int_0^1 h\left(a + \frac{1-r}{r}\right)\frac{dr}{r^2}. \tag{2.28}$$

Usando el cambio de variable $q = 2r - 1$ con $dq = 2dr$ produce (2.24), obteniendo

$$\int_a^\infty h(s)ds = \int_{-1}^1 h\left(a + \frac{1-q}{1+q}\right)\frac{2dq}{(1+q)^2}. \tag{2.29}$$

Usando la regla de cuadratura de Gauss-Legendre se obtiene

$$\int_a^\infty h(s)ds \approx \sum_{\tau=1}^{L} \omega_\tau h\left(a + \frac{1-\psi_\tau}{1+\psi_\tau}\right)\frac{2}{(1+\psi_\tau)^2}. \tag{2.30}$$

Integral sobre el intervalo $(-\infty, \infty)$

La integral sobre el intervalo infinito $(-\infty, \infty)$ puede ser mapeada al intervalo finito $(-1, 1)$ usando el cambio de variable

$$s = \frac{r}{1-r^2}, \qquad ds = \frac{1+r^2}{(1-r^2)^2} dr, \tag{2.31}$$

lo que produce

$$\int_{-\infty}^{\infty} h(s)ds = \int_{-1}^{1} h\left(\frac{r}{1-r^2}\right) \frac{1+r^2}{(1-r^2)^2} dr. \tag{2.32}$$

Finalmente, usando la regla de cuadratura gaussiana se obtiene la siguiente aproximación

$$\int_{-\infty}^{\infty} h(s)ds \approx \sum_{\tau=1}^{L} \omega_\tau h\left(\frac{\psi_\tau}{1-\psi_\tau^2}\right) \frac{1+\psi_\tau^2}{(1-\psi_\tau^2)^2}. \tag{2.33}$$

2.3.2 Cuadratura de Gauss-Hermite

La cuadratura de Gauss-Hermite, como en el caso anterior, es una forma de integración numérica que consiste en aproximar una integral definida en el intervalo infinito $(-\infty, \infty)$ de la siguiente forma

$$\int_{-\infty}^{\infty} \exp\left\{-s^2\right\} h(s)ds \approx \sum_{\tau=1}^{L} \sigma_\tau h\left(\zeta_\tau\right), \tag{2.34}$$

donde σ_τ son los pesos y ζ_τ son las raíces de los polinomios de Hermite de grado L, los cuales vienen dados en su representación de sumatoria de la siguiente forma (Cohen, 2011):

$$P_L(s) = \sum_{k=0}^{U} (-1)^k \frac{L!}{k!(L-2k)!} (2s)^{L-2k}, \tag{2.35}$$

donde $U = (L-1)/2$ si L es impar y $U = L/2$ si L es par. Los ceros del polinomio $P_L(s)$, ζ_τ, permiten encontrar los pesos σ_τ de la siguiente manera:

$$\sigma_\tau = \frac{1}{P_L'(\zeta_\tau)} \int_{-\infty}^{\infty} \exp\left\{-s^2\right\} \frac{P_L(s)}{s-\zeta_\tau} ds, \tag{2.36}$$

donde $P_L'(\zeta_\tau)$ es la derivada de $P_L(s)$ con respecto a s y evaluada en los ceros ζ_τ. En (Arasaratnam et al., 2007; Golub and Welsch, 1969) se presenta una forma numéricamente estable de calcular los pesos σ_τ y puntos ζ_τ de la siguiente manera: suponga que se tiene una matriz $\mathcal{W}$ que es simétrica, tri-diagonal y con ceros en los elementos de la diagonal principal y con

$$\mathcal{W}_{i,i+1} = \sqrt{i/2}, \qquad 1 \leq i \leq (m-1), \tag{2.37}$$

entonces los puntos ζ_τ se escogen de tal manera que $\zeta_\tau = \sqrt{2}\varpi_\tau$ donde ϖ_τ es el τth autovalor de $\mathcal{W}$, y el correspondiente peso $\sigma_\tau = (\bar{\varpi})_1^2$ donde $(\bar{\varpi})_1$ es el primer elemento del τth autovector de $\mathcal{W}$.

3

El problema de filtraje bayesiano

3.1 Introducción

El filtraje bayesiano es una forma óptima de abordar el problema de filtraje, en el cual la optimalidad es en el sentido estadístico. Matemáticamente hablando, el filtraje óptimo bayesiano se refiere a un problema inverso estadístico en el cual una serie temporal de vectores $\{x_1, x_2, x_3, \dots\}$ se observan indirectamente a través de un conjunto de mediciones contaminadas con ruido $\{y_1, y_2, y_3, \dots\}$. Entonces, el problema que se desea resolver es estimar el conjunto de estados $x_{1:T} = \{x_1, x_2, \dots, x_T\}$ usando el conjunto observado $y_{1:T} = \{y_1, y_2, \dots, y_T\}$ (Särkkä, 2013). En términos bayesianos, esto significa caracterizar la PDF conjunta de los estados $x_{1:T}$ dadas las mediciones $y_{1:T}$, que mediante el teorema de Bayes se obtiene como:

$$p(x_{1:T}|y_{1:T}) = \frac{p(y_{1:T}|x_{1:T})p(x_{1:T})}{p(y_{1:T})}, \tag{3.1}$$

donde $p(x_{1:T})$ es la distribución a priori definida por la dinámica del sistema, $p(y_{1:T}|x_{1:T})$ es el modelo de probabilidad de la mediciones y $p(y_{1:T})$ es una constante de normalización. Sin embargo, esta formulación del problema de filtraje tiene un serio inconveniente: cada vez que una nueva medición está disponible, obtener la PDF conjunta en (3.1) que incorpore la información actual y_{T+1} significa realizar todos los cálculos del paso previo, lo que implica un alto costo computacional a medida que crece T. Afortunadamente, se puede explotar la propiedad markoviana del sistema representado en espacio de estados para desarrollar algoritmos recursivos que en cada iteración puedan incluir las nuevas mediciones sin incurrir en un costo computacional elevado.

Contribución

La principal contribución de este capítulo es la obtención del modelo de la función de probabilidad de la salida no lineal condicionada al estado actual, es decir la función $p(y_t|x_t)$, que es requisito en la formulación bayesiana para resolver el problema de filtraje. Se define el modelo de $p(y_t|x_t)$ para todas las funciones no lineales consideradas en esta tesis, a saber: función estrictamente monótona a trozos, cuantización binaria, saturación y zona muerta. El modelo obtenido para cada no linealidad tiene una única estructura de suma de gaussianas, aunque el cómputo de cada elemento de la función difiere según la no linealidad.

3.2 Filtraje bayesiano

En esta sección describiremos en detalle como funciona el proceso de filtraje bayesiano. Considere el sistema genérico discreto en espacio de estados dado en forma recursiva (Särkkä, 2013)

$$x_1 \sim p(x_1), \tag{3.2}$$
$$x_{t+1}|x_t \sim p(x_{t+1}|x_t), \tag{3.3}$$
$$y_t|x_t \sim p(y_t|x_t), \tag{3.4}$$

donde $x_t \in \mathbb{R}^n$, $y_t \in \mathbb{R}^p$ son el vector de estados y salidas respectivamente. La PDF $p(x_{t+1}|x_t)$ representa la distribución de probabilidad de transición de estados, la PDF $p(y_t|x_t)$ representa el modelo probabilístico de las mediciones, y la PDF $p(x_1)$ representa la distribución del estado inicial. Adicionalmente, bajo el supuesto de no correlación entre los ruidos de la ecuación de estados y de la salida y considerando que el sistema en espacio de estados es un proceso markoviano:

$$p(y_t|x_t, y_{1:t-1}) = p(y_t|x_t), \tag{3.5}$$
$$p(x_{t+1}|x_t, y_{1:t}) = p(x_{t+1}|x_t), \tag{3.6}$$

es decir, que la salida no depende de su pasado y que el estado en un instante solo depende del estado en un instante anterior, entonces, las ecuaciones de filtraje para el sistema (3.3)-(3.4) son (Söderström, 2002):

Teorema 4. *Considere el modelo en espacio de estados en (3.3)-(3.4), las mediciones de la salida y_t dadas hasta el instante t, $y_{1:t} = \{y_1, y_2, \ldots, y_t\}$, entonces las PDFs de corrección y predicción vienen dadas por:*

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \tag{3.7}$$
$$p(x_{t+1}|y_{1:t}) = \int p(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t, \tag{3.8}$$

donde $p(x_t|y_{1:t})$ y $p(x_{t+1}|y_{1:t})$ son las ecuaciones de corrección y predicción, respectivamente y $p(y_t|y_{1:t-1})$ es una constante de normalización dada por

$$p(y_t|y_{1:t-1}) = \int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t. \tag{3.9}$$

Demostración. *En primer lugar, para la PDF en la etapa de corrección se tiene*

$$\begin{aligned} p(x_t|y_{1:t}) &= p(x_t|y_t, y_{1:t-1}), \\ &\overset{(a)}{=} \frac{p(y_t|x_t, y_{1:t-1})p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \\ &\overset{(b)}{=} \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}. \end{aligned} \tag{3.10}$$

En (a) se ha usado el teorema de Bayes, que indica que $p(a,b) = p(b|a)p(a) = p(a|b)p(b)$. En (b) se ha usado el supuesto en (3.5), que indica que la medición en el instante actual t no depende de las mediciones pasadas en $t = 1, \ldots, t-1$. Nótese que la PDF $p(y_t|x_t)$ es conocida ya que viene del modelo de la salida en (3.4). La ecuación de la etapa de corrección permite incorporar la medición y_t en la estimación del estado x_t.

Para la etapa de predicción se tiene que $p(x_{t+1}|y_{1:t})$ se puede obtener mediante marginalización de la PDF conjunta del estado x_{t+1} y x_t como sigue:

$$\begin{aligned} p(x_{t+1}|y_{1:t}) &= \int p(x_{t+1}, x_t|y_{1:t})dx_t, \\ &\overset{(c)}{=} \int p(x_{t+1}|x_t, y_{1:t})p(x_t|y_{1:t})dx_t, \\ &\overset{(d)}{=} \int p(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t. \end{aligned} \tag{3.11}$$

En (c) se ha usado nuevamente el teorema de Bayes y en (d) se ha usado el supuesto en (3.6) que establece que el estado es independiente de las mediciones y solo depende del estado anterior. Nótese que la PDF $p(x_{t+1}|x_t)$ es conocida y se obtiene del modelo del sistema en (3.3). □

En la Figura 3.1 se muestra una representación de las dos etapas del filtraje bayesiano, la misma que se describe a continuación. Dada la PDF de la de condición inicial que corresponde a la PDF de predicción del tiempo $t = 0$

$$p(x_1|y_0) = p(x_1), \tag{3.12}$$

entonces se obtienen las PDFs de corrección y predicción en el tiempo $t = 1$ como sigue:

$$p(x_1|y_1) \propto p(y_1|x_1)p(x_1), \tag{3.13}$$

$$p(x_2|y_1) = \int p(x_2|x_1)p(x_1|y_1)dx_1, \tag{3.14}$$

luego, de manera similar se obtienen las PDFs de corrección y predicción en el tiempo $t = 2$:

$$p(x_2|y_{1:2}) \propto p(y_2|x_2)p(x_2|y_1), \tag{3.15}$$

$$p(x_3|y_{1:2}) = \int p(x_3|x_2)p(x_2|y_{1:2})dx_2, \tag{3.16}$$

y así sucesivamente. Note que hemos escrito las ecuaciones (3.13) y (3.15) en forma proporcional, ya

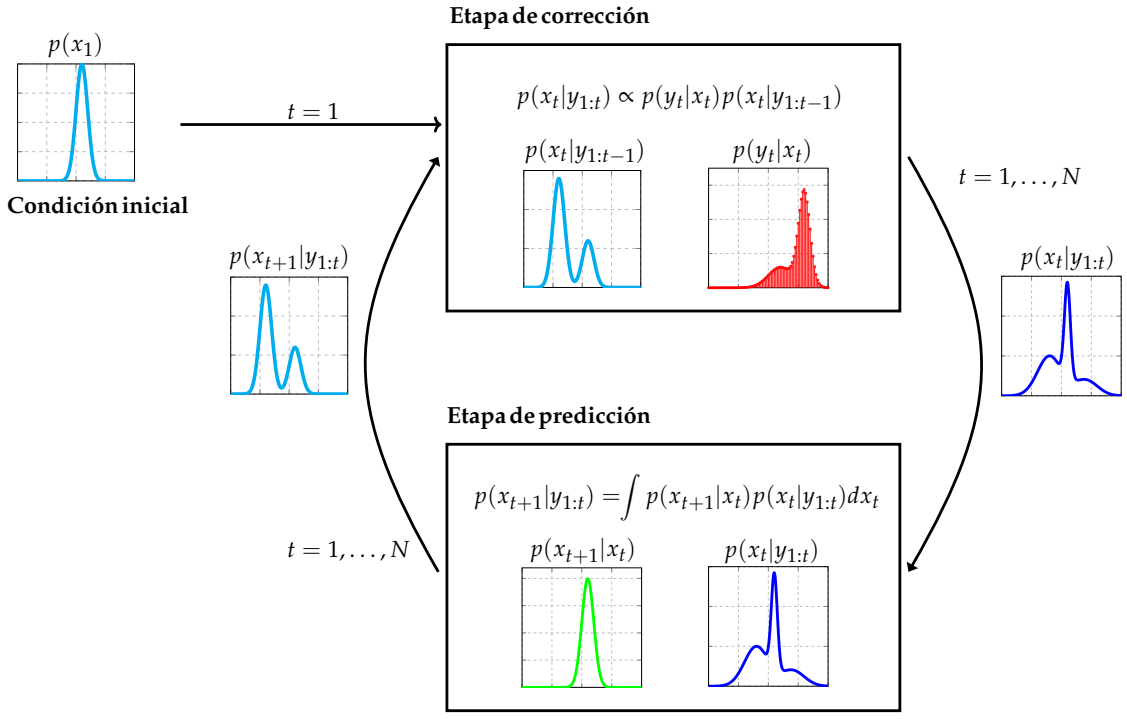


Figura 3.1: Representación de las etapas del filtraje bayesiano.

que la constante de normalización se obtiene una vez que se haya definido el producto $p(y_t|x_t)p(x_t|y_{1:t-1})$ según el modelo probabilístico de los ruidos de estado y salida. En la Figura 3.1 se muestra gráficamente el proceso de filtraje, donde se observa cómo se propaga el modelo Gaussiano o no-Gaussiano de $p(y_t|x_t)$ a todos los instantes futuros. Por ejemplo, supongamos que $p(x_1)$ es Gaussiano, entonces en el instante $t = 1$ la PDF $p(x_1|y_1)$ es no-gaussiana si $p(y_t|x_t)$ es no-gaussiana. En el mismo instante, la PDF de predicción $p(x_2|y_1)$ toma una característica similar a $p(x_1|y_1)$, asumiendo que $p(x_{t+1}|x_t)$ es gaussiana. Esto nos indica lo importante que son las PDFs de transición de estados, el modelo probabilístico de las mediciones, y la condición inicial en el proceso de filtraje.

Note que para usar las ecuaciones de filtraje bayesiano dadas en el Teorema 4 se requieren conocer las PDFs $p(x_{t+1}|x_t)$ y $p(y_t|x_t)$. Sólo en casos particulares muy sencillos estas PDFs se pueden obtener directamente del modelo del sistema en espacio de estados, como ocurre en el caso de un sistema lineal con ruidos gaussianos. Sin embargo, en general, estas PDFs se deben obtener (si es posible) de alguna forma tal que se puedan expresar en un modelo probabilístico que permita resolver las ecuaciones (3.7) y (3.8). Para el caso de interés en esta tesis, donde el sistema es lineal, la PDF de transición de estados $p(x_{t+1}|x_t)$ y de la salida lineal $p(r_t|x_t)$ se obtienen de la siguiente manera:

Lema 5. *Considere la ecuación de estados en (1.1) y la ecuación de salida en (1.2), entonces las PDFs $p(x_{t+1}|x_t)$ y $p(r_t|x_t)$ vienen dadas, respectivamente, por:*

$$p(x_{t+1}|x_t) = \mathcal{N}\left(x_{t+1}; Ax_t + Bf(u_t), Q\right), \tag{3.17}$$

$$p(r_t|x_t) = \mathcal{N}\left(r_t; Cx_t + Df(u_t), R\right). \tag{3.18}$$

Demostración. *Ver por ejemplo (Särkkä, 2013), capítulo 4, pág 56.* □

Por otro lado, obtener la PDF $p(y_t|x_t)$ es un poco más complicado ya que se debe considerar cómo la no linealidad de la salida afecta el modelo probabilístico de $p(y_t|x_t)$, lo que se describe a continuación.

3.3 Modelo de la función de probabilidad $p(y_t|x_t)$

El objetivo de esta sección es determinar el modelo probabilístico de la salida no lineal y_t dada una realización del estado, $p(y_t|x_t)$. Para ello se consideran las funciones no lineales descritas en la Figura 1.4: función estrictamente monótona a trozos, cuantizador binario, saturación y zona muerta.

3.3.1 Función estrictamente monótona a trozos

En este caso, la no linealidad $g(\cdot)$ puede ser cualquier función no monótona que se pueda dividir en un conjunto finito de M de funciones estrictamente monótonas, ver Figura 3.2. Esto significa que para cada división se cumple que su inversa existe y tiene derivada diferente de cero. Con esta división, la salida z_t se define de la siguiente manera:

$$z_t = \begin{cases} g_1(r_t) & \text{if} \quad r_t \in \mathcal{J}_1, \\ \vdots & \vdots \\ g_M(r_t) & \text{if} \quad r_t \in \mathcal{J}_M, \end{cases} \tag{3.19}$$

y la salida no lineal y_t de la cual se tienen mediciones está dada en (1.4).

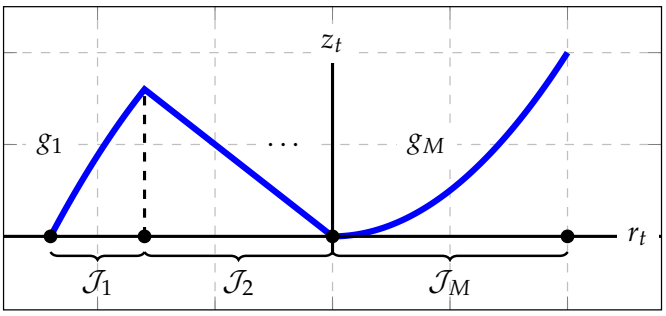


Figura 3.2: Gráfica de una función definida a trozos.

Considere la salida no lineal y_t dada en (1.4), donde z_t viene dada en (3.19), entonces la PDF $p(y_t|x_t)$ se puede obtener mediante el siguiente teorema:

Teorema 6. *Considere el sistema en (1.1)-(1.3), y (1.4) y las funciones $f(\cdot)$ y $g(\cdot)$ definidas como en (1.5)-(1.6). Asuma que el dominio de la variable aleatoria r_t puede ser particionado en un número finito de intervalos M en los cuales $g(\cdot)$ es una función estrictamente monótona y su inversa es diferenciable con derivada distinta de cero. Luego la PDF $p(y_t|x_t)$ de la salida no lineal, dada una realización del estado x_t, esta dada por:*

$$p(y_t|x_t) = \sum_{i=1}^{M} \int \phi_{it}(y_t - \eta_t)\mathcal{N}(\eta_t; 0, P)\,\mathcal{N}(\gamma_{it}(y_t - \eta_t); Cx_t + Df(u_t), R)\,d\eta_t, \tag{3.20}$$

donde $\gamma_{it}(z_t)$ son las raíces reales de la ecuación $z_t = g(r_t)$ y $\phi_{it}(z_t)$ es el valor absoluto de la derivada de γ_{it} con respecto a z_t, como sigue:

$$\gamma_{it}(z_t) = g^{-1}(z_t), \tag{3.21}$$

$$\phi_{it}(z_t) = \left|\frac{d\gamma_{it}(z_t)}{dz_t}\right|. \tag{3.22}$$

Por otro lado, la integral en la ecuación de $p(y_t|x_t)$ en (3.20) puede ser aproximada usando la técnica de cuadratura de Gauss-Legendre, lo que produce que $p(y_t|x_t)$ se pueda escribir de la siguiente manera

$$p(y_t|x_t) \approx \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right), \tag{3.23}$$

donde $K = ML$, L es el número de puntos de la regla de cuadratura de Gauss-Legendre (elegido por el usuario), j es un nuevo índice que combina cada par (i, τ) con $i = 1, \ldots, M$ y $\tau = 1, \ldots, L$ tal que $j = (\tau - 1)M + i$. Además β_j y ξ_t^j se definen como

$$\lambda_\tau = \psi_\tau/(1 - \psi_\tau^2), \tag{3.24}$$

$$\xi_t^j = \gamma_{it}(y_t - \lambda_\tau), \tag{3.25}$$

$$\beta_j = \omega_\tau \phi_{it}(y_t - \lambda_\tau)\mathcal{N}(\lambda_\tau; 0, P)\,(1 - \psi_\tau)/(1 - \psi_\tau^2)^2, \tag{3.26}$$

donde ω_τ y ψ_τ son definidos en la regla de cuadratura de Gauss-Legendre, dados por ejemplo en (Cohen, 2011).

Demostración. *Del sistema en (1.1), (1.2), (1.3), y (1.4) podemos observar que la PDF $p(y_t|x_t)$ puede ser*

obtenida mediante marginalización, como sigue

$$p(y_t|x_t) = \int p(y_t, \eta_t|x_t)d\eta_t. \tag{3.27}$$

Para obtener la distribución conjunta $p(y_t, \eta_t|x_t)$, usamos el teorema de transformación de variables aleatorias. Primero notamos que

$$\begin{bmatrix} y_t \\ \eta_t \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_t \\ \eta_t \end{bmatrix}, \qquad \rightarrow \qquad Y = FX, \tag{3.28}$$

y resolviendo esta ecuación para X tenemos

$$X = F^{-1}Y \qquad \rightarrow \qquad \begin{bmatrix} z_t \\ \eta_t \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_t \\ \eta_t \end{bmatrix}, \tag{3.29}$$

donde $\det F = 1$. El teorema de transformación de variables aleatorias establece que (Papoulis and Pillai, 1989):

$$p_Y(y) = \frac{1}{\det F} p_X(F^{-1}y). \tag{3.30}$$

En este caso, $p_X(x) = p(z_t, \eta_t|x_t)$ puede ser reescrita usando el teorema de Bayes como sigue

$$p(z_t, \eta_t|x_t) = p(z_t|x_t)p(\eta_t|x_t). \tag{3.31}$$

Luego, usando (3.30) podemos reescribir $p(y_t|x_t)$ como

$$p(y_t|x_t) = \int p(z_t|x_t)\Big|_{z_t=y_t-\eta_t} p(\eta_t|x_t)d\eta_t. \tag{3.32}$$

Para obtener la distribución $p(z_t|x_t)$, consideramos la función no monótona $g(\cdot)$ en (1.6), ver Figura 3.2. Luego, siguiendo la forma general del teorema de transformación de variables aleatorias (Papoulis and Pillai, 1989) [pag 130], obtenemos las raíces de la ecuación $z_t = g(r_t)$

$$\begin{aligned} \hat{\gamma}_{it}(z_t) &= g^{-1}(z_t) - Cx_t - Df(u_t), \\ &= \gamma_{it}(z_t) - Cx_t - Df(u_t). \end{aligned} \tag{3.33}$$

Note que, el término $-Cx_t - Df(u_t)$ no depende de z_t. Luego

$$\phi_{it}(z_t) = \left|\frac{d\hat{\gamma}_{it}(z_t)}{dz_t}\right| = \left|\frac{d\gamma_{it}(z_t)}{dz_t}\right|, \tag{3.34}$$

donde $|\cdot|$ indica la función valor absoluto. Ahora podemos escribir la distribución $p(z_t|x_t)$ como una suma de distribuciones obtenidas de las M funciones $g_i(\cdot)$ provenientes de $g(\cdot)$ como sigue

$$p(z_t|x_t) = \sum_{i=1}^{M} \phi_{it}(z_t)\mathcal{N}\left(\gamma_{it}(z_t); Cx_t + Df(u_t), R\right), \tag{3.35}$$

donde $\phi_{it}(z_t) \neq 0$ para que (3.35) exista. Finalmente el resultado mostrado en (3.20) se obtiene reemplazando (3.35) en (3.32).

Por otro lado, la regla de cuadratura de Gauss-Legendre para aproximar una integral en el intervalo infinito $(-\infty, \infty)$ puede ser calculada usando (2.33). Luego, usando (3.35) en (3.32), y usando la regla de cuadratura de Gauss-Legendre obtenemos

$$p(y_t|x_t) \approx \sum_{i=1}^{M}\sum_{\tau=1}^{L} \phi_{it}(y_t - \lambda_\tau)\mathcal{N}\left(\lambda_\tau; 0, P\right)\mathcal{N}\left(\gamma_{it}(y_t - \lambda_\tau); Cx_t + Df(u_t), R\right), \tag{3.36}$$

donde $\lambda_\tau = \psi_\tau/(1-\psi_\tau^2)$. Para reescribir la doble sumatoria en una sola usamos el Lema 20. Para cada par (i, τ) donde $i = 1, \ldots, M$ y $\tau = 1, \ldots, L$ definimos el nuevo índice $j = (\tau - 1)M + i$ tal que

$$p(y_t|x_t) \approx \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right), \tag{3.37}$$

donde $K = ML$, y ξ_t^j y β_j están definidos en (3.25), y (3.26), respectivamente. Con esto termina la prueba. □

3.3.2 Función Afín

Aunque esta función es lineal, ver Figura 3.3, para poder caracterizar otras no linealidades como la función saturación o la función zona muerta, es importante resolver el problema de filtraje para esta función, la cual está definida como:

$$z_t = \mathbb{A}r_t + b, \tag{3.38}$$

y la salida no lineal y_t de la cual se tiene mediciones está dada en (1.4). Note que la función afín ($\mathbb{A} \neq 0$) es un caso particular de la función definida en trozos, en la cual existe un solo intervalo $(-\infty, \infty)$ en donde la función es estrictamente monótona. Considerando la salida y_t dada en (1.4), donde z_t viene dada en (3.38), entonces la PDF $(y_t|x_t)$ se puede obtener mediante el siguiente corolario:

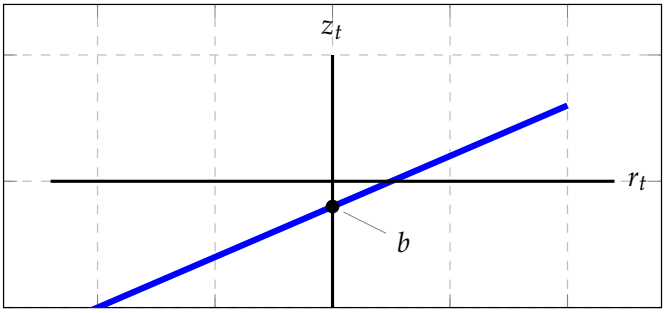


Figura 3.3: Gráfica de una función afín.

Corolario 7. *Dadas las condiciones del Teorema 6 donde z_t está definida en (3.38), entonces la PDF $p(y_t|x_t)$ de la salida dada una realización del estado x_t, está dada por:*

$$p(y_t|x_t) = \frac{1}{\mathbb{A}} \int \mathcal{N}(\eta_t; 0, P)\, \mathcal{N}\left(\frac{y_t - \eta_t - b}{\mathbb{A}}; Cx_t + Df(u_t), R\right) d\eta_t, \tag{3.39}$$

Por otro lado, la integral en la ecuación de $p(y_t|x_t)$ en (3.39) se resuelve de manera aproximada usando la regla de cuadratura de Gauss-Legendre, con lo cual

$$p(y_t|x_t) \approx \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right), \tag{3.40}$$

donde K es el número de puntos de la regla de cuadratura de Gauss-Legendre (elegido por el usuario). Además β_j y ξ_t^j se definen como

$$\lambda_\tau = \psi_\tau/(1-\psi_\tau^2), \tag{3.41}$$

$$\xi_t^j = \frac{y_t - \lambda_\tau - b}{\mathbb{A}}, \tag{3.42}$$

$$\beta_j = \omega_\tau \mathcal{N}(\lambda_\tau; 0, P)\,(1-\psi_\tau)/\mathbb{A}(1-\psi_\tau^2)^2, \tag{3.43}$$

donde ω_τ y ψ_τ son definidos en la regla de cuadratura de Gauss-Legendre, dados por ejemplo en (Cohen, 2011).

Demostración. *Directamente usando el Teorema 6.* □

Observación 8. *Note que al ser $g(\cdot)$ un función afín, entonces $p(y_t|x_t)$ admite una solución dada por:*

$$p(y_t|x_t) = \mathcal{N}\left(y_t; \mathbb{A}\left[Cx_t + Df(u_t)\right] + b, \mathbb{A}^2 R + P\right), \tag{3.44}$$

obtenida usando el resultado en el Lema 5. Por lo tanto, para esta función se podría usar el filtro de Kalman estándar. Sin embargo, aquí usaremos el filtro desarrollado en este trabajo para mantener la generalidad, por lo cual escribimos $p(y_t|x_t)$ como en el Teorema 6. En el apéndice A.3 demostramos que $p(y_t|x_t)$ dada en (3.44) es exactamente igual al $p(y_t|x_t)$ dada en (3.39) una vez calculada la integral. Esto indica que el filtro propuesto en este trabajo corresponde al filtro de Kalman cuando la función $g(\cdot)$ es una función afín, es decir, el sistema es lineal.

3.3.3 Función cuantización binaria

En este caso, ver Figura 3.4, la no linealidad $g(\cdot)$ es un cuantizador que puede tomar solo dos posibles valores $\{\nu_1, \nu_2\}$ cuando la salida supera o no un cierto umbral δ, como sigue:

$$z_t = \begin{cases} \nu_1 & \text{if} \quad r_t < \delta, \\ \nu_2 & \text{if} \quad r_t \geq \delta. \end{cases} \tag{3.45}$$

Note que este caso la salida y_t corresponde a la ecuación $y_t = z_t$, ya que se asume que la salida binaria que toma solo dos valores conocidos no tiene ruido.

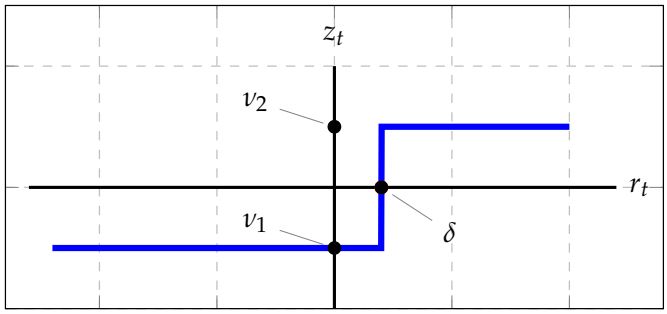


Figura 3.4: Gráfica de la función cuantización binaria

Teorema 9. *Considere el sistema en (1.1)-(1.3), y (1.4), y el cuantizador binario dado en (3.45). Luego la PMF de la variable aleatoria discreta y_t dada una realización del estado x_t, viene dada por*

$$p(y_t|x_t) = \int_{a_t}^{b_t} \mathcal{N}(v_t; 0, R)\, dv_t, \tag{3.46}$$

donde a_t y b_t son funciones de los valores extremos de cada región del cuantizador, con

$$a_t = \begin{cases} -\infty & y_t = \nu_1, \\ \delta - Cx_t - Df(u_t) & y_t = \nu_2, \end{cases} \tag{3.47}$$

$$b_t = \begin{cases} \delta - Cx_t - Df(u_t) & y_t = \nu_1, \\ \infty & y_t = \nu_2. \end{cases} \tag{3.48}$$

Usando la regla de cuadratura de Gauss-Legendre, la integral en la ecuación $p(y_t|x_t)$ dada en (3.46) puede ser aproximada como:

$$p(y_t|x_t) \approx \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right), \tag{3.49}$$

donde K es el número de puntos de la regla de cuadratura de Gauss-Legendre (elegido por el usuario), con β_j, y ξ_t^j definidos como:

$$\beta_j = 2\omega_j/(1+\psi_j)^2, \tag{3.50}$$

$$\xi_t^j = \begin{cases} \delta - (1-\psi_j)/(1+\psi_j) & \text{if} \quad y_t = \nu_1, \\ \delta + (1-\psi_j)/(1+\psi_j) & \text{if} \quad y_t = \nu_2, \end{cases} \tag{3.51}$$

donde ω_j y ψ_j vienen dados por la regla de cuadratura de Gauss-Legendre, ver por ejemplo (Cohen, 2011).

Demostración. *De la ecuación del cuantizador binario dada en (3.45), observamos que la variable aleatoria y_t solo puede tomar lo valores ν_1 o ν_2. Luego, la probabilidad de que $y_t = \nu_1$ o $y_t = \nu_2$ es la misma probabilidad de que la variable aleatoria r_t provenga del conjunto $\{r_t : r_t < \delta\}$ o $\{r_t : r_t \geq \delta\}$, respectivamente. Esta probabilidad puede ser calculada como sigue:*

$$\begin{aligned} \mathbb{P}\{y_t = \nu_1|x_t\} &= \mathbb{P}\{r_t < \delta|x_t\}, \\ \mathbb{P}\{y_t = \nu_2|x_t\} &= \mathbb{P}\{r_t \geq \delta|x_t\}, \end{aligned} \tag{3.52}$$

donde $\mathbb{P}\{\cdot\}$ indica probabilidad. Considerando (1.2), obtenemos

$$p(y_t|x_t) = \begin{cases} \mathbb{P}\{v_t < \delta - Cx_t - Df(u_t)|x_t\} & \text{if} \quad y_t = \nu_1, \\ \mathbb{P}\{v_t \geq \delta - Cx_t - Df(u_t)|x_t\} & \text{if} \quad y_t = \nu_2. \end{cases} \tag{3.53}$$

luego, usando $p(v_t) = \mathcal{N}(v_t; 0, R)$, obtenemos que

$$p(y_t|x_t) = \mathbb{P}\{a_t \leq v_t < b_t\} = \int_{a_t}^{b_t} \mathcal{N}(v_t; 0, R)\, dv_t, \tag{3.54}$$

donde los límites de integración a_t y b_t están definidos en (3.47) y (3.48), respectivamente. Por otro lado, esta integral puede ser aproximada usando la regla de cuadratura gaussiana como se explica a continuación. Para $y_t = \nu_2$, la integral en el intervalo semi-infinito $[a_t, \infty)$ se calcula como en (2.30), con lo que se obtiene

$$\begin{aligned} \int_{a_t}^{\infty} \mathcal{N}(v_t; 0, R)\, dv_t &\approx \sum_{j=1}^{K} \omega_j \mathcal{N}\left(a_t + \frac{1-\psi_j}{1+\psi_j}; 0, R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\delta - Cx_t - Df(u_t) + \frac{1-\psi_j}{1+\psi_j}; 0, R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\delta + \frac{1-\psi_j}{1+\psi_j}; Cx_t + Df(u_t), R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right), \end{aligned} \tag{3.55}$$

donde β_j, y ξ_t^j están definidos en (3.50)-(3.51). De manera similar, para $y_t = \nu_1$, la integral en el intervalo

semi-infinito $(-\infty, b_t]$ *se puede calcular como en (2.26), con lo que se obtiene*

$$\begin{aligned}\int_{-\infty}^{b_t} \mathcal{N}(v_t; 0, R)\, dv_t &\approx \sum_{j=1}^{K} \omega_j \mathcal{N}\left(b_t - \frac{1-\psi_j}{1+\psi_j}; 0, R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\delta - Cx_t - Df(u_t) - \frac{1-\psi_j}{1+\psi_j}; 0, R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\delta - \frac{1-\psi_j}{1+\psi_j}; Cx_t + Df(u_t), R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right),\end{aligned} \tag{3.56}$$

donde β_j, *y* ξ_t^j *están definidos en (3.50)-(3.51). Con esto termina la prueba* □

3.3.4 Función saturación

En este caso, ver Figura 3.5, la no linealidad de la salida $g(\cdot)$ es una mezcla entre los dos tipos de funciones anteriores, donde en algunos intervalos se satura comportándose como un cuantizador binario y en otros casos devuelve la señal de r_t sin modificaciones como sigue:

$$z_t = \begin{cases} \nu_1 & \text{if} \quad r_t < \nu_1, \\ r_t & \text{if} \quad \nu_1 \le r_t < \nu_2, \\ \nu_2 & \text{if} \quad r_t \ge \nu_2. \end{cases} \tag{3.57}$$

Luego, de manera similar a las no linealidades anteriores y considerando que $y_t = z_t$, la función de probabilidad $p(y_t|x_t)$ se determina mediante el siguiente teorema:

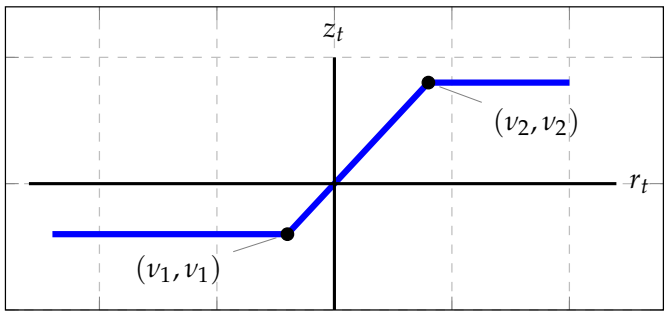


Figura 3.5: Gráfica de la función saturación.

Teorema 10. *Considere el sistema en (1.1)-(1.3), y (1.4), y la función saturación dada en (3.57). Luego la función de probabilidad de la variable aleatoria mixta* y_t *dada una realización del estado* x_t *viene dada por*

$$p(y_t|x_t) = \begin{cases} \displaystyle\int_{a_t}^{b_t} \mathcal{N}(v_t; 0, R)\, dv_t & si \quad y_t = \nu_1, \quad ó \quad y_t = \nu_2, \\ \displaystyle\int \mathcal{N}(\eta_t; 0, P)\, \mathcal{N}(y_t - \eta_t; Cx_t + Df(u_t), R)\, d\eta_t & si \quad \text{Otro caso} \end{cases} \tag{3.58}$$

donde a_t y b_t son funciones de los valores extremos de cada región del saturador, con

$$a_t = \begin{cases} -\infty & \text{if} \quad y_t = \nu_1, \\ \nu_2 - Cx_t - Df(u_t) & \text{if} \quad y_t = \nu_2, \end{cases} \tag{3.59}$$

$$b_t = \begin{cases} \nu_1 - Cx_t - Df(u_t) & \text{if} \quad y_t = \nu_1, \\ \infty & \text{if} \quad y_t = \nu_2. \end{cases} \tag{3.60}$$

Usando la regla de cuadratura de Gauss-Legendre, las integrales en la ecuación $p(y_t|x_t)$ dada en (3.58) pueden ser aproximadas como:

$$p(y_t|x_t) \approx \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right), \tag{3.61}$$

donde K es el número de puntos de la regla de cuadratura de Gauss-Legendre (elegido por el usuario), con β_j, y ξ_t^j definidos como:

$$\beta_j = \begin{cases} 2\omega_j/(1+\psi_j)^2 & si \quad y_t = \nu_1, \\ 2\omega_j/(1+\psi_j)^2 & si \quad y_t = \nu_2, \\ \omega_\tau \mathcal{N}(\lambda_\tau; 0, P)(1-\psi_\tau)/(1-\psi_\tau^2)^2 & si \quad Otro\ caso, \end{cases} \tag{3.62}$$

$$\xi_t^j = \begin{cases} \nu_1 - (1-\psi_j)/(1+\psi_j) & si \quad y_t = \nu_1, \\ \nu_2 + (1-\psi_j)/(1+\psi_j) & si \quad y_t = \nu_2, \\ y_t - \lambda_\tau & si \quad Otro\ caso, \end{cases} \tag{3.63}$$

donde $\lambda_\tau = \psi_\tau/(1-\psi_\tau^2)$, ω_j y ψ_j vienen dados por la regla de cuadratura de Gauss-Legendre, ver por ejemplo (Cohen, 2011).

Demostración. *De la ecuación de la función saturación dada en (3.57), observamos que la variable aleatoria y_t puede tomar los valores discretos ν_1 y ν_2, ó todo el rango de valores r_t. Luego, la probabilidad de que $y_t = \nu_1$ o $y_t = \nu_2$ es la misma probabilidad de que la variable aleatoria r_t provenga del conjunto $\{r_t : r_t < \nu_1\}$, o $\{r_t : r_t \geq \nu_2\}$, respectivamente. Esta probabilidad puede ser calculada usando la función de distribución como sigue:*

$$\begin{aligned} \mathbb{P}\{y_t = \nu_1|x_t\} &= \mathbb{P}\{r_t < \nu_1|x_t\}, \\ \mathbb{P}\{y_t = \nu_2|x_t\} &= \mathbb{P}\{r_t \geq \nu_2|x_t\}. \end{aligned} \tag{3.64}$$

Considerando (1.2), obtenemos

$$p(y_t|x_t) = \begin{cases} \mathbb{P}\{v_t < \nu_1 - Cx_t - Df(u_t)|x_t\} & si \quad y_t = \nu_1, \\ \mathbb{P}\{v_t \geq \nu_2 - Cx_t - Df(u_t)|x_t\} & si \quad y_t = \nu_2, \end{cases} \tag{3.65}$$

luego, usando $p(v_t) = \mathcal{N}(v_t; 0, R)$ obtenemos que

$$p(y_t|x_t) = \mathbb{P}\{a_t \leq v_t < b_t\} = \int_{a_t}^{b_t} \mathcal{N}(v_t; 0, R)\, dv_t, \tag{3.66}$$

donde los límites de integración a_t y b_t están definidos en (3.59) y (3.60), respectivamente. Por otro lado, para el caso cuando $y_t = r_t$ con $\{r_t : \nu_1 \leq r_t < \nu_2\}$, dado que en este intervalo la salida y_t es una variable aleatoria continua correspondiente a la salida lineal, entonces se puede obtener $p(y_t|x_t)$ usando el corolario 7 con $K = 1$ y $b = 0$ lo que permite obtener directamente la integral dada en (3.58).

Usando la regla de cuadratura gaussiana, podemos aproximar la integral en (3.66). En los intervalo de saturación, es decir, cuando $y_t = \nu_1$ o $y_t = \nu_2$, se tienen dos integrales en intervalos semi-infinitos como las que se resuelven en el Teorema 9 correspondiente al cuantizador binario. Es decir, para $y_t = \nu_2$, la integral en el intervalo semi-infinito $[a_t, \infty)$ se calcula como en (2.30) con lo que se obtiene

$$\begin{aligned}\int_{a_t}^{\infty} \mathcal{N}(v_t; 0, R)\, dv_t &\approx \sum_{j=1}^{K} \omega_j \mathcal{N}\left(a_t + \frac{1-\psi_j}{1+\psi_j}; 0, R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\nu_2 - Cx_t - Df(u_t) + \frac{1-\psi_j}{1+\psi_j}; 0, R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\nu_2 + \frac{1-\psi_j}{1+\psi_j}; Cx_t + Df(u_t), R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right),\end{aligned} \tag{3.67}$$

donde β_j, y ξ_t^j están definidos en (3.62) y (3.63). De manera similar, para $y_t = \nu_1$, la integral en el intervalo semi-infinito $(-\infty, b_t]$ se puede calcular como en (2.26) con lo que se obtiene

$$\begin{aligned}\int_{-\infty}^{b_t} \mathcal{N}(v_t; 0, R)\, dv_t &\approx \sum_{j=1}^{K} \omega_j \mathcal{N}\left(b_t - \frac{1-\psi_j}{1+\psi_j}; 0, R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\nu_1 - Cx_t - Df(u_t) - \frac{1-\psi_j}{1+\psi_j}; 0, R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\nu_1 - \frac{1-\psi_j}{1+\psi_j}; Cx_t + Df(u_t), R\right) \frac{2}{(1+\psi_j)^2}, \\ &= \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right),\end{aligned} \tag{3.68}$$

donde β_j y ξ_t^j están definidos en (3.62)-(3.63). Para $y_t = r_t$, se aproxima la integral en (3.58) como en el corolario 7 con $K = 1$ y $b = 0$, obteniéndose β_j y ξ_t^j dados en (3.62)-(3.63). Con esto termina la prueba □

Observación 11. *Note que para la función saturación se ha considerado $y_t = z_t$ ya que en dos de los intervalos la función se comporta como el cuantizador binario (que no tiene ruido en la ecuación de salida). Sin embargo, la parte lineal de la función saturación se trata como una función afín (ver el corolario 7) que si contempla ruido de medición. Para solucionar este inconveniente, como se verá en las simulaciones, es suficiente con escoger P pequeño para obtener estimaciones precisas.*

3.3.5 Función zona muerta

Este caso es similar al anterior, ver Figura 3.5, ya que la no linealidad de la salida $g(\cdot)$ posee un intervalo en donde devuelve cero y dos intervalos donde devuelve la señal de z_t desplazada en un valor ν_1 o ν_2 como sigue:

$$z_t = \begin{cases} r_t - \nu_1 & \text{if} \quad r_t < \nu_1, \\ 0 & \text{if} \quad \nu_1 \leq r_t < \nu_2, \\ r_t - \nu_2 & \text{if} \quad r_t \geq \nu_2. \end{cases} \tag{3.69}$$

La función de probabilidad $p(y_t|x_t)$ para esta no linealidad, considerando $y_t = z_t$, viene dada por el siguiente teorema:

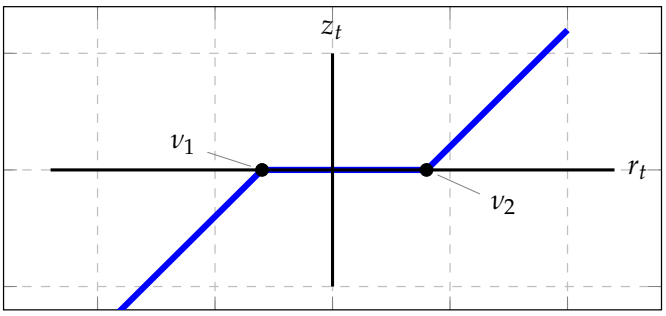


Figura 3.6: Gráfica de la función zona muerta.

Teorema 12. *Considere el sistema en (1.1)-(1.3), y (1.4), y la función zona muerta dada en (3.69). Luego, la función de probabilidad de la variable aleatoria mixta y_t dada una realización del estado x_t, viene dada por*

$$p(y_t|x_t) = \begin{cases} \int \mathcal{N}(\eta_t; 0, P)\, \mathcal{N}(y_t - \eta_t + \nu_1; Cx_t + Df(u_t), R)\, d\eta_t & si \quad y_t < 0, \\ \displaystyle\int_{a_t}^{b_t} \mathcal{N}(v_t; 0, R)\, dv_t & si \quad y_t = 0, \\ \int \mathcal{N}(\eta_t; 0, P)\, \mathcal{N}(y_t - \eta_t + \nu_2; Cx_t + Df(u_t), R)\, d\eta_t & si \quad yt > 0, \end{cases} \tag{3.70}$$

donde a_t y b_t son funciones de los valores extremos de la región cero en la función zona muerta, con

$$a_t = \nu_1 - Cx_t - Df(u_t), \tag{3.71}$$

$$b_t = \nu_2 - Cx_t - Df(u_t), \tag{3.72}$$

Usando la regla de cuadratura de Gauss-Legendre, las integrales en la ecuación $p(y_t|x_t)$ dadas en (3.70) pueden ser aproximadas como:

$$p(y_t|x_t) \approx \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right), \tag{3.73}$$

donde K es el número de puntos de la regla de cuadratura de Gauss-Legendre (elegido por el usuario), con β_j, y ξ_t^j definidos como:

$$\beta_j = \begin{cases} \omega_\tau \mathcal{N}(\lambda_\tau; 0, P)\,(1-\psi_\tau)/(1-\psi_\tau^2)^2 & si \quad y_t < 0, \\ \omega_j(\nu_2 - \nu_1)/2 & si \quad y_t = 0, \\ \omega_\tau \mathcal{N}(\lambda_\tau; 0, P)\,(1-\psi_\tau)/(1-\psi_\tau^2)^2 & si \quad y_t > 0, \end{cases} \tag{3.74}$$

$$\xi_t^j = \begin{cases} y_t - \lambda_\tau + \nu_1 & si \quad y_t < 0, \\ \psi_j(\nu_2 - \nu_1)/2 + (\nu_2 + \nu_1)/2 & si \quad y_t = 0, \\ y_t - \lambda_\tau + \nu_2 & si \quad y_t > 0, \end{cases} \tag{3.75}$$

donde $\lambda_\tau = \psi_\tau/(1-\psi_\tau^2)$, ω_j y ψ_j vienen dados por la regla de cuadratura de Gauss-Legendre, ver por ejemplo (Cohen, 2011).

Demostración. *De la ecuación de la función zona muerta dada en (3.69) observamos que la variable aleatoria y_t puede tomar un rango de valores cuando $r_t < \nu_1$ o cuando $r_t \geq \nu_2$, y un valor discreto, cero, cuando $\nu_1 < r_t \leq \nu_2$. Para los dos rangos continuos, las integrales en (3.70) se obtienen directamente al aplicar el*

corolario 7 con $K = 1$ y $b = -\nu_1$ para el caso cuando $r_t < \nu_1$, y con $K = 1$ y $b = -\nu_2$ para el caso cuando $r_t \geq \nu_2$. Además, la probabilidad de que $y_t = 0$ es la misma probabilidad de que la variable aleatoria r_t provenga del conjunto $\nu_1 \leq r_t < \nu_2$. Esta probabilidad puede ser calculada usando la función de distribución como sigue:

$$\mathbb{P}\left\{y_t = 0 | x_t\right\} = \mathbb{P}\left\{\nu_1 \leq r_t < \nu_2 | x_t\right\}. \tag{3.76}$$

Considerando (1.2), obtenemos

$$p(y_t|x_t) = \mathbb{P}\left\{\nu_1 - Cx_t - Df(u_t) \leq v_t < \nu_2 - Cx_t - Df(u_t) | x_t\right\}, \tag{3.77}$$

luego, usando $p(v_t) = \mathcal{N}(v_t; 0, R)$, obtenemos que

$$p(y_t|x_t) = \mathbb{P}\left\{a_t \leq v_t < b_t\right\} = \int_{a_t}^{b_t} \mathcal{N}\left(v_t; 0, R\right) dv_t, \tag{3.78}$$

donde los límites de integración a_t y b_t están definidos en (3.71) y (3.72), respectivamente.

Por otro lado, usando la regla de cuadratura gaussiana, podemos aproximar las integrales en (3.78). En los intervalos $r_t < \nu_1$ o $r_t \geq \nu_2$, la salida experimenta un desplazamiento a través de una función afín, por lo tanto las integrales en (3.70) se obtiene directamente del corolario 7. Para $y_t = 0$, la integral en el intervalo finito $[a_t, b_t)$ se puede calcular usando (2.21), lo que produce

$$\begin{aligned}
\int_{a_t}^{b_t} \mathcal{N}\left(v_t; 0, R\right) dv_t &\approx \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\frac{b_t - a_t}{2}\psi_j + \frac{b_t + a_t}{2}; 0, R\right) \frac{b_t - a_t}{2}, \\
&= \sum_{j=1}^{K} \omega_j \mathcal{N}\left(\frac{\nu_2 - \nu_1}{2}\psi_j + \frac{\nu_2 + \nu_1}{2}; Cx_t + Df(u_t), R\right) \frac{\nu_2 - \nu_1}{2}, \\
&= \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right),
\end{aligned} \tag{3.79}$$

donde β_j, y ξ_t^j están definidos en (3.74) y (3.75). Con esto termina la prueba. □

Observación 13. *Como en el caso de la función saturación, para la función zona muerta también se ha considerado $y_t = z_t$ ya que en dos de los intervalos la función se comporta como una función afín (ver el corolario 7). Sin embargo, en la zona muerta, la función se trata como un cuantizador binario (que no tiene ruido en la ecuación de salida). Para solucionar este inconveniente, como se verá en las simulaciones, es suficiente con escoger P pequeño para obtener estimaciones precisas.*

4

Algoritmo de filtraje para sistemas Hammerstein-Wiener

4.1 Introducción

En esta sección, usando el método de filtraje bayesiano, se obtiene el algoritmo de filtraje para sistemas Hammerstein-Wiener considerando el modelo probabilístico $p(y_t|x_t)$ que se obtuvo en la sección anterior. Note que este modelo fue descrito de tal manera que para todas las no linealidades $g(\cdot)$ resultó el mismo modelo con estructura de suma de gaussianas, y la única diferencia está en la forma de calcular las cantidades β_j y ξ_t^j. Por esta razón, se puede diseñar un único algoritmo del filtro suma de gaussianas ya que para lidiar con cada no linealidad basta con calcular β_j y ξ_t^j en $p(y_t|x_t)$ de acuerdo a dicha no linealidad.

Contribución

La principal contribución de este capítulo es desarrollar el algoritmo de filtraje para sistemas Hammerstein-Wiener. El algoritmo resultante es un filtro suma de gaussianas debido a la estructura de la función $p(y_t|x_t)$. Una vez resuelto el problema de filtraje y obtenida la PDF $p(x_t|y_{1:t})$ como una suma de gaussianas, el estimador de estados dado en (1.7) y la matriz de covarianza del error de estimación en (1.8) se definen con fórmulas cerradas.

4.2 Filtro suma de gaussianas (GSF)

Como se mencionó previamente, para obtener las ecuaciones de filtraje para el sistema dado en (3.2)-(3.4), en general se requiere conocer $p(x_{t+1}|x_t)$ y $p(y_t|x_t)$. Como se mostró en el Lema 5, la PDF de transición de estados $p(x_{t+1}|x_t)$ viene dada por:

$$p(x_{t+1}|x_t) = \mathcal{N}\left(x_{t+1}; Ax_t + Bf(u_t), Q\right). \tag{4.1}$$

Además, como se mostró en los Teoremas 6, 9, 10 y 12, la función de probabilidad $p(y_t|x_t)$ tiene la siguiente forma

$$p(y_t|x_t) \approx \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right). \tag{4.2}$$

Una vez definidas las funciones necesarias podemos usar las ecuaciones generales de filtraje bayesiano (3.7) y (3.8) definidas en el Teorema 4 para derivar el algoritmo filtraje deseado. Note que, debido a que $p(y_t|x_t)$ tiene una forma de suma de gaussianas, la ecuación de la etapa de corrección $p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1})$ en (3.7) es el producto de dos modelos de sumas de gaussianas, lo que produce un nuevo modelo de suma de gaussianas que tiene más componentes que las dos sumas de gaussianas originales. Esto implica que la ecuación de la etapa de predicción $p(x_{t+1}|y_{1:t})$ en (3.8) sea también un modelo de suma de gaussianas. Para entender cómo el producto de dos sumas de gaussianas se transforma en otra suma de gaussianas ver el lema 20.

Usando el modelo explícito para $p(y_t|x_t)$ que se determinó en el capítulo anterior para cada una de las funciones no lineales tratadas en este trabajo y siguiendo (Kitagawa, 1994a), se diseña el algoritmo de filtraje de sumas de gaussianas (GSF) para sistemas Hammerstein-Wiener en espacio de estados, el cual se resume en el siguiente teorema:

Teorema 14. *Considere el sistema en (1.1)-(1.3) y el modelo de $p(y_t|x_t)$ en los Teoremas 6, 9, 10 y 12. Luego, el filtro suma de gaussianas para sistemas Hammerstein-Wiener en espacio de estados está dado de la siguiente manera:*

Inicialización: *Para el instante $t = 1$ la PDF de la etapa de predicción está dada por*

$$p(x_1) = \mathcal{N}(x_1; \mu_1, P_1), \tag{4.3}$$

y para $t = 1, \ldots, N$ se definen los siguientes pasos:

Etapa de corrección: *la PDF de filtraje del estado actual x_t dadas las mediciones de la salida no lineal $y_1, \ldots, y_t$, es la suma de gaussianas definida como*

$$p(x_t|y_{1:t}) = \sum_{\ell=1}^{\mathcal{S}_{t|t}} \delta_{t|t}^{\ell} \mathcal{N}(x_t; \hat{x}_{t|t}^{\ell}, \Gamma_{t|t}^{\ell}), \tag{4.4}$$

donde $\mathcal{S}_{t|t} = K\mathcal{S}_{t|t-1}$, con K definido según la aproximación de $p(y_t|x_t)$. Para cada par (j,k), donde $j = 1, \ldots, K$ y $k = 1, \ldots, \mathcal{S}_{t|t-1}$, se obtiene un nuevo índice $\ell = (k-1)K + j$ de tal manera que los pesos, medias y matrices de covarianzas están dadas por

$$\delta_{t|t}^{\ell} = \frac{\bar{\delta}_{t|t}^{\ell}}{\sum_{r=1}^{\mathcal{S}_{t|t}} \bar{\delta}_{t|t}^{r}}, \tag{4.5}$$

$$\bar{\delta}_{t|t}^{\ell} = \beta_j \delta_{t|t-1}^{k} \mathcal{N}\left(\xi_t^j; \varsigma_t^k, \Phi_t^k\right), \tag{4.6}$$

$$\varsigma_t^k = C\hat{x}_{t|t-1}^k + Df(u_t), \tag{4.7}$$

$$\Phi_t^k = R + C\Gamma_{t|t-1}^k C^T, \tag{4.8}$$

$$K_t^k = \Gamma_{t|t-1}^k C^T \left(\Phi_t^k\right)^{-1}, \tag{4.9}$$

$$\hat{x}_{t|t}^{\ell} = \hat{x}_{t|t-1}^k + K_t^k(\xi_t^j - \varsigma_t^k), \tag{4.10}$$

$$\Gamma_{t|t}^{\ell} = (I - K_t^k C)\Gamma_{t|t-1}^k, \tag{4.11}$$

donde β_j, ξ_t^j están definidos en los Teoremas 6, 9, 10 y 12 según cada no linealidad. Los valores iniciales de la recursión son: $\mathcal{S}_{1|0} = 1$, $\delta_{1|0} = 1$, $\hat{x}_{1|0} = \mu_1$, y $\Gamma_{1|0} = P_1$.

***Etapa de predicción:** la PDF de predicción del estado x_{t+1}, dadas las mediciones de la salida no lineal $y_1, \ldots, y_t$, es la suma de gaussianas definida como*

$$p(x_{t+1}|y_{1:t}) = \sum_{\ell=1}^{\mathcal{S}_{t+1|t}} \delta_{t+1|t}^{\ell} \mathcal{N}(x_{t+1}; \hat{x}_{t+1|t}^{\ell}, \Gamma_{t+1|t}^{\ell}), \tag{4.12}$$

donde $\mathcal{S}_{t+1|t} = \mathcal{S}_{t|t}$, y los pesos, medias y matrices de covarianzas están dadas por

$$\delta_{t+1|t}^{\ell} = \delta_{t|t}^{\ell}, \tag{4.13}$$
$$\hat{x}_{t+1|t}^{\ell} = A\hat{x}_{t|t}^{\ell} + Bf(u_t), \tag{4.14}$$
$$\Gamma_{t+1|t}^{\ell} = Q + A\Gamma_{t|t}^{\ell} A^T. \tag{4.15}$$

Demostración. *Considere las ecuaciones de filtraje bayesiano dadas en el Teorema 4. En la ecuación de la etapa de corrección (3.7), la constante de normalización $p(y_t|y_{1:t-1})$ viene dada por*

$$p(y_t|y_{1:t-1}) = \int p(y_t|x_t) p(x_t|y_{1:t-1}) dx_t. \tag{4.16}$$

Luego, tanto el numerador como el denominador se calculan usando el producto $p(y_t|x_t)p(x_t|y_{1:t-1})$, por esta razón definimos

$$F(x_t, y_t) = p(y_t|x_t) p(x_t|y_{1:t-1}). \tag{4.17}$$

Dado que las ecuaciones de filtraje son recursivas, y si notamos que $p(x_t|y_{1:t-1})$ es la versión retrasada un instante de $p(x_{t+1}|y_{1:t})$ de la cual argumentamos antes que es una suma de gaussianas, entonces podemos asumir que tiene la siguiente forma

$$p(x_t|y_{1:t-1}) = \sum_{k=1}^{\mathcal{S}_{t|t-1}} \delta_{t|t-1}^{k} \mathcal{N}(x_t; \hat{x}_{t|t-1}^{k}, \Gamma_{t|t-1}^{k}), \tag{4.18}$$

con lo cual el producto $F(x_t, y_t)$, considerando el modelo $p(y_t|x_t)$ dado en los Teoremas 6, 9, 10 y 12, sería

$$F(x_t, y_t) = \sum_{j=1}^{K} \sum_{k=1}^{\mathcal{S}_{t|t-1}} \beta_j \delta_{t|t-1}^{k} \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right) \mathcal{N}\left(x_t; \hat{x}_{t|t-1}^{k}, \Sigma_{t|t-1}^{k}\right), \tag{4.19}$$

usando el Lema 19 con $\mathcal{N}(x_t; \hat{x}_{t|t-1}^{k}, \Sigma_{t|t-1}^{k})$ como $p(x)$ y $\mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right)$ como $p(y|x)$ se obtiene

$$F(x_t, y_t) = \sum_{j=1}^{K} \sum_{k=1}^{\mathcal{S}_{t|t-1}} \beta_j \delta_{t|t-1}^{k} \mathcal{N}\left(\xi_t^j; \varsigma_t^k, \Phi_t^k\right) \mathcal{N}\left(x_t; v_t^{jk}, U_t^k\right), \tag{4.20}$$

donde ς_t^k y Φ_t^k están definidos en (4.7) y (4.8) respectivamente. Note que una vez que se tiene una nueva medición y_t, el término ξ_t^j dado en (3.25), (3.42), (3.51), (3.63), y (3.75), en dicho instante es una constante, por lo cual $\mathcal{N}\left(\xi_t^j; \varsigma_t^k, \Phi_t^k\right)$ es sólo un coeficiente numérico. Además, usando el Lema 20, podemos reescribir la doble sumatoria en $F(x_t, y_t)$ como una sola sumatoria, definiendo un nuevo índice $\ell = (k-1)K + j$ tal que

$$F(x_t, y_t) = \sum_{\ell=1}^{\mathcal{S}_{t|t}} \bar{\delta}_{t|t}^{\ell} \mathcal{N}\left(x_t; \hat{x}_{t|t}^{\ell}, \Sigma_{t|t}^{\ell}\right), \tag{4.21}$$

donde $\bar{\delta}^k_{t|t}$, $\hat{x}^k_{t|t}$, $\Sigma^k_{t|t}$ *están definidos en (4.5), (4.10), y (4.11), respectivamente, y* $\mathcal{S}_{t|t} = K\mathcal{S}_{t|t-1}$. *Note que hemos definido* $\hat{x}^\ell_{t|t} \equiv v^{jk}_t$ *y* $\Sigma^\ell_{t|t} \equiv U^k_t$ *para resaltar la transformación de índices de una sumatoria doble a una simple. Luego, la constante de normalización viene dada por*

$$p(y_t|y_{1:t-1}) = \int F(x_t, y_t)dx_t = \sum_{k=1}^{\mathcal{S}_{t|t}} \bar{\delta}^k_{t|t}, \tag{4.22}$$

con lo cual, la ecuación de filtraje de la etapa de corrección es la suma de gaussianas

$$p(x_t|y_{1:t}) = \sum_{\ell=1}^{\mathcal{S}_{t|t}} \delta^\ell_{t|t} \mathcal{N}\left(x_t; \hat{x}^\ell_{t|t}, \Sigma^\ell_{t|t}\right), \tag{4.23}$$

con $\delta^k_{t|t} = \bar{\delta}^k_{t|t} / \sum_{s=1}^{\mathcal{S}_{t|t}} \bar{\delta}^s_{t|t}$. *Por otro lado, la ecuación de la etapa de predicción se obtiene como*

$$\begin{aligned} p(x_{t+1}|y_{1:t}) &= \int p(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t, \\ &= \sum_{\ell=1}^{\mathcal{S}_{t|t}} \delta^\ell_{t|t} \int \left[\mathcal{N}(x_{t+1}; Ax_t + Bf(u_t), Q)\mathcal{N}\left(x_t; \hat{x}^\ell_{t|t}, \Sigma^\ell_{t|t}\right)\right] dx_t. \end{aligned} \tag{4.24}$$

Usando el Lema 19 con $\mathcal{N}(x_t; \hat{x}^\ell_{t|t}, \Sigma^\ell_{t|t})$ *como* $p(x)$ *y* $\mathcal{N}(x_{t+1}; Ax_t + Bf(u_t), Q)$ *como* $p(y|x)$ *obtenemos*

$$p(x_{t+1}|y_{1:t}) = \sum_{\ell=1}^{\mathcal{S}_{t|t}} \delta^\ell_{t|t} \int \left[\mathcal{N}\left(x_{t+1}; \hat{x}^\ell_{t+1|t}, \Sigma^\ell_{t+1|t}\right)\mathcal{N}\left(x_t; m^k_t, S^k_t\right)\right] dx_t, \tag{4.25}$$

donde $\delta^\ell_{t+1|t}$, $\hat{x}^\ell_{t+1|t}$ *y* $\Sigma^\ell_{t+1|t}$ *están definidos en (4.13), (4.14), y (4.15), respectivamente. Note que* $\mathcal{N}(x_t; m^k_t, S^k_t)$ *es una PDF cuya integral en* $\mathbb{R}$ *es igual a uno. Entonces*

$$p(x_{t+1}|y_{1:t}) = \sum_{\ell=1}^{\mathcal{S}_{t+1|t}} \delta^\ell_{t|t} \mathcal{N}\left(x_{t+1}; \hat{x}^\ell_{t+1|t}, \Sigma^\ell_{t+1|t}\right), \tag{4.26}$$

donde $\mathcal{S}_{t+1|t} = \mathcal{S}_{t|t}$. *Lo que completa la prueba.* □

4.2.1 Reducción de gaussianas

Note que en la etapa de corrección del Teorema 14, el número de componentes gaussianas en cada iteración sigue la siguiente ecuación $\mathcal{S}_{t|t} = K\mathcal{S}_{t|t-1}$. Es claro que este número de componentes gaussianas crece muy rápidamente con el tiempo. Por ejemplo si $K = 3$, en 10 iteraciones del algoritmo se tendrían 59049 componentes gaussianas y en 20 iteraciones se tendrían alrededor de 3486 millones, como se muestra en la Figura 4.1

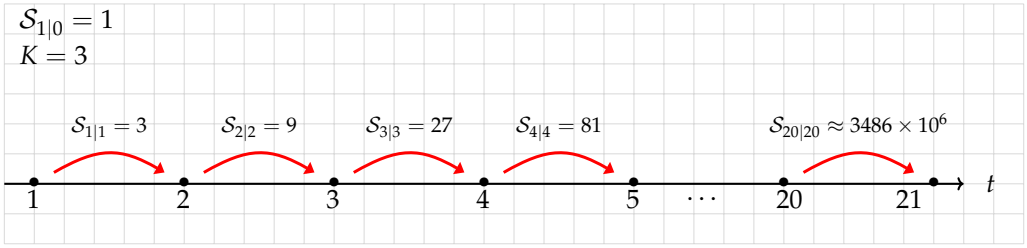


Figura 4.1: Ejemplo del crecimiento exponencial del número de componentes gaussianas en la etapa de corrección.

Este crecimiento exponencial causa que el costo computacional del algoritmo sea prácticamente inmanejable. Por esta razón, es necesario limitar este crecimiento en cada iteración de tal manera de mantener la representación de las PDFs de filtraje pero con una cantidad reducida de componentes gaussianas. Es decir, reducir la suma de gaussianas en la variable x

$$p(x) = \sum_{i=1}^{H} \alpha_i \mathcal{N}\left(x; \mu_i, \Gamma_i\right), \tag{4.27}$$

con $H \in \mathbb{N}$ reduciendo el número de componentes tal que

$$p(x) = \sum_{i=1}^{J} \alpha_i \mathcal{N}\left(x; \mu_i, \Gamma_i\right), \tag{4.28}$$

con $J \in \mathbb{N}$ tal que $1 \leq J \leq H$. En la literatura existen técnicas que realizan la reducción de componentes gaussianas (GSR). Estos incluyen a los métodos basados en *Pruning*, donde la idea es mantener solo aquellas componentes cuyo peso es considerable y despreciar el resto. Típicamente se define un umbral para descartar aquellas componentes cuya contribución no supera dicho límite (Arasaratnam et al., 2007). También existen métodos de *Joining* en donde la idea es combinar pares de componentes que minimizan una medida de similaridad (Runnalls, 2007). En la literatura también encontramos los métodos basados en la integral del error cuadrático, los cuales se enfocan en considerar todas las componentes gaussianas (no en pares) y minimizar un funcional de costo basado en la integral del error cuadrático entre la suma de gaussianas original y la suma de gaussianas reducida (Arasaratnam et al., 2007). Cada una de estas técnicas tienen sus pros y contras en términos de precisión de la aproximación así como en el costo computacional, siendo la más fácil, rápida y menos precisa la técnica de pruning, seguida por joining y finalmente, con alto costo computacional, la técnica basada en la integral del error cuadrático. En esta tesis usamos la técnica de joining, combinando dos componentes que minimizan la medida de similaridad dada por el número Kullback-Leibler. Esta idea fue propuesta inicialmente por (Kitagawa, 1994b), luego en (Runnalls, 2007) se propone un método basado en esta técnica, el cual combina dos componentes de tal manera que se preserven el primer y segundo momento de las componentes gaussianas originales, de la siguiente manera:

$$\left(\alpha_{ij}, \mu_{ij}, \Gamma_{ij}\right) = \mathcal{M}\left\{\left(\alpha_i, \mu_i, \Gamma_i\right), \left(\alpha_j, \mu_j, \Gamma_j\right)\right\}, \tag{4.29}$$

donde $\mathcal{M}\{\cdot,\cdot\}$ es una función que convierte dos componentes gaussianas en una sola tal que:

$$\alpha_{ij} = \alpha_i + \alpha_j, \tag{4.30}$$

$$\mu_{ij} = \bar{\alpha}_i \mu_i + \bar{\alpha}_j \mu_j, \tag{4.31}$$

$$\Gamma_{ij} = \bar{\alpha}_i \Gamma_i + \bar{\alpha}_j \Gamma_j + \bar{\alpha}_i \bar{\alpha}_j \left(\mu_i - \mu_j\right)\left(\mu_i - \mu_j\right)^T, \tag{4.32}$$

donde $\bar{\alpha}_i = \alpha_i/(\alpha_i + \alpha_j)$ y $\bar{\alpha}_j = \alpha_j/(\alpha_i + \alpha_j)$. La función $\mathcal{M}\{\cdot,\cdot\}$ combina dos componentes gaussianas i y j $(i \neq j)$, minimizando la función de disimilitud $\mathcal{D}(i,j)$ definida como:

$$\mathcal{D}(i,j) = \frac{1}{2}\left[\alpha_{ij} \log\det\left\{\Gamma_{ij}\right\} - \alpha_i \log\det\left\{\Gamma_i\right\} - \alpha_j \log\det\left\{\Gamma_j\right\}\right], \tag{4.33}$$

donde $\mathcal{D}(i,j)$ satisface que $\mathcal{D}(i,j) = \mathcal{D}(j,i)$ y $\mathcal{D}(i,i) = 0$. En este trabajo usamos el algoritmo propuesto en (Balenzuela et al., 2019), el cual está basado en el método de (Runnalls, 2007) con una ligera modificación para incluir un umbral ε definido por el usuario que satisface $\mathcal{D}(i,j) < \varepsilon$. En el Algoritmo 1 presentamos el método de reducción de sumas de gaussianas usado en este trabajo (Balenzuela et al., 2019).

Algorithm 1: Algoritmo para la reducción de una suma de gaussianas.

Input: Mínimo y máximo número de componentes gaussianas LI y LS, respectivamente, que tendrá la versión reducida, con $LI > 0$ y $LI \leq LS$ números enteros. La información de la suma de gaussianas original, es decir, el conjunto de parámetros $\mathcal{G} = \{\alpha_i, \mu_i, \Gamma_i\}_{i=1}^{H}$.

Establecer $k = N$. Definir el conjunto de índices $\mathcal{I} = \{1, \ldots, H\}$. Calcular la matriz $\mathcal{D}(i,j)$ usando (4.33) para $i \in \mathcal{I}$ y $j \in \mathcal{I}$, definiendo $\mathcal{D}(i,j) = \infty$ si $i \geq j$.

while $k > LS$ **or** $(k > LI$ **and** $\min_{i,j} \mathcal{D}(i,j) \leq \varepsilon)$ **do**

Encontrar $(i^*, j^*) = \arg\min_{i,j} \mathcal{D}(i,j)$.

Combinar las componentes (i^*, j^*) usando (4.29) y reemplazar la i^*th componente, es decir

$$(\alpha_{i^*}, \mu_{i^*}, \Gamma_{i^*}) \leftarrow \left(\alpha_{i^*j^*}, \mu_{i^*j^*}, \Gamma_{i^*j^*}\right).$$

Eliminar la j^*th componente gaussiana del conjunto $\mathcal{G}$ tal que:

$$\mathcal{G} \leftarrow \mathcal{G} \backslash \left\{(\alpha_{j^*}, \mu_{j^*}, \Gamma_{j^*})\right\}.$$

Eliminar la j^*th fila y columna de la matriz $\mathcal{D}$ tal que:

$$\mathcal{D} \leftarrow \mathcal{D} \backslash \{j^*\text{th fila}, j^*\text{th columna}\}.$$

Eliminar el índice j^* del conjunto $\mathcal{I}$ de tal manera de obtener:

$$\mathcal{I} \leftarrow \mathcal{I} \backslash \{j^*\}.$$

Recalcular la matriz $\mathcal{D}(i,j)$ usando (4.33) con $i \in \mathcal{I}$ y $j \in \mathcal{I}$, definiendo $\mathcal{D}(i,j) = \infty$ si $i \geq j$. Notar además que solo la i^*-ésima fila y columna de la matriz $\mathcal{D}(i,j)$ debe ser modificada.

end

Output: La información de la suma de gaussianas reducidas contenida en el conjunto $\mathcal{G}$.

Observación 15. *La notación $\mathcal{V} \leftarrow \mathcal{V} \backslash \{\lambda\}$ indica que se debe formar un nuevo conjunto a partir de $\mathcal{V}$ que no posea el elemento λ. A este nuevo conjunto se le da el mismo nombre del original, es decir, $\mathcal{V}$.*

Observación 16. *El Algoritmo 2 se resumen los pasos para implementar el filtro suma de gaussianas para sistemas Hammerstein-Wiener. El algoritmo GSF requiere la información la PDF del estado inicial $p(x_1)$, es decir, la media μ_1 y la matriz de covarianza P_1. Requiere además los L puntos de la cuadratura de Gauss-Legendre y el número M de trozos estrictamente monótonos en que se divide la función $g(\cdot)$. Recuerde que para el caso del Teorema 6 en donde se trata una función estrictamente monótona a trozos, $K = LM$. Para los casos en los Teoremas 9, 10 y 12 en donde se tratan las funciones cuantizador binario, saturación, y zona muerta, $M = 1$, por lo cual $K = L$ corresponde al número de puntos de la cuadratura de Gauss-Legendre. Por otra parte, note que la reducción de sumas de gaussianas puede realizarse tanto en la etapa de corrección como en la etapa de predicción. Sin embargo, para reducir el tiempo de cómputo se recomienda realizar la reducción de gaussianas en la etapa de corrección de tal manera que la etapa de predicción siempre trabaje con un número reducido de gaussianas*

Algorithm 2: GSF para sistemas Hammerstein-Wiener.

Input: La PDF del estado inicial $p(x_1)$, e.g. $\mathcal{S}_{1|0} = 1$, $\delta_{1|0} = 1$, $\hat{x}_{1|0} = \mu_1$, $\Gamma_{1|0} = P_1$. Los puntos de la cuadratura de Gauss-Legendre $\{\omega_\tau, \psi_\tau\}_{\tau=1}^{L}$. El número M de trozos estrictamente monótonos en que se divide $g(\cdot)$, ver la observación 16.
for $t = 1$ **to** N **do**
  Calcular β_j y ξ_t^j de acuerdo a los Teoremas 6, 9, 10 y 12 según la función no lineal de la salida del sistema.
  Etapa de corrección:
  Establecer $\mathcal{S}_{t|t} = K\mathcal{S}_{t|t-1}$.
  for $k = 1$ **to** $\mathcal{S}_{t|t-1}$ **do**
    for $j = 1$ **to** K **do**
      Calcular el índice $\ell = (k-1)K + j$.
      Calcular $\delta_{t|t}^{\ell}$,$\hat{x}_{t|t}^{\ell}$, y $\Sigma_{t|t}^{\ell}$ de acuerdo al Teorema 14.
    end
  end
  Calcular la estimación de estado dada en (1.7) y la matriz de covarianza del error de estimación dada en (1.8).
  Realizar la reducción de gaussianas usando el Algoritmo 1 con el conjunto de datos $\delta_{t|t}^{\ell}$,$\hat{x}_{t|t}^{\ell}$, y $\Sigma_{t|t}^{\ell}$ para obtener la versión reducida de $p(x_t|y_{1:t})$.
  Etapa de predicción:
  Establecer $\mathcal{S}_{t+1|t} = \mathcal{S}_{t|t}$.
  for $\ell = 1$ **to** $\mathcal{S}_{t+1|t}$ **do**
    Calcular $\delta_{t+1|t}^{\ell}$,$\hat{x}_{t+1|t}^{\ell}$, y $\Sigma_{t+1|t}^{\ell}$ de acuerdo al Teorema 14.
  end
end
Output: La estimación de estado en (1.7) y la matriz de covarianza del error de estimación dada en (1.8). La PDF $p(x_t|y_{1:t})$ y la PDF de predicción $p(x_{t+1}|y_{1:t})$ para $t = 1, \ldots, N$.

4.2.2 Estimador de estados y error de estimación

Una vez calculada la PDF de filtraje $p(x_t|y_{1:t})$ en (4.4) que viene dada como una suma de gaussianas, el estimador de estados dado en (1.7) y la matriz de covarianza del error de estimación en (1.8) se pueden calcular como sigue:

$$\hat{x}_{t|t} = \sum_{\ell=1}^{\mathcal{S}_{t|t}} \delta_{t|t}^{\ell} \hat{x}_{t|t}^{\ell}, \tag{4.34}$$

$$\Gamma_{t|t} = \sum_{\ell=1}^{\mathcal{S}_{t|t}} \delta_{t|t}^{\ell} \left[\Gamma_{t|t}^{\ell} + (\hat{x}_{t|t}^{\ell} - \hat{x}_{t|t})(\hat{x}_{t|t}^{\ell} - \hat{x}_{t|t})^T \right]. \tag{4.35}$$

5

Ejemplos Numéricos

En esta sección presentamos ejemplos numéricos para analizar el desempeño del algoritmo de filtraje propuesto, al cual denominamos con las siglas GSF. Comparamos nuestra propuesta con los filtros de Kalman: estándar (KF), extendido (EKF), en cuadratura (QKF), cuantizado (QTKF) y con el filtro de partículas (PF), ver el apéndice A.1. Consideramos además que las PDFs filtradas verdaderas (GT) se obtienen usando el filtro de partículas con 20000 partículas, en donde $p(y_t|x_t)$ se calcula usando las ecuaciones integrales en (3.20), (3.39), (3.46), (3.58), y (3.70) y el método de integración de Monte Carlo. Usamos $L = 10$ puntos de la cuadratura de Gauss-Legendre para aproximar la PDF $p(y_t|x_t)$, 300 partículas para el algoritmo del filtro de partículas y 6 puntos sigma para el filtro de Kalman en cuadratura. Simulamos $N = 100$ datos de entrada y salida. A continuación se presenta el cálculo de $\gamma_{it}(z_t)$ y $\phi_{it}(z_t)$ dadas en (3.21) y (3.22), respectivamente, para algunos ejemplos usados en las simulaciones:

Función afín: Considere la función afín dada por $z_t = \mathbb{A}r_t + b$ donde $\mathbb{A}$ y b son constantes conocidas. Esta función es estrictamente monótona en toda la recta real, por lo tanto $M = 1$ y

$$\gamma_{1t}(z_t) = \frac{(z_t - b)}{\mathbb{A}}, \qquad \phi_{1t}(z_t) = \frac{1}{\mathbb{A}}, \tag{5.1}$$

Función valor absoluto: Considere la función valor absoluto dada por $z_t = a|r_t - b| + c$ donde a, b, c son constantes conocidas, entonces z_t se puede escribir como

$$z_t = \begin{cases} a\,(r_t - b) + c & \text{si} \quad r_t \geq b, \\ -a\,(r_t - b) + c & \text{si} \quad r_t < b, \end{cases} \tag{5.2}$$

entonces se tienen dos intervalos en donde la función es estrictamente monótona, por lo tanto $M = 2$ y

$$\gamma_{1t}(z_t) = \frac{(z_t + ab - c)}{a}, \qquad \phi_{1t}(z_t) = \frac{1}{a}, \tag{5.3}$$

$$\gamma_{2t}(z_t) = -\frac{(z_t + ab - c)}{a}, \qquad \phi_{2t}(z_t) = \frac{1}{a}. \tag{5.4}$$

Función cuadrática: Considere la función cuadrática dada por $z_t = r_t^2$. Esta función tiene dos intervalos en donde es estrictamente monótona, por lo tanto $M = 2$ y

$$\gamma_{1t}(z_t) = \sqrt{z_t}, \qquad \phi_{1t}(z_t) = 0.5/\sqrt{z}, \tag{5.5}$$

$$\gamma_{2t}(z_t) = -\sqrt{z_t}, \qquad \phi_{2t}(z_t) = 0.5/\sqrt{z}. \tag{5.6}$$

Función cúbica: Considere la función cúbica dada por $z_t = r_t^3$. Esta función es estrictamente monótona con toda la recta real, por lo tanto $M = 1$ y

$$\gamma_{1t}(z_t) = \sqrt[3]{z_t}, \qquad \phi_{1t}(z_t) = \frac{1}{3\sqrt[3]{z^2}}. \tag{5.7}$$

Función definida a trozos: Considere la función definida a trozos

$$z_t = \begin{cases} |r_t| & \text{if} \quad r_t \leq 0, \\ r_t^2 & \text{if} \quad r_t > 0. \end{cases} \tag{5.8}$$

Claramente esta función tiene dos intervalos en donde es estrictamente monótona, por lo tanto $M = 2$ y

$$\gamma_{1t}(z_t) = \sqrt{z_t}, \qquad \phi_{1t}(z_t) = 0.5/\sqrt{z_t}, \tag{5.9}$$
$$\gamma_{2t}(z_t) = -z_t, \qquad \phi_{2t}(z_t) = 1. \tag{5.10}$$

Contribución

La principal contribución de este capítulo es analizar el rendimiento del algoritmo propuesto comparado con los distintos enfoques que existen en la literatura. El análisis se efectuá en términos de la precisión en la estimación de los estados del sistema así como en términos de la carga computacional.

5.1 Ejemplo 1: función cuadrática

En este ejemplo consideramos el siguiente sistema Hammerstein-Wiener, donde la no linealidad de la salida, $g(\cdot)$, es una función cuadrática:

$$\begin{aligned} x_{t+1} &= 0.9x_t + 2.5\arctan(u_t) + w_t, \\ r_t &= 1.1x_t + 1.5\arctan(u_t) + v_t, \\ z_t &= r_t^2, \\ y_t &= z_t + \eta_t, \end{aligned} \tag{5.11}$$

donde $w_t \sim \mathcal{N}(w_t; 0, 1)$, $v_t \sim \mathcal{N}(v_t; 0, 0.5)$, y $\eta_t \sim \mathcal{N}(\eta_t; 0, 0.5)$. Además, consideramos que la señal de entrada u_t es muestreada de $\mathcal{N}(0, 2)$ y $p(x_1) = \mathcal{N}(x_1; 1, 0.1)$.

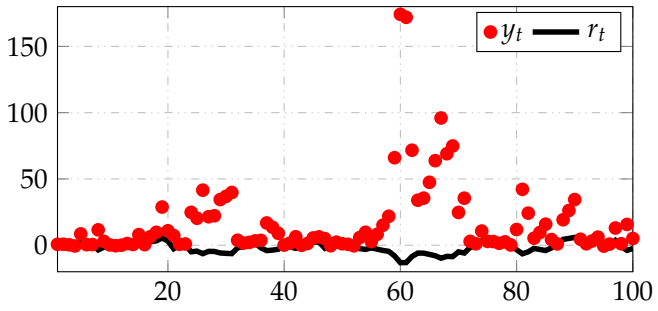


Figura 5.1: Ejemplo 1. Gráfica de la salida lineal r_t y salida no lineal y_t.

En la Figura 5.1 se muestra la salida lineal r_t antes de pasar por la función cuadrática y la salida y_t que corresponde a las mediciones de la salida no lineal. Note que en este caso la función cuadrática transforma en valores positivos y de considerable magnitud los valores de la salida no medible r_t.

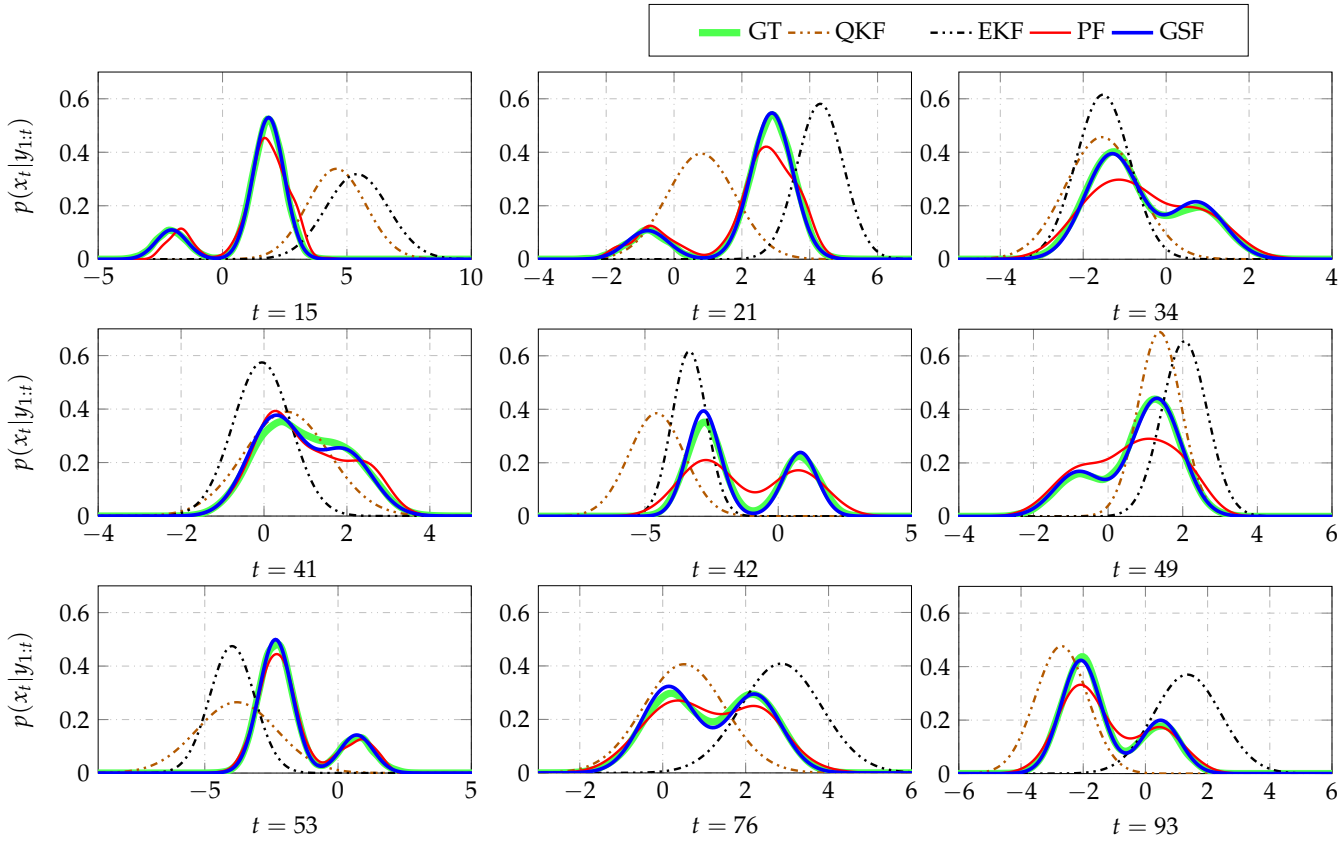


Figura 5.2: Ejemplo 1. PDF filtrada $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 10 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF.

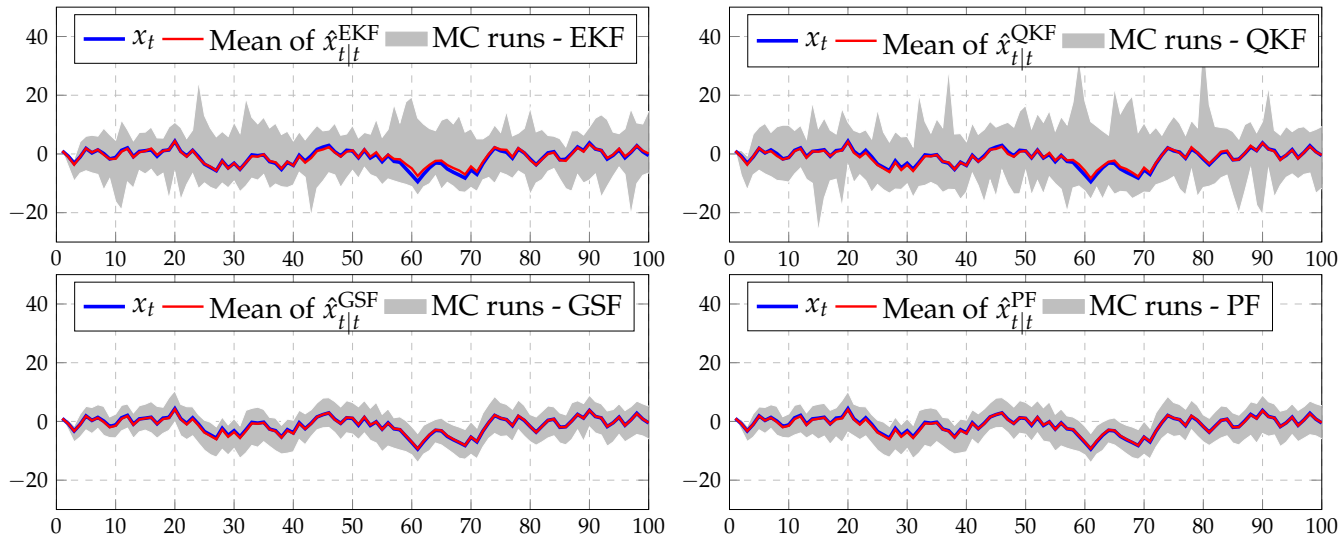


Figura 5.3: Ejemplo 1. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo. El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo.

En la Figura 5.2 se muestran las PDFs de filtraje para algunos instantes de tiempo. Se puede observar

que las PDFs obtenidas usando los algoritmos EKF, QKF son diferentes de las PDFs verdaderas GT. En cambio, los resultados obtenidos con el algoritmo propuesto se ajustan con mucha precisión a las PDFs GT. Se observa además que los resultados obtenidos con PF, al usar pocas partículas, produce un resultado poco ajustado. Sin embargo, es claro que al aumentar la cantidad de partículas los resultados mejoran, pero a un costo computacional elevado. En la Figura 5.3 se muestra el resultado de un análisis de Monte Carlo con 100 experimentos para la estimación del estado $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$. La región gris muestra el área donde se encuentran todas las estimaciones del estado y la línea roja muestra la media de las 100 estimaciones. En esta figura se puede observar que la variabilidad obtenida usando EKF, QKF es mucho más alta comparada con la variabilidad obtenida usando GSF y PF.

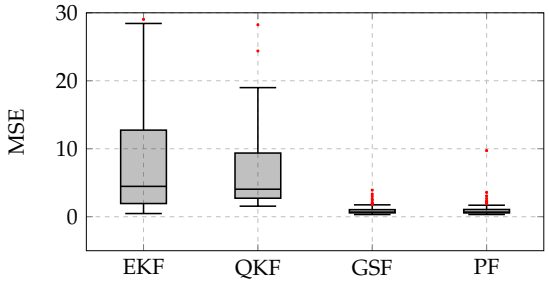


Figura 5.4: Ejemplo 1. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t.

En la Figura 5.4 se muestran las gráficas boxplot del error cuadrático medio entre la secuencia de estados estimados con cada algoritmo y el estado verdadero. Se observa cómo los filtros EKF y QKF producen resultados con alta varianza a diferencia de los algoritmos GSF y PF cuya varianza es baja. Además, PF produce más outliers que el filtro GSF.

5.2 Ejemplo 2: función definida a trozos

En este ejemplo consideramos el siguiente sistema Hammerstein-Wiener, donde la no linealidad de la salida, $g(\cdot)$, es una función definida a trozos como sigue:

$$\begin{aligned} x_{t+1} &= \begin{bmatrix} 0.9 & 0.1 \\ -0.1 & 0.7 \end{bmatrix} x_t + \begin{bmatrix} 2.5 \\ 1.0 \end{bmatrix} \sin(u_t) + w_t, & z_t &= \begin{cases} |r_t| & \text{if} \quad r_t \leq 0, \\ r_t^2 & \text{if} \quad r_t > 0, \end{cases} \\ r_t &= \begin{bmatrix} 1.0 & 1.0 \end{bmatrix} x_t + 2.1 \sin(u_t) + v_t, & y_t &= z_t + \eta_t, \end{aligned} \tag{5.12}$$

donde $w_t \sim \mathcal{N}\left(w_t; [0\ 0]^T, 0.5I_2\right)$, $v_t \sim \mathcal{N}\left(v_t; 0, 0.1\right)$, y $\eta_t \sim \mathcal{N}\left(\eta_t; 0, 0.2\right)$. Además, consideramos que la señal de entrada u_t es muestreada de $\mathcal{N}\left(0, 0.5\right)$, y $p(x_1) = \mathcal{N}(x_1; [1\ 1.5]^T, I_2)$, donde I_2 representa la matriz identidad de orden 2.

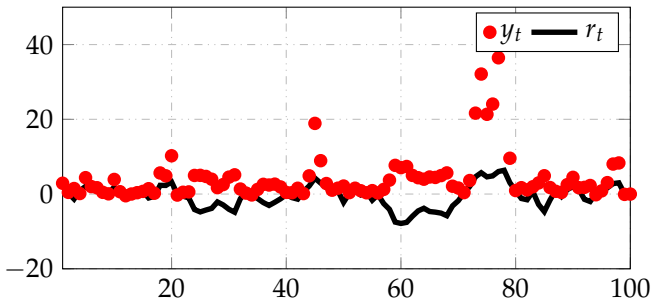


Figura 5.5: Ejemplo 2. Gráfica de la salida lineal r_t y salida no lineal y_t.

En la Figura 5.5 se muestra la salida lineal r_t antes de pasar por la función definida a trozos y la salida y_t que corresponde a las mediciones de la salida no lineal. Como en el ejemplo anterior, se observa el efecto de la no linealidad en la salida no medible r_t.

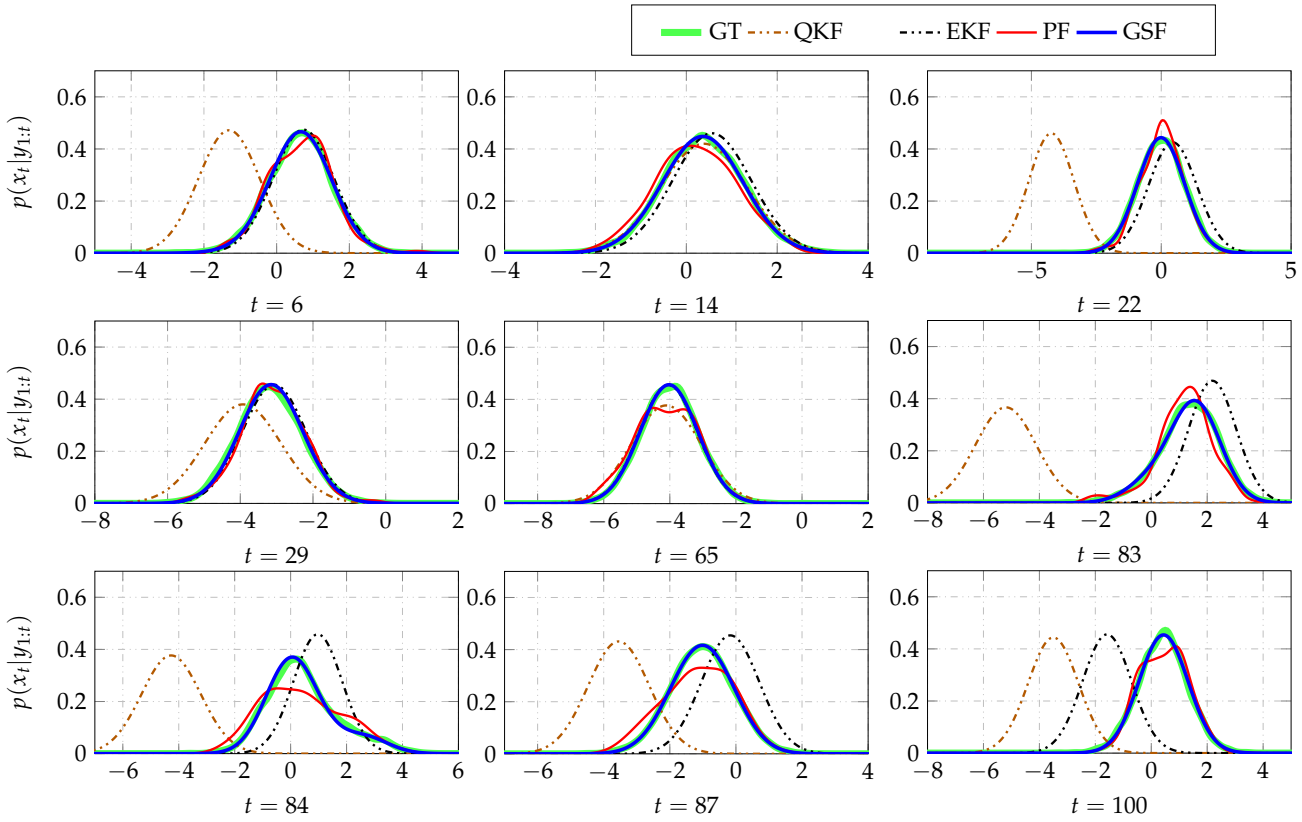


Figura 5.6: Ejemplo 2. PDF filtrada (marginal) $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 10 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF.

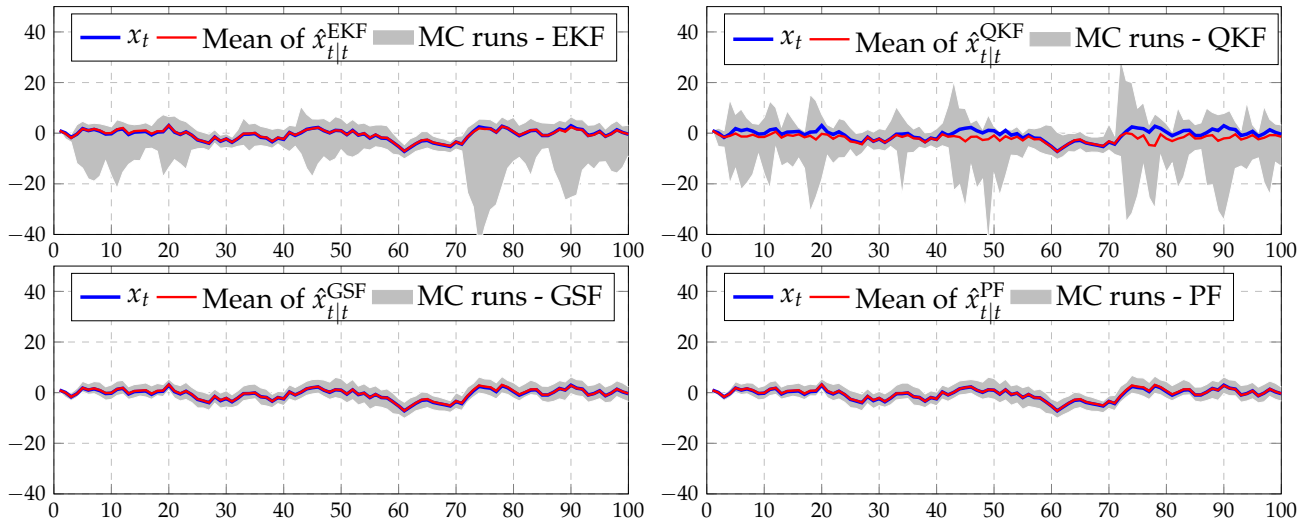


Figura 5.7: Ejemplo 2. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo (Gráfica del primer estado). El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo.

En la Figura 5.6 se muestran las PDFs de filtraje para algunos instantes de tiempo. Se puede observar que los resultados obtenidos usando los algoritmos EKF, QKF proveen una PDF diferente que la PDF GT, siendo el peor resultado el obtenido con QKF. No obstante, los resultados obtenidos con el algoritmo GSF se ajustan con mucha precisión a las PDFs GT. Así mismo, el resultado obtenido con pocas partículas es poco ajustado a la PDF GT pero es mejor que el resultado obtenido con EKF y QKF. En la Figura 5.7 se muestra el resultado de un análisis de Monte Carlo con 100 experimentos para la estimación del estado $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$. En esta figura se puede observar que la variabilidad obtenida usando el enfoque propuesto en este trabajo GSF, y el filtro de PF es considerablemente baja respecto a la variabilidad del EKF y QKF. Además, el filtro QKF produce una estimación del estado en media diferente al estado verdadero.

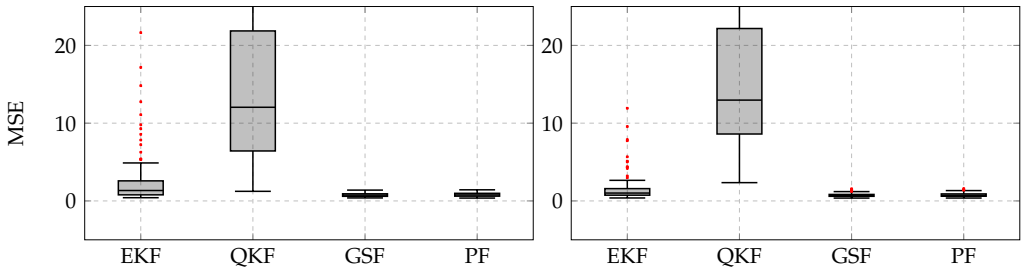


Figura 5.8: Ejemplo 2. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t.

En la Figura 5.8 se muestran las gráficas boxplot del error cuadrático medio entre la secuencia de estados estimados con cada algoritmo y el estado verdadero. Se observa como el filtro QKF produce resultados con la más alta varianza seguido del filtro EKF. En este caso, el MSE con los resultados del filtro GSF y PF son similares.

5.3 Ejemplo 3: función cúbica

En este ejemplo consideramos el siguiente sistema Hammerstein-Wiener

$$\begin{aligned} x_{t+1} &= \begin{bmatrix} 0.7 & 0 & 0.1 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} x_t + \begin{bmatrix} 2.0 \\ 1.5 \\ 1.0 \end{bmatrix} |u_t| + w_t, & z_t &= r_t^3, \\ r_t &= \begin{bmatrix} 1.0 & 1.2 & 0.2 \end{bmatrix} x_t + 1.8|u_t| + v_t, & y_t &= z_t + \eta_t, \end{aligned} \tag{5.13}$$

donde $w_t \sim \mathcal{N}\left(w_t; [0\ 0\ 0]^T, 0.1I_3\right)$, $v_t \sim \mathcal{N}\left(v_t; 0, 0.4\right)$, y $\eta_t \sim \mathcal{N}\left(\eta_t; 0, 0.3\right)$. Además, consideramos que la señal de entrada u_t es muestreada de $\mathcal{N}\left(0, 1.2\right)$, y $p(x_1) = \mathcal{N}(x_1; [1\ 1.5\ 0.9]^T, 0.1I_3)$, donde I_3 representa la matriz identidad de orden 3.

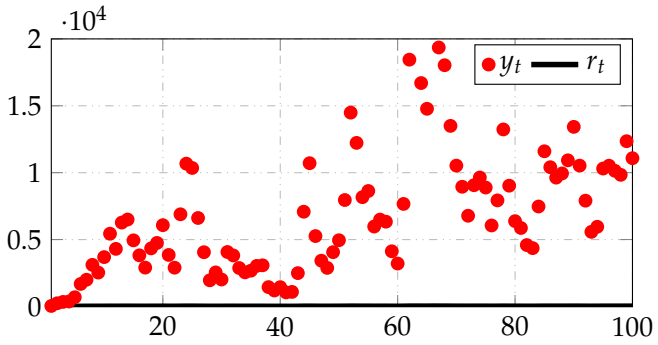


Figura 5.9: Ejemplo 3. Gráfica de la salida lineal r_t y salida no lineal y_t.

En la Figura 5.9 se muestra la salida lineal r_t antes de pasar por la función cúbica y la salida y_t que corresponde a las mediciones de la salida no lineal. En este ejemplo es más evidente el efecto de la no linealidad en la salida no medible r_t.

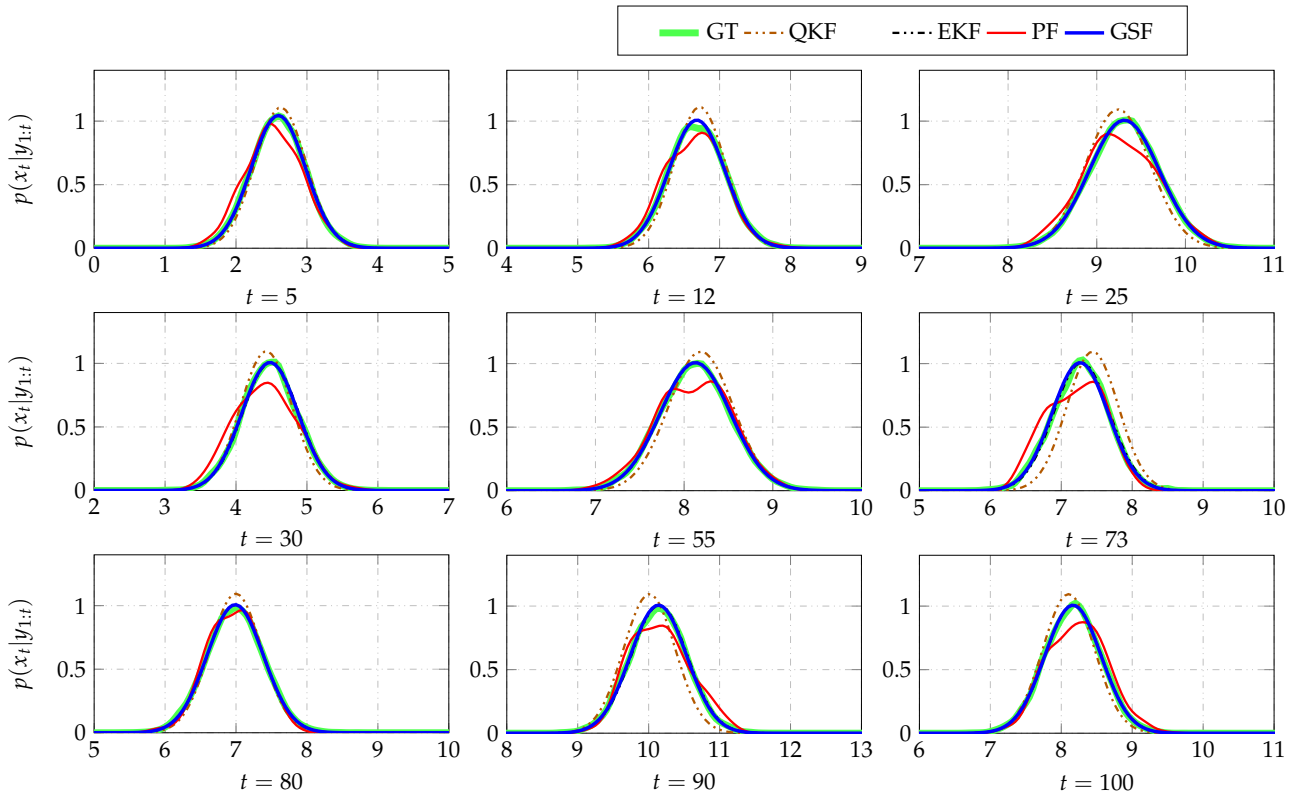


Figura 5.10: Ejemplo 3. PDF filtrada (marginal) $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 10 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF.

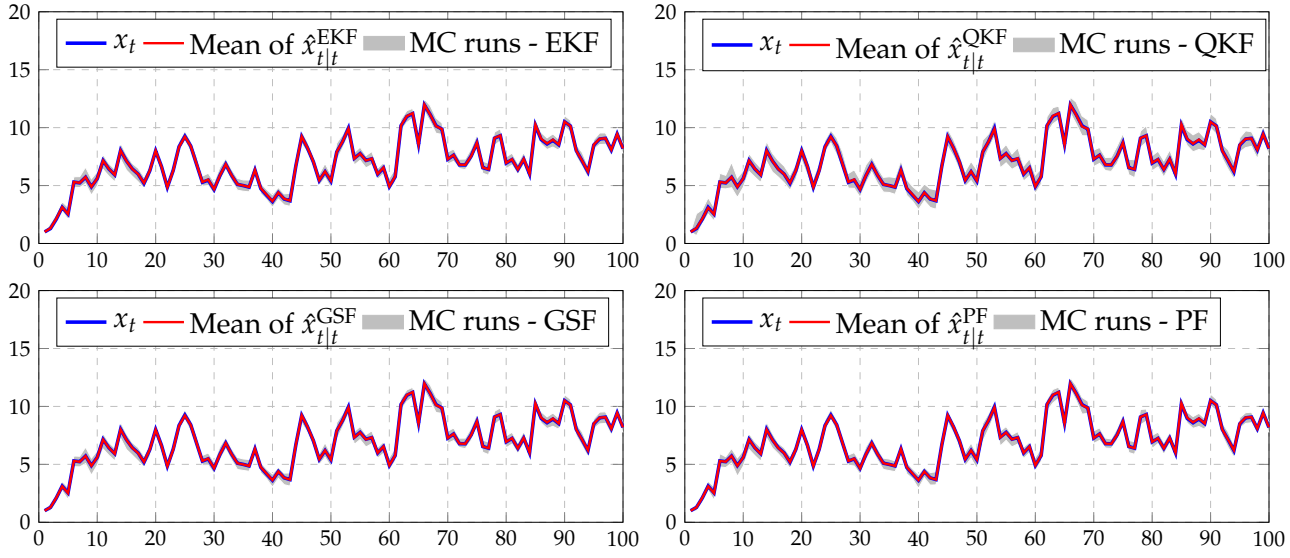


Figura 5.11: Ejemplo 3. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo (Gráfica del primer estado). El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo.

En la Figura 5.10 se muestran las PDFs de filtraje para algunos instantes de tiempo. En este ejemplo, la no linealidad de la salida es una función suave que es diferenciable en todo su dominio, entonces es de esperarse que los filtros EKF y QKF produzcan PDFs bien ajustadas a las PDFs verdaderas y similares a los resultados de los filtros GSF y PF. En la Figura 5.11 se muestra el resultado de un análisis de Monte Carlo con 100 experimentos para la estimación del estado $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$. En esta figura se puede observar que todos los filtros producen una variabilidad relativamente baja en la estimación del estado, aunque la más alta es obtenida por el filtro QKF.

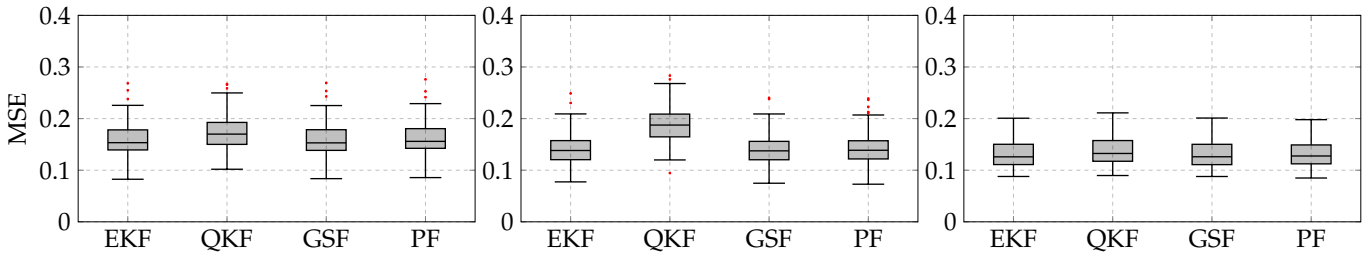


Figura 5.12: Ejemplo 3. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t.

En la Figura 5.12 se muestran las gráficas boxplot del error cuadrático medio entre la secuencia de estados estimados con cada algoritmo y el estado verdadero. Como se mencionó antes, en este ejemplo los resultados de los cuatro filtros son similares en términos de la variabilidad en la estimación del estado y con respecto al MSE entre las estimaciones y el valor verdadero.

5.4 Ejemplo 4: función cuantizador binario

En este ejemplo consideramos el siguiente sistema Hammerstein-Wiener, donde la no linealidad de la salida es una cuantizador binario

$$\begin{aligned} x_{t+1} &= 0.99x_t + 2.1\arctan(u_t) + w_t, \\ r_t &= 0.5x_t + 1.8\arctan(u_t) + v_t, \end{aligned} \qquad \begin{aligned} z_t &= \begin{cases} -2 & \text{if} \quad r_t < 2, \\ 3 & \text{if} \quad r_t \geq 2, \end{cases} \\ y_t &= z_t, \end{aligned} \tag{5.14}$$

donde $w_t \sim \mathcal{N}(w_t; 0, 0.1)$, $v_t \sim \mathcal{N}(v_t; 0, 0.05)$. Además, consideramos que la señal de entrada u_t es muestreada de $\mathcal{N}(0,1)$ y $p(x_1) = \mathcal{N}(x_1; 1, 1)$. En este ejemplo se usa $L = 100$.

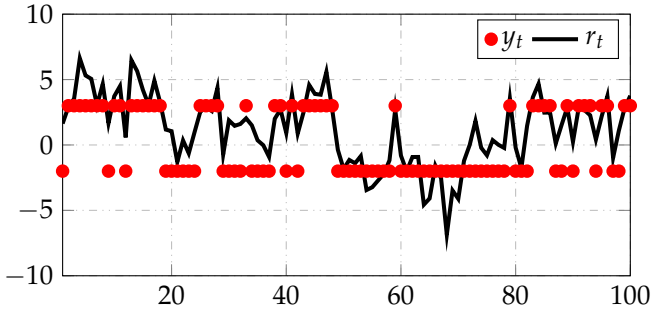


Figura 5.13: Ejemplo 4. Gráfica de la salida lineal r_t y salida no lineal y_t.

Note en la Figura 5.13 como la salida lineal r_t se mapea a los valores 3 y -2 cuando r_t está por arriba o por abajo del umbral 2. En este ejemplo se eligieron valores ν_1 y ν_2 al azar para demostrar la generalidad del método para tratar con valores binarios, aunque los valores más típicos suelen ser $\nu_1 = 0$ y $\nu_2 = 1$.

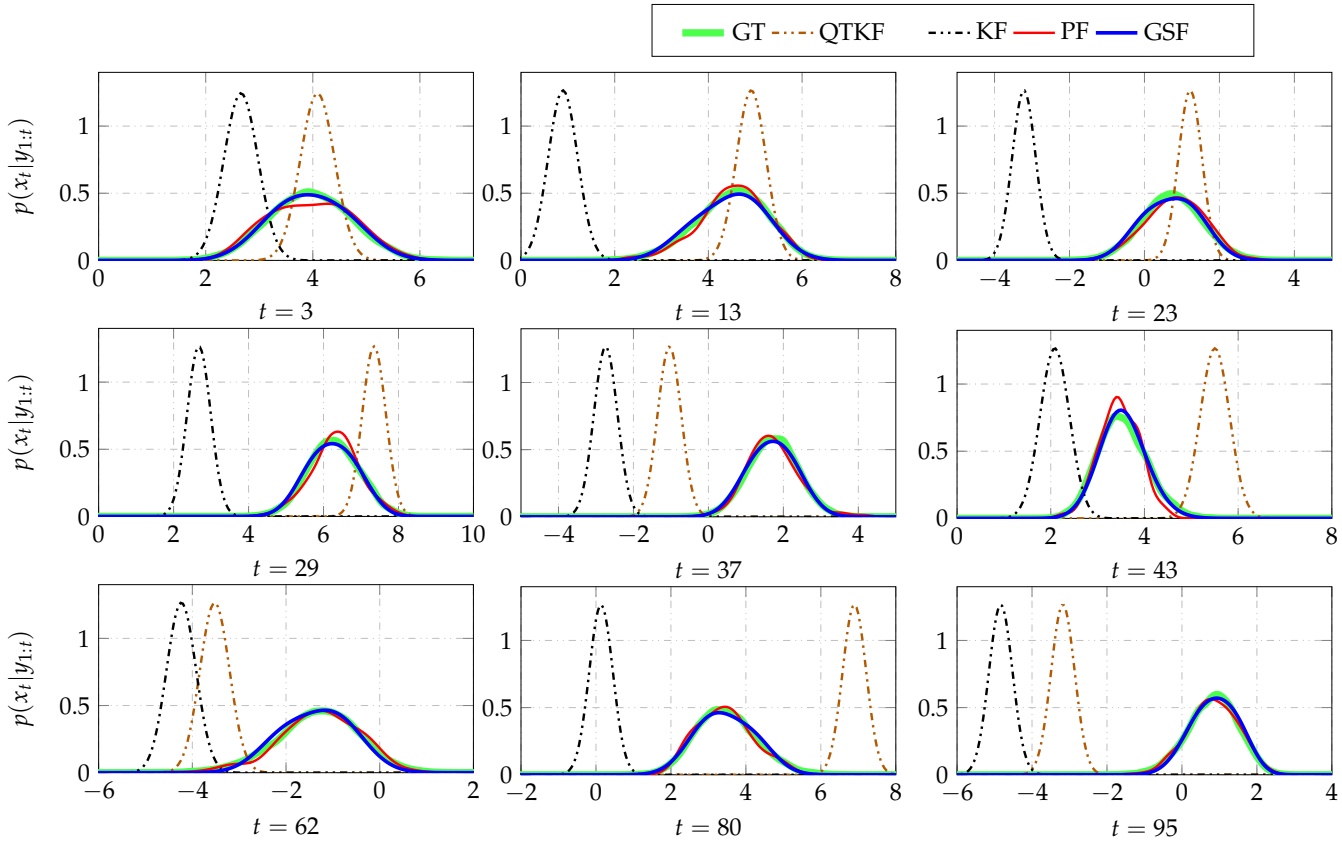


Figura 5.14: Ejemplo 4. PDF filtrada $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 100 componentes gaussianas en el filtro GSF y 300 partículas en el filtro PF.

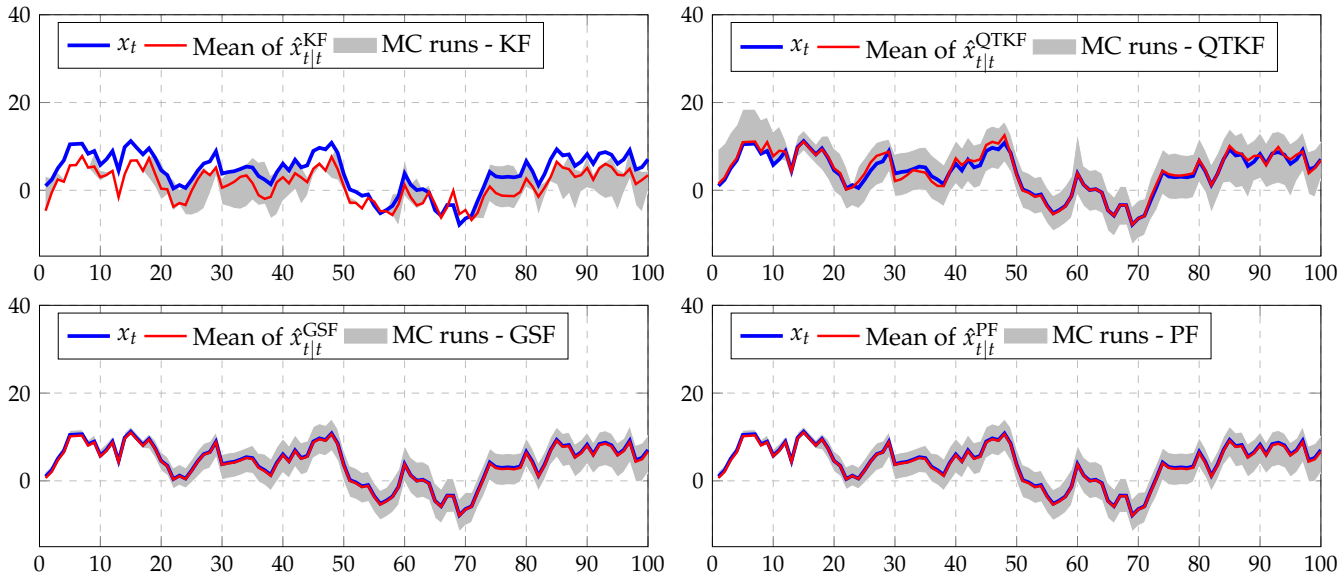


Figura 5.15: Ejemplo 4. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo. El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo.

En la Figura 5.14 se muestran las PDFs de filtraje para algunos instantes de tiempo. La cuantización binaria es uno de los casos más complejos de tratar por la gran pérdida de información que ocurre en la salida r_t, lo que se evidencia en las PDFs obtenidas con los filtros KF y QTKF, las mismas que son diferentes y están alejadas de las PDFs verdaderas. En cambio las PDFs obtenidas con los filtro GSF y PF se ajustan con precisión, en menor grado las obtenidas con el filtro PF. En la Figura 5.15 se muestra el resultado de un análisis de Monte Carlo con 100 experimentos para la estimación del estado $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$. En esta figura se puede observar que la variabilidad de los filtros KF y QTKF es considerablemente alta y además la media de todas las estimaciones difiere en gran medida del estado verdadero. Por otro lado, los filtros GSF y PF producen estimaciones del estado similares con poca variabilidad.

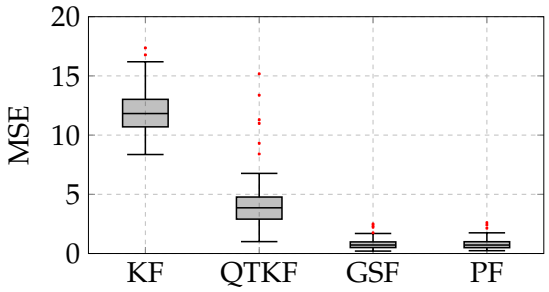


Figura 5.16: Ejemplo 4. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t.

En la Figura 5.16 se muestran las gráficas boxplot del error cuadrático medio entre la secuencia de estados estimados con cada algoritmo y el estado verdadero. En esta gráfica se mantiene el comportamiento de los filtros. El filtro KF produce el mayor MSE, seguido de QTKF, PF y GSF.

5.5 Ejemplo 5: función saturador

En este ejemplo consideramos el siguiente sistema Hammerstein-Wiener, donde la no linealidad de la salida es un saturador como se muestra a continuación:

$$\begin{aligned} x_{t+1} &= \begin{bmatrix} 0.9 & 0.1 \\ -0.3 & 0.7 \end{bmatrix} x_t + \begin{bmatrix} 1.7 \\ 0.7 \end{bmatrix} |u_t| + w_t, \\ r_t &= \begin{bmatrix} 0.9 & 1.2 \end{bmatrix} x_t + 1.2|u_t| + v_t, \end{aligned} \qquad \begin{aligned} z_t &= \begin{cases} -3 & \text{if} \quad r_t < -3, \\ r_t & \text{if} \quad -3 \leq r_t < 3, \\ 3 & \text{if} \quad r_t \geq 3, \end{cases} \\ y_t &= z_t, \end{aligned} \tag{5.15}$$

donde $w_t \sim \mathcal{N}\left(w_t; [0\ 0]^T, 0.5I_2\right)$, $v_t \sim \mathcal{N}\left(v_t; 0, 0.1\right)$, y $\eta_t \sim \mathcal{N}\left(\eta_t; 0, 0.00001\right)$. Además, consideramos que la señal de entrada u_t es muestreada de $\mathcal{N}\left(0, 2.5\right)$, y $p(x_1) = \mathcal{N}(x_1; [1\ 1.5]^T, I_2)$, donde I_2 representa la matriz identidad de orden 2. En este ejemplo se usa $L = 50$.

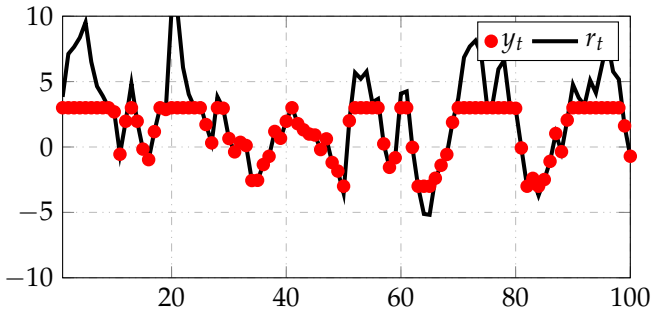


Figura 5.17: Ejemplo 5. Gráfica de la salida lineal r_t y salida no lineal y_t.

En la Figura 5.17 se observa que para este ejemplo la salida r_t está en la mayoría de veces encima de -3, por lo que la salida y_t muestra solo saturación en el limité superior correspondiente a 3.

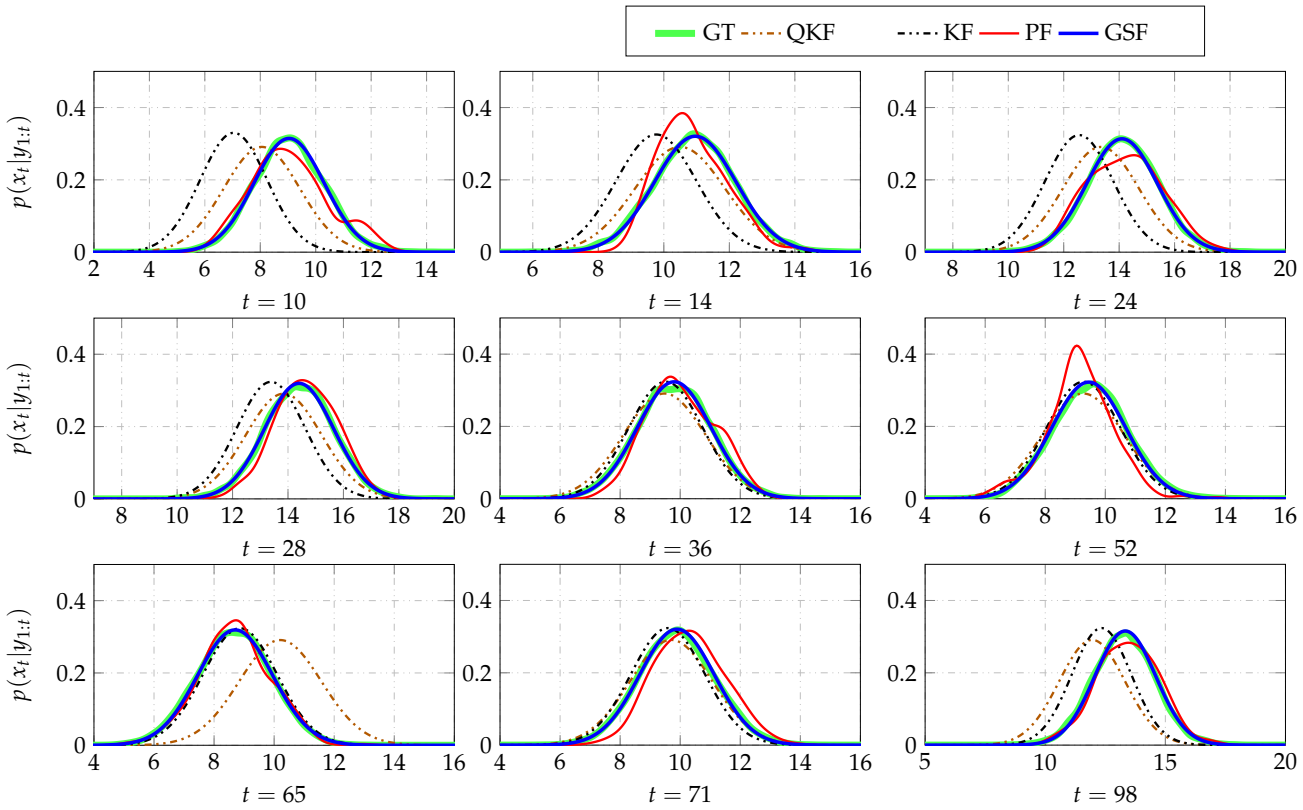


Figura 5.18: Ejemplo 5. PDF filtrada (marginal) $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 50 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF.

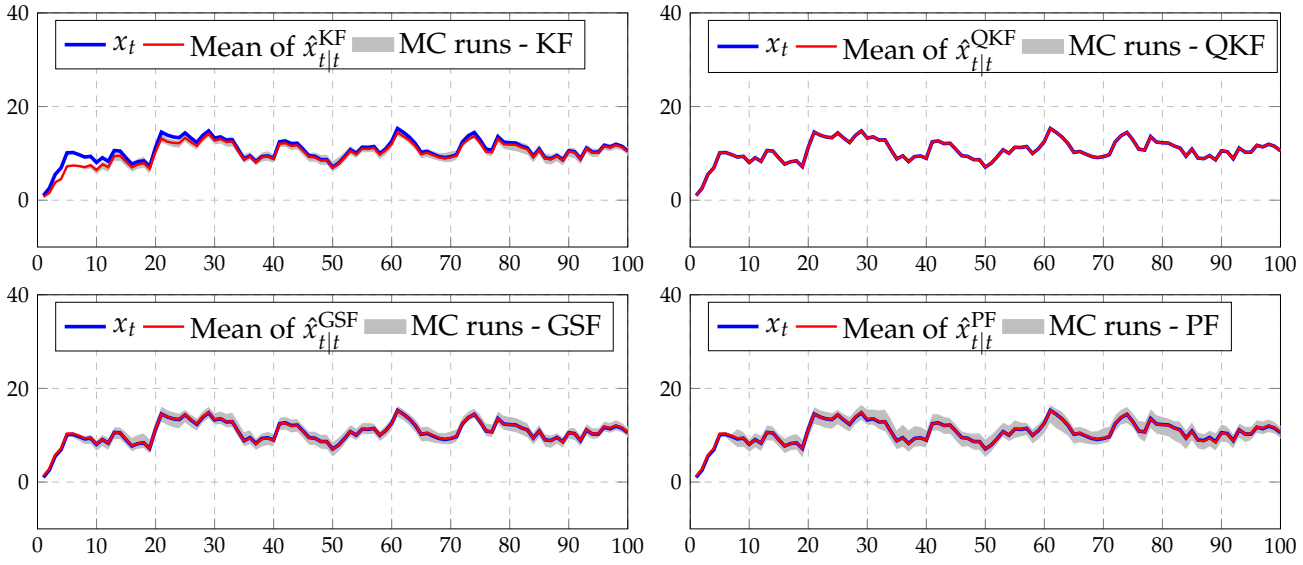


Figura 5.19: Ejemplo 5. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo (Gráfica del primer estado). El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo.

En la Figura 5.18 se muestran las PDFs de filtraje para algunos instantes de tiempo. La función saturación puede interpretarse como un cuantizador binario y una zona lineal. Por esta razón, cuando la salida y_t se encuentra en la zona lineal (en este ejemplo entre -3 y 3) los resultados de todos los filtro están muy cerca de la PDF verdadera, como ocurre en $t = 36$ y $t = 71$. Sin embargo, cuando la salida y_t está saturada en los valores máximos o mínimos, las PDFs obtenidas con los filtros KF, QKF son diferentes de las PDFs verdaderas, mientras que los resultados del filtro de partículas son mejores. Adicionalmente, las PDFs obtenidas con el filtro GSF son las que mejor se ajustan a las PDFs verdaderas. En la Figura 5.19 se muestra el resultado de un análisis de Monte Carlo con 100 experimentos para la estimación del estado $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$. En esta figura se puede observar que las variabilidades de los filtros KF y PF es mas alta que la variabilidad de los resultados del filtro GSF, seguido por el filtro QKF.

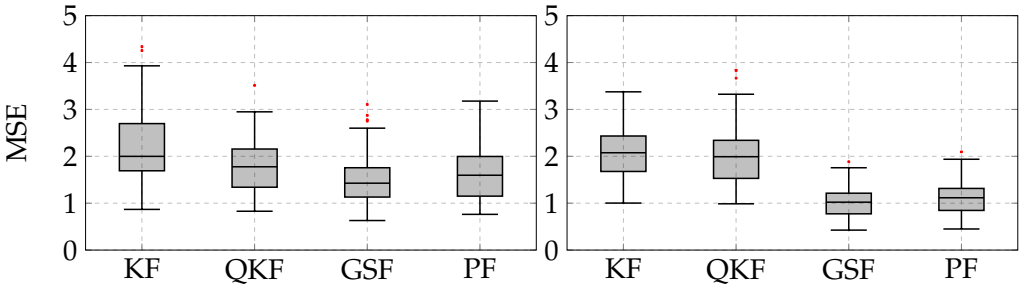


Figura 5.20: Ejemplo 5. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t.

En la Figura 5.20 se muestran las gráficas boxplot del error cuadrático medio entre la secuencia de estados estimados con cada algoritmo y el estado verdadero. En esta gráfica se evidencia, en términos de la mediana, que los mejores resultados se obtienen usando el filtro GSF seguido por los filtros PF, QKF, y KF.

5.6 Ejemplo 6: función zona muerta

En este ejemplo consideramos el siguiente sistema Hammerstein-Wiener, donde la no linealidad es una función zona muerta

$$
\begin{aligned}
x_{t+1} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.5 & -1 & -0.7 \end{bmatrix} x_t + \begin{bmatrix} 2.2 \\ 0.8 \\ 0.6 \end{bmatrix} \sinh(u_t) + w_t, & z_t &= \begin{cases} r_t + 3 & \text{if} \quad r_t < -3, \\ 0 & \text{if} \quad -3 \leq r_t < 3, \\ r_t - 3 & \text{if} \quad r_t \geq 3, \end{cases} \\
r_t &= \begin{bmatrix} 1.0 & 1.0 & 0.75 \end{bmatrix} x_t + 0.1 \sinh(u_t) + v_t, & y_t &= z_t,
\end{aligned}
\tag{5.16}
$$

donde $w_t \sim \mathcal{N}\left(w_t; [0\ 0\ 0]^T, 0.1 I_3\right)$, $v_t \sim \mathcal{N}\left(v_t; 0, 0.4\right)$, y $\eta_t \sim \mathcal{N}\left(\eta_t; 0, 0.0001\right)$. Además, consideramos que la señal de entrada u_t es muestreada de $\mathcal{N}\left(0, 0.5\right)$, y $p(x_1) = \mathcal{N}(x_1; [1\ 1.5\ 0.9]^T, I_3)$, donde I_3 representa la matriz identidad de orden 3.

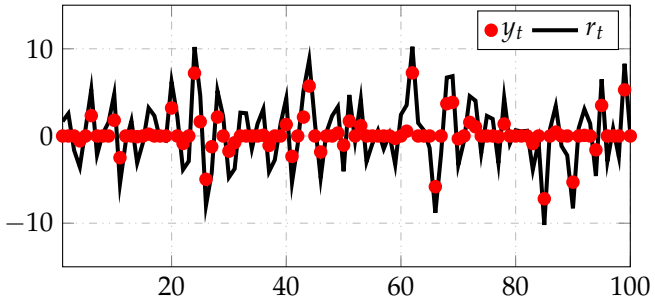


Figura 5.21: Ejemplo 6. Gráfica de la salida lineal r_t y salida no lineal y_t.

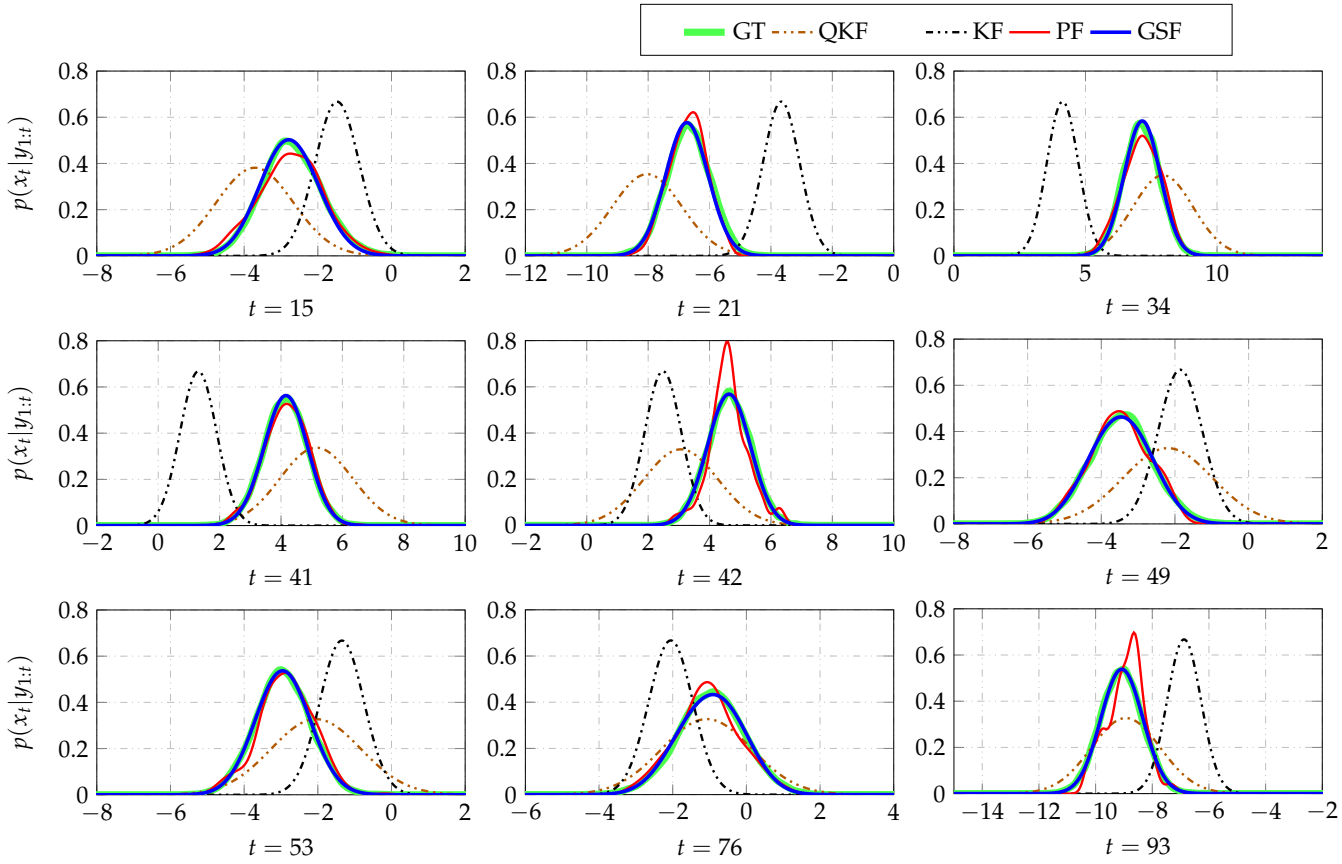


Figura 5.22: Ejemplo 6. PDF filtrada (marginal) $p(x_t|y_{1:t})$ para algunos instantes de tiempo. Se usaron 10 componentes gaussianas en el filtro GSF, 6 puntos sigma en el filtro QKF y 300 partículas en el filtro PF.

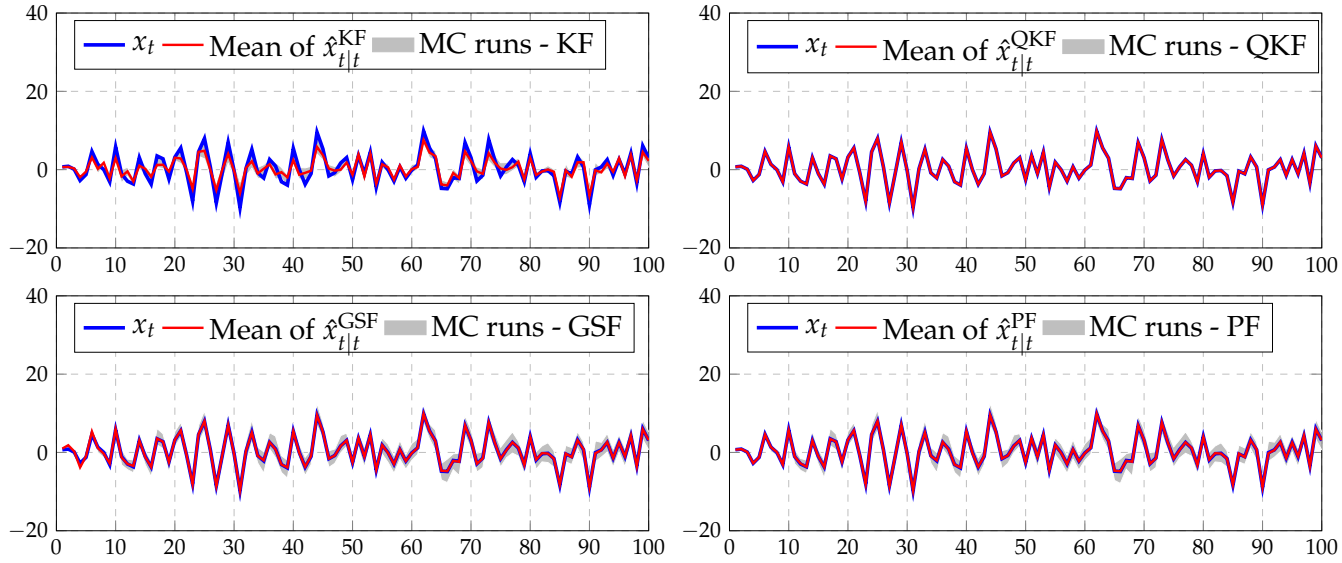


Figura 5.23: Ejemplo 6. Estimación del estado del sistema $\hat{x}_{t|t} = \mathbb{E}\{x_t|y_{1:t}\}$ para 100 experimentos de Monte Carlo (Gráfica del primer estado). El área gris representa el área donde se encuentran todas las estimaciones de la secuencia de estados del sistema. La línea azul representa el estado verdadero, y la línea roja representa la media de los 100 experimentos de Monte Carlo.

Al igual que en los dos ejemplos anteriores, se obtienen resultados muy precisos usando el algoritmo propuesto GSF, evidenciando su diferencia con los obtenidos empleando los otros enfoques objeto de comparación.

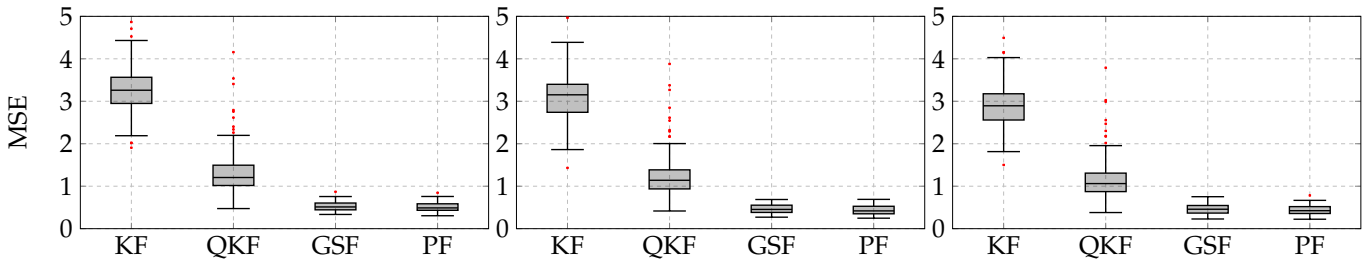


Figura 5.24: Ejemplo 6. MSE entre la secuencia de estados estimados $\hat{x}_{t|t}$ con cada algoritmo y el estado verdadero x_t.

En la Tabla 5.1 se resume la media estadística del tiempo de cómputo de cada algoritmo correspondiente a los 100 experimentos de Monte Carlo realizados. Para los ejemplos 1, 2, 3 y 6 se observa que el tiempo de cómputo del algoritmo propuesto GSF, al no variar la cantidad de componentes ($L = 10$) en la aproximación de $p(y_t|x_t)$, crece lentamente al aumentar el orden del sistema y se mantiene en valores por debajo del tiempo de cómputo del filtro PF que usa un número reducido de partículas (300). A si mismo, se observa que en el ejemplo 3 el tiempo de cómputo del algoritmo QKF se equipara con el de GFS (que ejecuta además el algoritmo GSR). Por otro lado, en el ejemplo 4 donde la no linealidad es el cuantizador binario, se obtiene un tiempo de cómputo alto para el filtro GSF comparado con el filtro PF. En este caso es normal este aumento en el tiempo de cómputo ya que el cuantizador es el caso de cuantización más demandante, por lo que se hizo necesario aumentar la cantidad de componentes ($L = 100$) en la aproximación de $p(y_t|x_t)$ para mejorar las estimaciones. Este aumento produce un aumento en la complejidad del algoritmo de reducción de sumas de Gausianas; esto debido a la gran cantidad de componentes por reducir. En el ejemplo 5, en donde la no linealidad es la función saturación, que como se indicó anteriormente se puede tratar con una mezcla entre un cuantizador binario y una función afín, se observa un comportamiento similar al caso anterior, con una reducción significativa del tiempo de cómputo. Esto se debe a que se han usado ($L = 50$) componentes en la aproximación de $p(y_t|x_t)$.

Tabla 5.1: Tiempo medio (segundos) del análisis de Monte Carlo para todos los ejemplos.

Algoritmo	Ejemplo 1	Ejemplo 2	Ejemplo 3	Ejemplo 4	Ejemplo 5	Ejemplo 6
KF	NA	NA	NA	0.0021	0.0023	0.0023
EKF	0.0744	0.1254	0.1401	NA	NA	NA
QKF	0.0513	0.2216	0.4905	NA	1.0260	6.1401
QTKF	NA	NA	NA	0.0021	NA	NA
GSF	0.4141	0.4660	0.5089	3.3541	1.0555	0.2135
PF	2.9885	3.2698	3.5012	1.2214	0.9257	0.8368

6

Conclusiones

En este trabajo se planteó el problema de filtraje y estimación de estados para sistemas Hammerstein-Wiener, donde el sistema es representado por dos no linealidades estáticas que afectan tanto la entrada como la salida, y la dinámica del sistema es representada mediante un modelo LTI en espacio de estados.

La idea principal del método propuesto en esta Tesis es proveer un único algoritmo que permita resolver el problema de filtraje y estimación de estados para sistemas Hammerstein-Wiener para una amplia variedad de funciones no lineales. Estas funciones incluyen las funciones monótonas a trozos que permiten representar muchas no linealidades que en general son no monótonas pero que se pueden dividir en un número finito de funciones estrictamente monótonas. Adicionalmente, para darle más generalidad al método, se incluyen funciones que no satisfacen la condición de monotonía estricta por trozos como el caso del cuantizador binario. Dado lo anterior, también se incluyen funciones no lineales típicas como lo son la función saturación y zona muerta, las que pueden abordarse como una mezcla entre el cuantizador binario y una función afín.

Para resolver las ecuaciones de filtraje bayesiano, en primer lugar se usó la regla de cuadratura gaussiana para resolver el modelo de $p(y_t|x_t)$, el cual se puede definir como una ecuación integral en todos los casos tratados en esta Tesis. Se aprovechó este hecho para generar un modelo probabilístico de $p(y_t|x_t)$ con una única representación, lo cual es posible ya que la aproximación de la integral mediante cuadratura gaussiana siempre generará una suma ponderada de gaussianas evaluadas en los puntos de dicha cuadratura. Por esta razón, el modelo de $p(y_t|x_t)$ siempre tiene una forma de suma de gaussianas dada por

$$p(y_t|x_t) \approx \sum_{j=1}^{K} \beta_j \mathcal{N}\left(\xi_t^j; Cx_t + Df(u_t), R\right), \tag{6.1}$$

en la cual, para cada función no lineal, se deben calcular los parámetros β_j y ξ_t^j de acuerdo a los respectivos teoremas desarrollados en el capítulo 3 sección 3.3.

Una vez definido el modelo de $p(y_t|x_t)$ como una suma de gaussianas, éste se propaga naturalmente en la ecuaciones de las etapas de predicción y corrección del algoritmo de filtraje bayesiano, provocando que las distribuciones de filtraje y predicción sean también sumas de gaussianas. Este hecho permitió definir el filtro suma de gaussianas para sistemas Hammerstein-Wiener, donde la no linealidad de la salida puede tomar muchas formas.

En los ejemplos simulados se puede verificar las bondades del método con respecto a algunos algoritmos de estimación de estados disponibles en la literatura. A continuación se listan las conclusiones al usar el método propuesto comparado con el filtro de Kalman estándar, el filtro de Kalman extendido, el filtro de Kalman en cuadratura, el filtro de Kalman cuantizado, y el filtro de partículas:

1. Las PDFs de filtraje obtenidas usando el filtro suma de gaussianas en todos los casos se ajustan con mucha precisión a las PDFs verdaderas proporcionadas por el filtro de partículas con 20000 partículas. En cambio, las PDFs obtenidas usando los otros métodos difieren en distinto grado de las PDFs verdaderas. En los ejemplos se observa que las PDFs obtenidas con el filtro de Kalman extendido, el filtro de Kalman en cuadratura y el filtro de Kalman cuantizado son las que están más alejadas tanto en media como en varianza de las PDFs verdaderas.

2. El filtro de partículas produce PDFs de filtraje muy precisas y dicha precisión puede mejorar si se aumenta el número de partículas, lo que trae consigo un dramático aumento en la carga computacional.

3. Respecto a la estimación de estados, tanto el filtro suma de gaussianas como el filtro de partículas producen resultados muy parecidos. Esto se debe a que aún cuando el filtro de partículas (con pocas partículas) no produce una PDF muy ajustada a las PDFs verdaderas, en la mayoría de casos coincide fielmente con la media de dicha distribución de filtraje de los estados. Por otro lado, el resto de filtros, como en la mayoría de casos, no coinciden con la media de las PDFs verdaderas, entonces estiman los estados del sistema con variabilidad alta. En las simulaciones se observa (ejemplos 2,4,5,6) que incluso la media de las 100 estimaciones de Monte Carlo difiere de los estados verdaderos.

4. Respecto al tiempo de cómputo, el filtro suma de gaussianas produce un tiempo considerablemente bajo cuando se usan 10 componentes gaussianas para aproximar $p(y_t|x_t)$. Este número produce en la mayoría de ejemplos resultados bastante precisos. Sin embargo, en los casos más demandantes como el cuantizador binario y la función saturación, en donde se hace necesario usar más componentes para mejorar las estimaciones, el tiempo de cómputo fue comparable con el tiempo del filtro de partículas. En estos dos casos se usaron 100 y 50 componentes gaussianas respectivamente. Los filtros basados en el filtro de Kalman exhiben tiempos de cómputo muy pequeños en todos los casos, aunque las estimaciones son poco precisas.

6.1 Trabajo futuro

Como trabajo futuro se proponen los siguientes tópicos que permitirán extender el trabajo desarrollado en esta tesis:

1. Extender el algoritmo de filtraje a un algoritmo suavizador que permita determinar los estados condicionados a los datos desde $t = 1, \ldots, N$. Este caso no es trivial ya que las PDFs de filtraje son no gaussianas (suma de gaussianas). En este contexto, aplicar la ecuación de suavizado tradicional no es posible ya que existe una división por una densidad no gaussiana, lo que dificulta los cálculos. Sin embargo, se puede usar la formulación *Two-filter formula* propuesta en (Kitagawa, 1994b), donde la ecuación de suavizado se divide en una recursión de filtraje en reversa llamada *backward filter* y con base en esta estimación en reversa se estiman los estados y las distribuciones de suavizado.

2. Usar los algoritmos de filtraje y suavizado para sistemas Hammerstein-Wiener e implementar el método de identificación de parámetros llamado Máxima Verosimilitud (ML) usando el método Expectation-Maximization (EM). El método EM, con base en los estados suavizados del sistema, permite definir un algoritmo recursivo para obtener una estimación de los parámetros del sistema, las varianzas de los ruidos e incluso una estimación de las no linealidades.

A

Apéndices

En este capítulo se presentan los algoritmos de filtraje disponibles en la literatura, así como algunos lemas y teoremas que se han usado en el desarrollo de esta tesis. Además se presentan los códigos usados en las simulaciones.

A.1 Algoritmos de filtraje en la literatura

Existen diversos algoritmos de filtraje en la literatura, muchos de ellos son variaciones del filtro de Kalman estándar para tratar con una no linealidad en específico y otros son enfoques más generales que permiten tratar con algunos tipos de no linealidades, particularmente con sistemas Hammerstein-Wiener. A continuación haremos un breve repaso de estos algoritmos de filtraje, los cuales serán usados para comparar los resultados obtenidos con el algoritmo de filtraje que se presentará en este trabajo.

A.1.1 Sistema extendido

Para facilitar la implementación de algunos de los algoritmos de filtraje que se describirán a continuación, es necesario reescribir el sistema en (1.1)-(1.3) y (1.4) como un sistema extendido que incluye en un nuevo estado extendido al estado x_t y a la salida no medible r_t de la siguiente manera:

$$\chi_{t+1} = \mathcal{A}\chi_t + \mathcal{B}f(\tilde{u}_t) + \tilde{w}_t, \tag{A.1}$$

$$y_t = g\left(\begin{bmatrix}0 & 1\end{bmatrix}\chi_t\right) + \eta_t, \tag{A.2}$$

donde

$$\mathcal{A} = \begin{bmatrix}A & 0\\ CA & 0\end{bmatrix}, \qquad \mathcal{B} = \begin{bmatrix}B & 0\\ CB & D\end{bmatrix}, \tag{A.3}$$

$\chi_t = [x_t^T \quad r_t]^T$ es el estado extendido, $\tilde{u}_t = [f(u_t)^T \quad f(u_{t+1})^T]$, es la entrada extendida con $u_{N+1} = 0$, y el ruido $\tilde{w}_t = [w_t^T \quad (Cw_t + v_{t+1})^T]^T$ satisface $\tilde{w}_t \sim \mathcal{N}(\tilde{w}_t; 0, \mathcal{Q})$ con

$$\mathcal{Q} = \begin{bmatrix}Q & QC^T\\ CQ^T & CQC^T + R\end{bmatrix}. \tag{A.4}$$

La condición inicial del estado extendido se puede describir de la siguiente manera:

$$\chi_1 \sim \mathcal{N}\left(\chi_1; \epsilon_1, \Psi_1\right), \tag{A.5}$$

donde $\epsilon_1 = \begin{bmatrix}\mu_1^T & 0\end{bmatrix}^T$ y $\Psi_1 = \text{diag}\{P_1, 1\}$. Note que la condición inicial del estado extendido requiere definir una condición inicial para la dinámica de la salida, en donde asumimos que $r_1 \sim \mathcal{N}(r_1; 0, 1)$. Este forma de extender el sistema se usa típicamente cuando se quiere estimar la salida no medible r_t a partir de mediciones de y_t.

A.1.2 Filtro de Kalman estándar

El filtro de Kalman estándar (Kalman, 1960; Kalman and Bucy, 1961) es uno de los algoritmos de filtraje más conocido y de mayor uso en la literatura. Este filtro ha sido usado en muchas aplicaciones en donde el sistema es lineal (estados y salida) y sus correspondientes ruidos son gaussianos. El filtro de Kalman estándar es el único filtro tratado en esta tesis que cumple con el lema 2, es decir, es óptimo y de mínima varianza (Friedland, 2012). Considere el sistema en (1.1) y (1.2), entonces las ecuaciones de filtraje en el Teorema 4 tienen solución cerrada dada por:

$$p(x_t|y_{1:t}) = \mathcal{N}\left(x_t; \hat{x}_{t|t}, \Sigma_{t|t}\right), \tag{A.6}$$

$$p(x_{t+1}|y_{1:t}) = \mathcal{N}\left(x_{t+1}; \hat{x}_{t+1|t}, \Sigma_{t+1|t}\right), \tag{A.7}$$

donde

$$K_t = \Sigma_{t|t-1}C^T\left(R + C\Sigma_{t|t-1}C^T\right)^{-1}, \tag{A.8}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t\left(y_t - C\hat{x}_{t|t-1} - Df(u_t)\right), \tag{A.9}$$

$$\Sigma_{t|t} = (I - K_tC)\Sigma_{t|t-1}, \tag{A.10}$$

$$\hat{x}_{t+1|t} = A\hat{x}_{t|t} + Bf(u_t), \tag{A.11}$$

$$\Sigma_{t+1|t} = Q + A\Sigma_{t|t}A^T. \tag{A.12}$$

En el algoritmo 3 se presentan los pasos para implementar el filtro de Kalman estándar.

Algorithm 3: Filtro de Kalman.

Input: La PDF del estado inicial $p(x_1)$, e.g. $\hat{x}_{1|0} = \mu_1$, $\Sigma_{1|0} = P_1$.
for $t = 1$ **to** N **do**
  Cacular la ganancia K_t en (A.8).
  Etapa de corrección:
  Calcular $\hat{x}_{t|t}$ según (A.9).
  Calcular $\Sigma_{t|t}$ según (A.10).
  Etapa de predicción:
  Calcular $\hat{x}_{t+1|t}$ según (A.11).
  Calcular $\Sigma_{t+1|t}$ según (A.12).
end
Output: La estimación del estado $\hat{x}_{t|t}$ y la matriz de covarianza del error de estimación $\Sigma_{t|t}$ para $t = 1, \ldots, N$.

A.1.3 Filtro de Kalman extendido

El filtro de Kalman extendido (Gelb et al., 1974; Grewal and Andrews, 2014) es una versión sub-óptima del filtro de Kalman estándar en donde se estiman los estados de un sistema no lineal usando las mismas

ideas del filtro de Kalman para sistemas lineales. Para aplicar el filtro de Kalman extendido se hará uso del sistema extendido en (A.1)-(A.2). En este caso, la ganancia del filtro es calculada mediante linealización del modelo no lineal en torno a una estimación de χ_t, mediante una serie de Taylor de primer orden (aunque también existe una versión que usa la serie de Taylor de segundo orden). Para el sistema (A.1)-(A.2), la función $g(\cdot)$ se expande alrededor del estado de la etapa de predicción $\hat{\chi}_{t|t-1}$ de la siguiente forma:

$$g(\chi_t) = g(\hat{\chi}_{t|t-1}) + C_t(\chi_t - \hat{\chi}_{t|t-1}), \tag{A.13}$$

donde C_t es el Jacobiano de $g(\cdot)$ dado por:

$$\mathcal{C}_t = \left.\frac{\partial g(\chi_t)}{\partial \chi_t}\right|_{\chi_t=\hat{\chi}_{t|t-1}}, \tag{A.14}$$

Con esta linealización el sistema dado en (A.1)-(A.2) se puede reescribir como un sistema lineal variante en el tiempo de la siguiente forma:

$$\begin{aligned} \chi_{t+1} &= \mathcal{A}\chi_t + \mathcal{B}f(\tilde{u}_t) + \tilde{w}_t, \\ y_t &= \mathcal{C}_t\chi_t + \mathcal{D}(\hat{\chi}_{t|t-1}) + \eta_t, \end{aligned} \tag{A.15}$$

donde $\mathcal{D}(\hat{\chi}_{t|t-1}) = g(\hat{\chi}_{t|t-1}) - \mathcal{C}_t\hat{\chi}_{t|t-1}$, con lo cual las ecuaciones del filtro de Kalman extendido son las siguientes:

$$K_t = \Gamma_{t|t-1}\mathcal{C}_t^T\left(P + \mathcal{C}_t\Gamma_{t|t-1}\mathcal{C}_t^T\right)^{-1}, \tag{A.16}$$

$$\hat{\chi}_{t|t} = \hat{\chi}_{t|t-1} + K_t\left(y_t - g(\hat{\chi}_{t|t-1})\right), \tag{A.17}$$

$$\Gamma_{t|t} = (I - K_t\mathcal{C}_t)\Gamma_{t|t-1}, \tag{A.18}$$

$$\hat{\chi}_{t+1|t} = \mathcal{A}\hat{\chi}_{t|t} + \mathcal{B}f(\tilde{u}_t), \tag{A.19}$$

$$\Gamma_{t+1|t} = \mathcal{Q} + \mathcal{A}\Gamma_{t|t}\mathcal{A}^T, \tag{A.20}$$

donde $\hat{\chi}_{1|0} = \epsilon_1$ y $\Gamma_{1|0} = \Psi_1$. En el algoritmo 4 se presentan los pasos para implementar el filtro de Kalman de extendido.

Algorithm 4: Filtro de Kalman Extendido.

Input: La PDF del estado inicial $p(x_1)$, e.g. $\hat{\chi}_{1|0} = \epsilon_1$, $\Gamma_{1|0} = \Psi_1$.
for $t = 1$ **to** N **do**
  Cacular la ganancia K_t en (A.16).
  Etapa de corrección:
  Calcular $\hat{\chi}_{t|t}$ según (A.17).
  Calcular $\Gamma_{t|t}$ según (A.18).
  Etapa de predicción:
  Calcular $\hat{\chi}_{t+1|t}$ según (A.19).
  Calcular $\Gamma_{t+1|t}$ según (A.20).
end
Output: La estimación del estado $\hat{\chi}_{t|t}$ y la matriz de covarianza del error de estimación $\Gamma_{t|t}$ para $t = 1, \ldots, N$.

A.1.4 Filtro de Kalman cuantizado

El filtro de Kalman cuantizado es otra versión del filtro de Kalman estándar que trata de incluir el efecto de la cuantización en la estimación de los estados. Note que en el caso de datos cuantizados, el filtro de

Kalman extendido no es directamente aplicable ya que el cuantizador no es una función diferenciable. El filtro de Kalman cuantizado modifica la ecuación (A.9) para introducir la cuantización. A continuación se presentan dos formas en que se suele modificar el filtro de Kalman para considerar datos cuantizados (Gómez and Sad, 2020; Leong et al., 2013):

$$\text{Forma 1:} \qquad \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \left(y_t - \mathfrak{q}\left\{C\hat{x}_{t|t-1} - Df(u_t)\right\}\right), \tag{A.21}$$

$$\text{Forma 2:} \qquad \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \mathfrak{q}\left\{y_t - C\hat{x}_{t|t-1} - Df(u_t)\right\}. \tag{A.22}$$

En este trabajo se usa la forma 1. El algoritmo del filtro de Kalman cuantizado es similar al mostrado en el algoritmo 3 reemplazando la ecuación (A.9) por la ecuación (A.21).

A.1.5 Filtro de Kalman en cuadratura

El filtro de de Kalman en cuadratura (Arasaratnam et al., 2007; Singh and Bhaumik, 2015), a diferencia del filtro de Kalman extendido que linealiza las ecuaciones no lineales del estado y/o salida mediante series de Taylor, estima una ecuación lineal que aproxime las ecuaciones no lineales del estado y/o salida ajustando mediante regresión lineal tal que se minimice el error cuadrático medio entre el sistema verdadero (no lineal) y el sistema aproximado. Para obtener esta aproximación del sistema, bajo el criterio de optimalidad del MSE, se necesita resolver integrales de la forma `polinomio*PDF_gaussiana`, las cuales se resuelven mediante cuadratura de Gauss-Hermite.

En el algoritmo 5 se presentan los pasos para implementar el filtro de Kalman en cuadratura para sistemas Hammerstein-Wiener.

Algorithm 5: Filtro de Kalman en cuadratura.

Input: La PDF del estado inicial $p(x_1)$, e.g. $\hat{\chi}_{1|0} = \epsilon_1$, $\Gamma_{1|0} = \Psi_1$. Los puntos de la cuadratura de Gauss-Hermite $\{\sigma_\tau, \zeta_\tau\}_{\tau=1}^{L}$.

for $t = 1$ **to** N **do**

  Etapa de corrección:

  Fatorizar $\Gamma_{t|t-1} = \Lambda_{t|t-1}\Lambda_{t|t-1}^T$.

  Para $\tau = 1, \ldots, L$, evaluar $X_{t|t-1}^\tau = \Lambda_{t|t-1}\zeta_\tau + \hat{\chi}_{t|t-1}$ y $Y_{t|t-1}^\tau = g(X_{t|t-1}^\tau)$.

  Calcular la predicción de la salida $\hat{y}_{t|t-1} = \sum_{\tau}^{L} \sigma_\tau Y_{t|t-1}^\tau$.

  Calcular la matriz de covarianza $P_{t|t-1}^{yy} = R + \sum_{\tau}^{L} \sigma_\tau Y_{t|t-1}^\tau Y_{t|t-1}^{\tau T} - \hat{y}_{t|t-1}\hat{y}_{t|t-1}^T$.

  Calcular la matriz de covarianza $P_{t|t-1}^{xy} = \sum_{\tau}^{L} \sigma_\tau X_{t|t-1}^\tau Y_{t|t-1}^{\tau T} - \hat{\chi}_{t|t-1}\hat{y}_{t|t-1}^T$.

  Calcular la ganancia $K_t = P_{t|t-1}^{xy}\left(P_{t|t-1}^{yy}\right)^{-1}$.

  Calcular $\hat{\chi}_{t|t} = \hat{\chi}_{t|t-1} + K_t\left(y_t - \hat{y}_{t|t-1}\right)$.

  Calcular $\Gamma_{t|t} = \Gamma_{t|t-1} + K_t P_{t|t-1}^{yy} K_t^T$.

  Etapa de predicción:

  Calcular $\hat{\chi}_{t+1|t}$ según (A.19).

  Calcular $\Gamma_{t+1|t}$ según (A.20).

end

Output: $\hat{\chi}_{t|t}$ y $\Gamma_{t|t}$ para $t = 1, \ldots, N$.

A.1.6 Filtro de partículas

El filtro de partículas (Doucet et al., 2000a; Gordon et al., 1993a), llamado también muestreo secuencial de importancia (sequential importance sampling), es un algoritmo basado en el método de Monte Carlo, que usa un conjunto de muestras y pesos para representar la PDF del estado en un instante particular. Al propagar estas muestras y pesos a través de las funciones no lineales del sistema es posible obtener estimaciones del estado directamente sin conocer explícitamente su distribución a posteriori. Esto significa que se puede representar aproximadamente la PDF de filtraje $p(x_t|y_{1:t})$ del estado condicionado a las observaciones $y_{1:t}$ mediante un conjunto de pesos y muestras llamadas partículas de tal forma que:

$$p(x_t|y_{1:t}) \approx \sum_{i=1}^{N_p} \kappa_t^{(i)} \delta\left(x_t - x_t^{(i)}\right), \tag{A.23}$$

donde $\delta(\cdot)$ es la función delta de Dirac, $\kappa_t^{(i)}$ es el i-ésimo peso, $x_t^{(i)}$ es la i-ésima partícula maestreada de la PDF de filtraje $p(x_t|y_{1:t})$, y N_p es el número de partículas. En general es difícil tomar muestras de $p(x_t|y_{1:t})$, por lo que las partículas se generan de otra distribución de la cual es fácil tomar muestras. Esta distribución se denomina *distribución de importancia*, con lo cual los pesos de importancia se pueden calcular en forma recursiva de la siguiente manera:

$$\kappa_t^{(i)} \propto \kappa_{t-1}^{(i)} \frac{p(y_t|x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t|x_{t-1}^{(i)}, y_t)}. \tag{A.24}$$

donde $q(x_t|x_{t-1}^{(i)}, y_t)$ es la distribución de importancia y $\kappa_{t-1}^{(i)}$ son los pesos de importancia de la iteración previa. En (Doucet et al., 2000b; Särkkä, 2013) se indica que la distribución de importancia óptima es $p(x_t|x_{t-1}^{(i)}, y_t)$ en el sentido de que minimiza la varianza de los pesos de importancia $\kappa_t^{(i)}$. Desafortunadamente, en la mayoría de casos, tomar muestras de la distribución de importancia óptima es también una tarea complicada, por lo que una de la elecciones típicas es asumir que la distribución de importancia es la PDF de transición de estados $p(x_t|x_{t-1})$ (Doucet et al., 2000b; Hostettler, 2015). Esta elección produce un algoritmo del filtro de partículas muy intuitivo y fácil de implementar, en donde $\kappa_t^{(i)} \propto \kappa_{t-1}^{(i)} p(y_t|x_t^{(i)})$. Sin embargo, cuando se usa un número elevado de muestras y/o el orden del sistema crece, este filtro presenta un alto costo computacional. El filtro de partículas que usa $p(x_t|x_{t-1})$ como distribución de importancia toma el nombre de *bootstrap* filter (Gordon et al., 1993b). En el algoritmo 6 se presentan los pasos para implementar el filtro de partículas para sistemas Hammerstein-Wiener.

Algorithm 6: Filtro de Partículas.

1 **Input:** La PDF del estado inicial $p(x_1)$. El número de partículas M.
2 Tomar muestras $x_1^{(i)} \sim p(x_1)$ con $i = 1, \ldots, M$.
3 **for** $t = 1$ **to** N **do**
4 Tomar muestras de la distribución de importacia $x_t^{(i)} \sim p(x_t|x_{t-1}^{(i)})$ con $i = 1, \ldots, M$.
5 Calcular los pesos $\kappa_t^{(i)}$ de acuerdo a: $\kappa_t^{(i)} = \dfrac{p(y_t|x_t^{(i)})}{\sum_{j=1}^{M} p(y_t|x_t^{(j)})}$ con $i = 1, \ldots, M$.
6 Para $j = 1, \ldots, M$, generar nuevas partículas $x_t^{(j)}$ usando resampling de acuerdo a $P(x_t^{(j)} = \tilde{x}_t^{(i)}) = \kappa_t^{(i)}$, con $i = 1, \ldots, M$.
7 **end**
8 **Output:** El conjunto de partículas $\{x_t^{(j)}\}_{j=1}^{M} \sim p(x_t|y_{1:t})$.

A.2 Lemas y Teoremas usados en está Tesis

Teorema 17 (Transformación de variables aleatorias). *Considere dos variables aleatoria continuas $X, Y \in \mathbb{R}^n$ con $Y = g(X)$. Suponga que $g^{-1}(\cdot)$ existe y que tanto $g(\cdot)$ como $g^{-1}(\cdot)$ son continuamente diferenciables. Entonces la PDF de Y esta dada por*

$$p_Y(y) = p_X\left[g^{-1}(y)\right]\left|\det\left(\frac{\partial g^{-1}(y)}{\partial y}\right)\right|, \tag{A.25}$$

donde $\left|\det\left(\partial g^{-1}(y)/\partial y\right)\right| > 0$ es el valor absoluto del determinante del jacobiano $\partial g^{-1}(y)/\partial y$.

Demostración. *Ver por ejemplo (Jazwinski, 1970)*

Teorema 18 (Transformación de variables aleatorias [Piecewise]). *Considere una variable aleatoria continua $X \in \mathbb{R}$ con dominio $R_X \subset \mathbb{R}$ y sea $g : \mathbb{R} \to \mathbb{R}$ tal que $Y = g(X)$. Suponga que R_X se puede particionar en un número finito de intervalos tal que $g(x)$ es estrictamente monótona y diferenciable en cada partición. Entonces la PDF de Y esta dada por*

$$p_Y(y) = \sum_{i=1}^{M}\left|\frac{dx_i}{dy}\right| p_X(x_i), \tag{A.26}$$

donde $|\cdot|$ es la función valor absoluto y $x_1, x_2, \ldots, x_M$ son las soluciones reales de la ecuación $g(x) = y$.

Demostración. *Ver por ejemplo (Evans et al., 2008; Papoulis and Pillai, 1989)*

Teorema 19. *Si la PDF $p(x) = \mathcal{N}(x; \mu_x, Q)$ y la PDF condicional $p(y|x) = \mathcal{N}\left(y; Cx + \mu_y, R\right)$, entonces*

- *La PDF conjunta $p(x, y) = p(x)p(y)$ donde*

$$\begin{aligned} p(x) &= \mathcal{N}\left(x; \mu_x + K(y - C\mu_x - \mu_y), (I - KC)Q\right), \\ p(y) &= \mathcal{N}\left(y; C\mu_x + \mu_y, R + CQC^T\right), \\ K &= QC^T\left(R + CQC^T\right)^{-1}, \end{aligned}$$

- *Las PDFs marginales*

$$p(x) = \int p(x, y)dy, \qquad p(y) = \int p(x, y)dx. \tag{A.27}$$

Demostración. *Ver por ejemplo (Kitagawa and Gersch, 1996) [pag 86].* □

Lema 20. *Considere las siguientes sumas*

$$\mathcal{G} = \sum_{j=1}^{K} \mathcal{G}_j, \qquad \mathcal{F} = \sum_{k=1}^{M} \mathcal{F}_k. \tag{A.28}$$

Luego, para cada par (j,k) donde $j = 1,\dots,K$ y $k = 1,\dots,M$, el producto $\mathcal{H} = \mathcal{G}\mathcal{F}$ tiene KM términos indexados por $\ell = (k-1)K + j$, ver Figura A.1. Luego reordenando los términos de la nueva suma se obtiene

$$\mathcal{H} = \sum_{\ell=1}^{KM} \mathcal{H}_\ell, \qquad \mathcal{H}_\ell = \mathcal{G}_j \mathcal{F}_k. \tag{A.29}$$

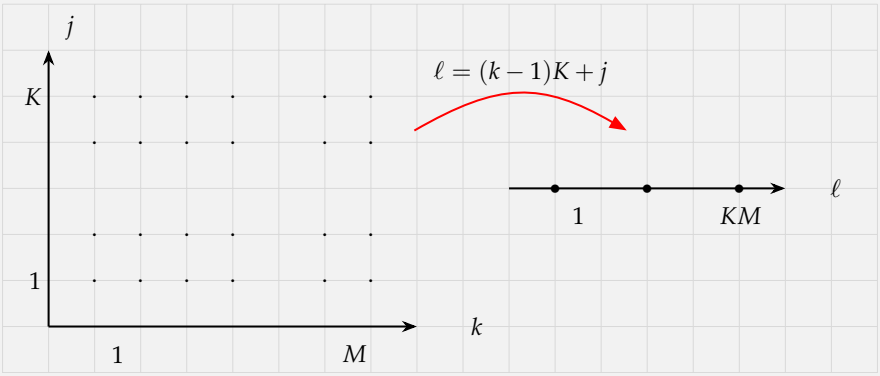


Figura A.1: Mapeo de los índices del producto de dos sumas.

Demostración. *Primero expresamos el producto de las dos sumas de la siguiente manera*

$$\mathcal{H} = \sum_{k=1}^{M} \sum_{j=1}^{K} \mathcal{F}_k \mathcal{G}_j, \tag{A.30}$$

Nótese que si expande la doble sumatoria se obtienen los siguientes índices

$$\begin{aligned} &k = 1 \quad j = 1,\dots,K \quad \longrightarrow \quad \ell = 1,\dots,K \\ &k = 2 \quad j = 1,\dots,K \quad \longrightarrow \quad \ell = K+1,\dots,2K \\ &\vdots \\ &k = M \quad j = 1,\dots,K \quad \longrightarrow \quad \ell = (M-1)K+1,\dots,KM. \end{aligned} \tag{A.31}$$

Es evidente que el total de términos en la nueva sumatoria $\mathcal{H}$ es KM y los índices ℓ satisfacen $\ell = (k-1)K + j$, con lo cual termina la prueba. □

A.3 Equivalencia con el filtro de Kalman cuando $g(\cdot)$ es una función afín

En este apartado de demuestra que resolviendo la integral en (3.39) correspondiente a $p(y_t|x_t)$ de una función afín, usando la metodología propuesta en esta tesis, es decir

$$p(y_t|x_t) = \frac{1}{\mathbb{A}} \int \mathcal{N}(\eta_t; 0, P)\, \mathcal{N}\left(\frac{y_t - \eta_t - b}{\mathbb{A}}; Cx_t + Df(u_t), R\right) d\eta_t, \tag{A.32}$$

se obtiene la siguiente PDF que corresponde al modelo de la salida cuando $g(\cdot)$ es una función afín, cuya ecuación se puede obtener directamente del modelo del sistema como

$$p(y_t|x_t) = \mathcal{N}\left(y_t; \mathbb{A}\left[Cx_t + Df(u_t)\right] + b, \mathbb{A}^2R + P\right). \tag{A.33}$$

Para esto trabajaremos en el argumento de la función exponencial de las dos PDFs gaussianas, de la siguiente manera

$$\text{Arg} = -\frac{1}{2}\left[\frac{(y_t - \eta_t - \lambda)^2}{\mathbb{A}^2R} + \frac{\eta_t^2}{P}\right], \tag{A.34}$$

en donde, por facilidad hemos llamado $\lambda = \mathbb{A}\left[Cx_t + Df(u_t)\right] + b$, luego

$$\begin{aligned}
\text{Arg} &= -\frac{1}{2}\left[\frac{P\left(y_t^2 + \eta_t^2 + \lambda^2 - 2y_t\eta_t - 2y_t\lambda + 2\eta_t\lambda\right) + \mathbb{A}^2R\eta_t^2}{\mathbb{A}^2RP}\right], \\
&= -\frac{1}{2}\left[\frac{\left(\mathbb{A}^2R + P\right)\eta_t^2 - 2P\left(y_t - \lambda\right)\eta_t + P\left(y_t - \lambda\right)^2}{\mathbb{A}^2RP}\right], \\
&= -\frac{1}{2}\left[a_1\eta_t^2 - 2a_2\eta_t + a_3\right],
\end{aligned} \tag{A.35}$$

donde

$$a_1 = \frac{\mathbb{A}^2R + P}{\mathbb{A}^2RP}, \tag{A.36}$$

$$a_2 = \frac{(y_t - \lambda)}{\mathbb{A}^2R}, \tag{A.37}$$

$$a_3 = \frac{(y_t - \lambda)^2}{\mathbb{A}^2R}. \tag{A.38}$$

Entonces la integral que deseamos resolver se convierte en

$$p(y_t|x_t) = \frac{1}{2\pi\mathbb{A}\sqrt{RP}}\int \exp\left\{-\frac{1}{2}\left[a_1\eta_t^2 - 2a_2\eta_t + a_3\right]\right\} d\eta_t, \tag{A.39}$$

Esta última integral tiene solución cerrada, ver (Harville, 1998), dada por

$$\sqrt{\frac{2\pi}{a_1}}\exp\left\{\frac{a_2^2}{2a_1} - \frac{a_3}{2}\right\}, \tag{A.40}$$

con lo cual

$$\begin{aligned}
p(y_t|x_t) &= \frac{1}{2\pi\mathbb{A}\sqrt{RP}}\sqrt{\frac{2\pi}{a_1}}\exp\left\{-\frac{1}{2}\left(a_3 - \frac{a_2^2}{a_1}\right)\right\}, \\
&= \frac{1}{\sqrt{2\pi\left(\mathbb{A}^2R + P\right)}}\exp\left\{-\frac{1}{2}\left(\frac{(y_t - \lambda)^2}{\mathbb{A}^2R} - \frac{P\left(y_t - \lambda\right)^2}{\mathbb{A}^2R(\mathbb{A}^2R + P)}\right)\right\}, \\
&= \frac{1}{\sqrt{2\pi\left(\mathbb{A}^2R + P\right)}}\exp\left\{-\frac{1}{2}\left(\frac{\left(\mathbb{A}^2R + P\right)\left(y_t - \lambda\right)^2 - P\left(y_t - \lambda\right)^2}{\mathbb{A}^2R(\mathbb{A}^2R + P)}\right)\right\}, \\
&= \frac{1}{\sqrt{2\pi\left(\mathbb{A}^2R + P\right)}}\exp\left\{-\frac{1}{2}\left(\frac{(y_t - \lambda)^2}{\mathbb{A}^2R + P}\right)\right\},
\end{aligned} \tag{A.41}$$

resultado que corresponde a la PDF gaussiana $\mathcal{N}\left(y_t; \lambda, \mathbb{A}^2R + P\right)$, y reemplazando λ se obtiene el resultado en (A.33).

A.4 Problemas numéricos en el cálculo de los pesos en el GSF.

Nótese que el cómputo de los pesos normalizados en la etapa de corrección del filtro suma de gaussianas envuelve la siguiente expresión

$$\delta_{t|t}^{\ell} = \frac{\bar{\delta}_{t|t}^{\ell}}{\sum_{r=1}^{\mathcal{S}_{t|t}} \bar{\delta}_{t|t}^{r}}, \qquad \bar{\delta}_{t|t}^{\ell} = \beta_j \delta_{t|t-1}^{k} \mathcal{N}\left(\xi_t^j; \varsigma_t^k, \Phi_t^k\right), \tag{A.42}$$

es decir, cada peso requiere evaluar un término con una exponencial y dividir por la suma de todas las evaluaciones de los pesos no normalizados $\bar{\delta}_{t|t}^{\ell}$. En general, cuando el argumento de la exponencial es relativamente alto (aproximadamente en el rango -800 y 800), Matlab produce 0 o ∞, respectivamente, debido al límite de la precisión con la que trabaja. Estos dos efectos son conocidos con underflow y overflow. Para nuestro caso de estudio, si una de las evaluaciones, ya sea en el numerador o denominador de la ecuación para calcular $\delta_{t|t}^{\ell}$ se hace ∞, entonces este símbolo se propaga a todos los demás cálculos produciendo que el algoritmo falle.

Afortunadamente, este problema ya ha sido estudiado en la literatura y es llamado log-sum-exp (Blanchard et al., 2020), en donde se requiere evaluar la siguiente función y su gradiente:

$$f(x) = \log\left\{\sum_{i=1}^{n} \exp\{x_i\}\right\}, \tag{A.43}$$

$$g_j(x) = \frac{\partial}{\partial x_j} f(x) = \frac{\exp\{x_j\}}{\sum_{i=1}^{n} \exp\{x_i\}}, \tag{A.44}$$

donde $x = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$ y las funciones $f : \mathbb{R}^n \to \mathbb{R}$ y $g : \mathbb{R}^n \to \mathbb{R}^n$. Note que para nuestro caso de interés se requiere evaluar los pesos que tienen la misma forma que $g_j(x)$. Una de las formas de abordar el problema log-sum-exp es multiplicar convenientemente por $\exp\{a\}\exp\{-a\} = 1$ con lo cual se obtiene

$$f(x) = a + \log\left\{\sum_{i=1}^{n} \exp\{x_i - a\}\right\}, \tag{A.45}$$

$$g_j(x) = \frac{\partial}{\partial x_j} f(x) = \frac{\exp\{x_j - a\}}{\sum_{i=1}^{n} \exp\{x_i - a\}}, \tag{A.46}$$

donde a se escoge como $a = \max_i x_i$. La manera como se ha elegido a permite evitar números excesivamente grandes en el argumento de la función exponencial, lo cual ayuda a evitar los efectos de underflow y overflow. Esta implementación es la más conocida y usada en muchos algoritmos y programas como en el toolbox de Deep learning de Matlab, así como en el programa R y en el paquete SciPy. En este trabajo se usa la implementación proporcionada en (Blanchard et al., 2020), la cual permite calcular tanto la función $f(x)$ como su gradiente $g_j(x)$, esta última llamada función softmax.

A.5 Código de las simulaciones

En está sección presentamos los códigos de las funciones necesarias para obtener los resultados mostrados en esta tesis. Por facilidad, presentamos únicamente el código del ejemplo 1, para correr los otros ejemplos basta con sustituir los valores de las matrices del sistema dados en cada ejemplo en el capítulo 5.

Ejemplo 1

```
%% EJEMPLO 1
    clc; clear all; close all;
% Modelo en espacio de estados.
    G.mm.A=0.9;
    G.mm.B=2.5;
    G.mm.C=1.1;
    G.mm.D=1.5;
    G.mm.Q=1;
    G.mm.R=0.5;
    G.mm.P=0.5;
% Semilla del generador de números aleatorios para reproducibilidad.
    rng(600000);
% Función no lineal g(.) y cáculo de gamma, phi y M.
    G.mm.fun='square';
    [gz,gamma,phi,M]=choosing_g_function(G.mm.fun);
    G.mm.gz=gz;
    G.mm.gamma=gamma;
    G.mm.phi=phi;
    G.sim.M=M;
% Parámetros de la condición inicial.
    G.mm.Icon.xin=1;
    G.mm.Icon.Pin=0.1*G.mm.Q;
% Parámetros de simulación.
    G.sim.MC=100;
    G.sim.N=100;
    G.sim.Npar=300;
    G.sim.Nsamp=2000;
    G.sim.minG=2;
    G.sim.maxG=2;
    G.sim.Gmin=2;
    G.sim.L=10;
% Entrada del sistema.
    G.sim.ut = num2cell(atan(sqrt(2)*randn(1,G.sim.N)),1);
% Escoge un experimento ('single') o 100 experimentos Monte Carlo ('MC').
    act='MC';
% Correr los algoritmos de filtraje - un solo experimento.
if strcmp(act,'single')
    % Corre los algoritmos de filtraje: EKF, QKF, GSF, PF.
    [G,EKF,QKF,GSF,PF,~] = running_monotone_filtering(G);
    % Corre el algoritmo de PF con 20000 partículas para obtener la PDF GT.
    G.sim.Npar=20000;
    [PFT] = particle_filter(G);
    % Datos de las salidas para graficar.
    t=1:1:G.sim.N;
    yt=cell2mat(G.sim.yt);
    rt=cell2mat(G.sim.rt);
    % Gráfica de las salidas yt y rt
    figure; grid on; hold on;
    plot(t,rt,'r', 'linewidth',1)
    plot(yt,'o', 'linewidth',1)
    lg=legend('$y_t$','$r_t$');
    set(lg,'Interpreter','latex');
    set(lg,'FontSize',12);
    grid on; box on; xlabel('t'); hold off;

    x=-10:0.05:10;
    % Cómputo de las PDFs obtenidas por cada algoritmo de filtraje para
    % algunos instantes de tiempo
    figure;
    ind=[15,21,34,41,42,49,53,76,93];
    for j=1:1:length(ind)
```

```
        % PDF de filtraje obtenida usando Extended Kalman Filter
        yek=kalpdf(x,ind(j),EKF,'filtered');
        % PDF de filtraje obtenida usando Quantized Kalman Filter
        yqk=kalpdf(x,ind(j),QKF,'filtered');
        % PDF de filtraje obtenida usando Gaussian Sum Filter
        yh=gmmpdf(x,ind(j),GSF,'filtered');
        % PDF de filtraje obtenida usando Particle Filter
        yp = parpdf(x,ind(j),PF,'filtered');
        % PDF GT obtenida usando Particle Filter
        yr = parpdf(x,ind(j),PFT,'filtered');
        % Gráfica de las PDFs de filtraje
        subplot(3,3,j); grid on; hold on;
        plot(x,yek,'LineWidth', 2)
        plot(x,yqk,'LineWidth', 2)
        plot(x,yh,'blue','LineWidth', 2)
        plot(x,yp,'cyan','LineWidth', 2)
        plot(x,yr,'black','LineWidth', 2)
        xlabel(sprintf('t=%d', ind(j))); ylabel('p(x_t|y_{1:t})')
        legend('EKF','QKF','GSF','PF','PFT');
        hold off;
    end
end
% Correr los algoritmos de filtraje - 100 experimentos Monte Carlo
if strcmp(act,'MC')
    for h=1:1:G.sim.MC
        % Corre los algoritmos de filtraje: EKF, QKF, GSF, PF
        [G,EKF,QKF,GSF,PF,TP] = running_monotone_filtering(G);
        xt=cell2mat(G.sim.xt);
        % Estimación del estado obtenido con cada filtro
        Xek(h,:)=cell2mat(EKF.xtt);
        Xqk(h,:)=cell2mat(QKF.xtt);
        Xh(h,:)=cell2mat(GSF.xtt);
        Xp(h,:)=cell2mat(PF.xtt);
        % Tiempo de cómputo de cada algoritmo de filtraje
        Tekf(h)=TP.tekf;
        Tqkf(h)=TP.tqkf;
        Tgsf(h)=TP.th;
        Tp(h)=TP.tp;
        % Cómputo del error cuadrático medio
        ekT(h)=immse(Xek(h,:),xt(1:G.sim.N));
        eqkT(h)=immse(Xqk(h,:),xt(1:G.sim.N));
        egsT(h)=immse(Xh(h,:),xt(1:G.sim.N));
        epT(h)=immse(Xp(h,:),xt(1:G.sim.N));
    end
    % Corre el algoritmo de PF con 20000 partículas para obtener a PDF GT.
    G.sim.Npar=20000;
    [PFT] = particle_filter(G);
    % Media del tiempo de cómputo
    mean(Tekf)
    mean(Tqkf)
    mean(Tgsf)
    mean(Tp)
    % Valores mínimos y máximos de la estimación de los estados
    Xek_min=min(Xek);       Xek_mean=mean(Xek);          Xek_max=max(Xek);
    Xqk_min=min(Xqk);       Xqk_mean=mean(Xqk);          Xqk_max=max(Xqk);
    Xh_min=min(Xh);         Xh_mean=mean(Xh);            Xh_max=max(Xh);
    Xp_min=min(Xp);         Xp_mean=mean(Xp);            Xp_max=max(Xp);
    %
    xt(:,1)=G.mm.Icon.xin;
    for i=1:1:G.sim.N
        xt(:,i+1) =  G.mm.A*xt(:,i) + G.mm.B*G.sim.ut{i};
    end
```

```
    % Gráfica de los resultados
    t=1:1:G.sim.N;
    figure; subplot(2,2,1); grid on; hold on;
    plot(t, Xek_min,'LineWidth', 1, 'Color','#CAD3D7')
    plot(t, Xek_max,'LineWidth', 1, 'Color','#CAD3D7')
    patch([t fliplr(t)], [Xek_min fliplr(Xek_max)], [202, 211,215]/255)
    plot(t, xt(t),'LineWidth', 2, 'Color','b')
    plot(t, Xek_mean,'LineWidth', 2, 'Color','r')
    set(gca,'ylim',[-30,50])
    hold off
    %
    subplot(2,2,2); grid on; hold on;
    plot(t, Xqk_min,'LineWidth', 1, 'Color','#CAD3D7')
    plot(t, Xqk_max,'LineWidth', 1, 'Color','#CAD3D7')
    patch([t fliplr(t)], [Xqk_min fliplr(Xqk_max)], [202, 211,215]/255)
    plot(t, xt(t),'LineWidth', 2, 'Color','b')
    plot(t, Xqk_mean,'LineWidth', 2, 'Color','r')
    set(gca,'ylim',[-30,50])
    hold off
    %
    subplot(2,2,3); grid on; hold on;
    plot(t, Xh_min,'LineWidth', 1, 'Color','#CAD3D7')
    plot(t, Xh_max,'LineWidth', 1, 'Color','#CAD3D7')
    patch([t fliplr(t)], [Xh_min fliplr(Xh_max)], [202, 211,215]/255)
    plot(t, xt(t),'LineWidth', 2, 'Color','b')
    plot(t, Xh_mean,'LineWidth', 2, 'Color','r')
    set(gca,'ylim',[-30,50])
    hold off
    %
    subplot(2,2,4); grid on; hold on;
    plot(t, Xp_min,'LineWidth', 1, 'Color','#CAD3D7')
    plot(t, Xp_max,'LineWidth', 1, 'Color','#CAD3D7')
    patch([t fliplr(t)], [Xp_min fliplr(Xp_max)], [202, 211,215]/255)
    plot(t, xt(t),'LineWidth', 2, 'Color','b')
    plot(t, Xp_mean,'LineWidth', 2, 'Color','r')
    set(gca,'ylim',[-30,50])
    hold off
    %
    figure; grid on;  hold on;   hAx=gca;
    boxplot([ekT',eqkT',egsT',epT'],'Labels',{'EKF','QKF','GSF','PF'})
    set(gca,'ylim',[0,30])
end
```

```
function [G,EKF,QKF,HF,PF,TP] = running_monotone_filtering(G)
    n=size(G.mm.A,1);
    p=size(G.mm.C,1);
    % Ruido del proceso
    wt = num2cell(chol(G.mm.Q)*randn(n,G.sim.N),1);
    % Ruido de la salida lineal (nomedible)
    vt = num2cell(chol(G.mm.R)*randn(p,G.sim.N),1);
    % Ruido de la salida no lineal (medible)
    nt = num2cell(chol(G.mm.P)*randn(p,G.sim.N),1);
    % Simulación del sistema
    xt(:,1)=G.mm.Icon.xin;
    for i=1:1:G.sim.N
        xt(:,i+1) =  G.mm.A*xt(:,i) + G.mm.B*G.sim.ut{i} +  wt{i};
        rt(i) = G.mm.C*xt(:,i) + G.mm.D*G.sim.ut{i} + vt{i} ;
        yt(1,i) = G.mm.gz(rt(i)) + nt{i};
    end
    % Estado y salidas lineal y no lineal
    G.sim.xt=num2cell(xt(:,1:G.sim.N),1);
    G.sim.rt=num2cell(rt,1);
```

```
    G.sim.yt=num2cell(yt,1);
    % Filtro de Kalman extendido.
    tic;
    [EKF] = extended_kalman(G);
    TP.tekf=toc;
    % Filtro de Kalman en cuadratura.
    tic;
    [QKF] = quadrature_kalman(G);
    TP.tqkf=toc;
    % Filtro de Sumas de Gaussianas.
    tic;
    [HF,~]= gaussian_sum_filter(G);
    TP.th=toc;
    % Filtro de particulas.
    tic;
    [PF] = particle_filter(G);
    TP.tp=toc;
end
```

```
function [G,KF,QKF,GSF,PF,TP] = running_quantizer_filtering(G)
    n=size(G.mm.A,1);
    p=size(G.mm.C,1);
    % Ruido del proceso
    wt = num2cell(chol(G.mm.Q)*randn(n,G.sim.N),1);
    % Ruido de la salida lineal (nomedible)
    vt = num2cell(chol(G.mm.R)*randn(p,G.sim.N),1);
    % Simulación del sistema
    xt(:,1)=G.mm.Icon.xin;
    for i=1:1:G.sim.N
        xt(:,i+1) = G.mm.A*xt(:,i) + G.mm.B*G.sim.ut{i} + wt{i};
        rt(1,i)   = G.mm.C*xt(:,i) + G.mm.D*G.sim.ut{i} + vt{i};
        if rt(1,i)>=G.sim.threshold
            yt(1,i)=G.sim.v2;
        else
            yt(1,i)=G.sim.v1;
        end
    end
    % Estado y salidas lineal y no lineal.
    G.sim.xt=num2cell(xt(:,1:G.sim.N),1);
    G.sim.rt=num2cell(rt,1);
    G.sim.yt=num2cell(yt,1);
    % Filtro de Kalman estándar.
    tic
    [KF] = standard_kalman(G);
    TP.tkf=toc;
    % Filtro de Kalman cuantizado.
    tic
    [QKF] = quantized_kalman(G);
    TP.tqkf=toc;
    % Filtro de Sumas de Gaussianas.
    tic
    [GSF]= gaussian_sum_filter(G);
    TP.tgsf=toc;
    % Filtro de particulas.
    tic
    PF= particle_filter(G);
    TP.tpf=toc;
end
```

```
function [G,KF,QKF,HF,PF,TP] = running_saturation_filtering(G)
    n=size(G.mm.A,1);
```

```
    p=size(G.mm.C,1);
    % Ruido del proceso
    wt = num2cell(chol(G.mm.Q)*randn(n,G.sim.N),1);
    % Ruido de la salida lineal (nomedible)
    vt = num2cell(chol(G.mm.R)*randn(p,G.sim.N),1);
    % Simulación del sistema
    xt(:,1)=G.mm.Icon.xin;
    for i=1:1:G.sim.N
        xt(:,i+1) =  G.mm.A*xt(:,i) + G.mm.B*G.sim.ut{i} +  wt{i};
        rt(i) = G.mm.C*xt(:,i) + G.mm.D*G.sim.ut{i} + vt{i} ;
        yt(1,i) = evaluate(G.mm.gz,rt(i));
    end
    % Estado y salidas lineal y no lineal
    G.sim.xt=num2cell(xt(:,1:G.sim.N),1);
    G.sim.rt=num2cell(rt,1);
    G.sim.yt=num2cell(yt,1);
    % Filtro de Kalman estándar.
    tic
    [KF] = standard_kalman(G);
    TP.tekf=toc;
    % Filtro de Kalman en cuadratura.
    tic
    [QKF] = quadrature_kalman(G);
    TP.tqkf=toc;
    % Filtro de Sumas de Gaussianas.
    tic
    [HF,~]= gaussian_sum_filter(G);
    TP.th=toc;
    % Filtro de particulas.
    tic
    [PF] = particle_filter(G);
    TP.tp=toc;
end
```

```
function [G,KF,QKF,HF,PF,TP] = running_deadzone_filtering(G)
    n=size(G.mm.A,1);
    p=size(G.mm.C,1);
    % Ruido del proceso
    wt = num2cell(chol(G.mm.Q)*randn(n,G.sim.N),1);
    % Ruido de la salida lineal (nomedible)
    vt = num2cell(chol(G.mm.R)*randn(p,G.sim.N),1);
    % Simulación del sistema
    xt(:,1)=G.mm.Icon.xin;
    for i=1:1:G.sim.N
        xt(:,i+1) =  G.mm.A*xt(:,i) + G.mm.B*G.sim.ut{i} +  wt{i};
        rt(i) = G.mm.C*xt(:,i) + G.mm.D*G.sim.ut{i} + vt{i} ;
        yt(1,i) = evaluate(G.mm.gz,rt(i));
    end
    % Estado y salidas lineal y no lineal
    G.sim.xt=num2cell(xt(:,1:G.sim.N),1);
    G.sim.rt=num2cell(rt,1);
    G.sim.yt=num2cell(yt,1);
    % Filtro de Kalman estándar.
    tic
    [KF] = standard_kalman(G);
    TP.tekf=toc;
    % Filtro de Kalman en cuadratura.
    tic
    [QKF] = quadrature_kalman(G);
    TP.tqkf=toc;
    % Filtro de Sumas de Gaussianas.
    tic
```

```
    [HF,~]= gaussian_sum_filter(G);
    TP.th=toc;
    % Filtro de particulas.
    tic
    [PF] = particle_filter(G);
    TP.tp=toc;
end
```

Funciones $g(\cdot)$ y cálculo de γ_{it} y ϕ_{it}

```
function [gz,gamma,phi,M]=choosing_g_function(fun)
    z=sym('z');
    % Función cudrática
    if strcmp(fun,'square')
        M=2;
        gz=matlabFunction((z)^2,'Vars',{z});
        gamma{1}=matlabFunction(sqrt(z),'Vars',{z});
        gamma{2}=matlabFunction(-sqrt(z),'Vars',{z});
        phi{1}=matlabFunction(0.5/sqrt(z),'Vars',{z});
        phi{2}=matlabFunction(0.5/sqrt(z),'Vars',{z});
    end
    % Función cúbica
    if strcmp(fun,'cubic')
        M=1;
        gz=matlabFunction(z^3,'Vars',{z});
        gamma{1}=matlabFunction(sign(z)*abs((z)^(1/3)),'Vars',{z});
        phi{1}=matlabFunction((1/3)*((z^2)^(1/3)),'Vars',{z});
    end
    % Función defina a trozos
    if strcmp(fun,'piece')
        M=2;
        sympref('HeavisideAtOrigin',1);
        gz=matlabFunction(heaviside(z)*z^2+heaviside(-z)*abs(z),'Vars',{z});
        gamma{1}=matlabFunction(sqrt(z),'Vars',{z});
        gamma{2}=matlabFunction(-z,'Vars',{z});
        phi{1}=matlabFunction(0.5/sqrt(z),'Vars',{z});
        phi{2}=matlabFunction(1+0*z,'Vars',{z});
    end
    % Función saturación
    if strcmp(fun,'sat')
        M=1; a=-3; b=3;
        gz = saturation('LinearInterval',[a,b]);
        gamma{1}=matlabFunction(z,'Vars',{z});
        phi{1}=matlabFunction(1+0*z,'Vars',{z});
    end
    % Función zona muerta
    if strcmp(fun,'dead')
        M=1; a=-3; b=3;
        gz = deadzone('ZeroInterval',[a,b]);
        gamma{1}=matlabFunction(z+a,'Vars',{z});
        phi{1}=matlabFunction(1+0*z,'Vars',{z});
        gamma{2}=matlabFunction(z+b,'Vars',{z});
        phi{2}=matlabFunction(1+0*z,'Vars',{z});
    end
end
```

Cómputo del sistema extendido

```
function [Gx]=extended_system(G)
    n=size(G.mm.A,1);    m=size(G.mm.B,2);    p=size(G.mm.C,1);
    Gx.mm.A=[G.mm.A,zeros(n,p);G.mm.C*G.mm.A,zeros(p,p)];
    Gx.mm.B=[G.mm.B,zeros(n,m);G.mm.C*G.mm.B,G.mm.D];
    Gx.mm.C=[zeros(p,n),eye(p,p)];
```

```
    Gx.mm.D=zeros(p,2*m);
    Gx.mm.Q=[G.mm.Q,G.mm.Q*G.mm.C';G.mm.C*G.mm.Q',G.mm.R+G.mm.C*G.mm.Q*G.mm.C'];
    Gx.mm.R=G.mm.P;
    Gx.mm.Icon.xin=[G.mm.Icon.xin;0.001*ones(p,1)];
    Gx.mm.Icon.Pin=blkdiag(G.mm.Icon.Pin,eye(p));
    us=cell2mat(G.sim.ut);
    u=[us; [us(2:end),0]];
    Gx.sim.ut=num2cell(u,1);
end
```

Filtro de Kalman Estándar

```
function [F] = standard_kalman(G)
% Valores iniciales.
F.xtmt{1}=G.mm.Icon.xin;
F.Stmt{1}=G.mm.Icon.Pin;
% Filtro de Kalman estándar
for t=1:1:G.sim.N
    % Ganancia de Kalman.
    F.K{t}=(F.Stmt{t}*G.mm.C')/(G.mm.R+G.mm.C*F.Stmt{t}*G.mm.C');
    % Actualización de la medición.
    F.xtt{t}=F.xtmt{t}+F.K{t}*(G.sim.yt{t}-G.mm.C*F.xtmt{t}-G.mm.D*G.sim.ut{t});
    F.Stt{t}=(eye(length(G.mm.C))-F.K{t}*G.mm.C)*F.Stmt{t};
    % Actualización temporal.
    F.xtmt{t+1}=G.mm.A*F.xtt{t}+G.mm.B*G.sim.ut{t};
    F.Stmt{t+1}=G.mm.Q+G.mm.A*F.Stt{t}*G.mm.A';
end
```

Filtro de Kalman Extendido

```
function [F,Fx] = extended_kalman(G0)
    N = G0.sim.N;
    yt=G0.sim.yt;
    n0= size(G0.mm.A,1);
    gz=G0.mm.gz;
    % Obtener el sistema extendido .
    [G]=extended_system(G0);
    ut=G.sim.ut;
    n= size(G.mm.A,1);
    m= size(G.mm.B,2);
    %
    x=sym('x',[n,1]);
    u=sym('u',[m,1]);
    % Función no lineal g(.) y Jacobiano de g(.).
    f=G.mm.A*x+G.mm.B*u;
    g=gz(G.mm.C*x);
    G.mm.fa=matlabFunction(f, 'Vars',{x,u});
    G.mm.ga=matlabFunction(g, 'Vars',{x});
    G.mm.At=matlabFunction(jacobian(f,x),'Vars', {x});
    G.mm.Ct=matlabFunction(jacobian(g,x),'Vars', {x});
    % Valores iniciales.
    Fx.xtmt{1}=G.mm.Icon.xin;
    Fx.Stmt{1}=G.mm.Icon.Pin;
    % Filtro de Kalman extendido.
    for t=1:1:N
        % Ganancia de Kalman.
        Fx.K{t}=(Fx.Stmt{t}*G.mm.Ct(Fx.xtmt{t})')/...
                (G.mm.R+G.mm.Ct(Fx.xtmt{t})*Fx.Stmt{t}*G.mm.Ct(Fx.xtmt{t})');
        % Actualización de la medición.
        Fx.xtt{t}=Fx.xtmt{t}+Fx.K{t}*(yt{t}-G.mm.ga(Fx.xtmt{t}));
        Fx.Stt{t}=(eye(n)-Fx.K{t}*G.mm.Ct(Fx.xtmt{t}))*Fx.Stmt{t};
        % Actualización temporal.
        Fx.xtmt{t+1}=G.mm.fa(Fx.xtt{t},ut{t});
```

```
        Fx.Stmt{t+1}=G.mm.Q+G.mm.At(Fx.xtt{t})*Fx.Stt{t}*G.mm.At(Fx.xtt{t})';
    end
    % Almacenar las estimaciones.
    for t=1:1:N
        F.xtt{t}=Fx.xtt{t}(1:n0,1);
        F.rtt{t}=Fx.xtt{t}(1+n0:end,1);
        F.Stt{t}=Fx.Stt{t}(1:n0,1:n0);
        F.Rtt{t}=Fx.Stt{t}(1+n0:end,1+n0:end);
    end
end
```

Filtro de Kalman en Cuadratura

```
function [F] =quadrature_kalman(G0)
    N = G0.sim.N;
    yt=G0.sim.yt;
    gz=G0.mm.gz;
    n0= size(G0.mm.A,1);
    fun=G0.mm.fun;
    % Obtener el sistema extendido.
    [G]=extended_system(G0);
    ut=G.sim.ut;
    G.mm.fun=fun;
    n=size(G.mm.A,1);
    m=size(G.mm.B,2);
    p=size(G.mm.C,1);
    %
    x=sym('x',[n,1]);
    u=sym('u',[m,1]);
    %
    f=G.mm.A*x+G.mm.B*u;
    G.mm.fa=matlabFunction(f, 'Vars',{x,u});
    if ~strcmp(G.mm.fun,'sat') && ~strcmp(G.mm.fun,'dead')
        g=gz(G.mm.C*x);
        G.mm.ga=matlabFunction(g, 'Vars',{x});
    end
    % Ontiene los puntos de la cuadratura de Gauss-Hermite.
    GH=6; GHn=GH^n;
    [wn,Xn] = gauss_hermite(GH);
    w=prod(wn(fliplr(fullfact(ones(1,n)*GH))),2);
    X=Xn(fliplr(fullfact(ones(1,n)*GH)))';
    % Valores iniciales.
    Fx.xtmt{1}=G.mm.Icon.xin;
    Fx.Stmt{1}=G.mm.Icon.Pin;
    % Filtro de Kalman en cuadratura.
    for t=1:1:N
        Stmt=chol(Fx.Stmt{t});
        for j=1:1:GHn
            Xl(:,j)=Stmt*X(:,j)+Fx.xtmt{t};
        end
        for j=1:1:GHn
            if ~strcmp(G.mm.fun,'sat') && ~strcmp(G.mm.fun,'dead')
                Zl(:,j)=G.mm.ga(Xl(:,j));
            else
                Zl(:,j)=evaluate(gz,Xl(n+1:end,j));
            end
        end
        zhat=zeros(p,1);
        for j=1:1:GHn
            zhat=zhat+w(j)*Zl(:,j);
        end
        Phat=zeros(p,p);
        for j=1:1:GHn
```

```
            Phat=Phat+w(j)*(Zl(:,j)-zhat)*(Zl(:,j)-zhat)';
        end
        Pzz=G.mm.R+Phat;
        Pxz=zeros(n,p);
        for j=1:1:GHn
            Pxz=Pxz+w(j)*(Xl(:,j)-Fx.xtmt{t})*(Zl(:,j)-zhat)';
        end
        Fx.K{t}=Pxz/Pzz;
        % Actualización de la medición.
        Fx.xtt{t}=Fx.xtmt{t}+Fx.K{t}*(yt{t}-zhat);
        Fx.Stt{t}=Fx.Stmt{t}-Fx.K{t}*Pzz*Fx.K{t}';
        % Actualización temporal.
        Fx.xtmt{t+1}=G.mm.fa(Fx.xtt{t},ut{t});
        Fx.Stmt{t+1}=G.mm.Q+G.mm.A*Fx.Stt{t}*G.mm.A';
    end
    % Almacenar las estimaciones
    F=Fx;
    for t=1:1:N
        F.xtt{t}=Fx.xtt{t}(1:n0,1);
        F.rtt{t}=Fx.xtt{t}(1+n0:end,1);
        F.Stt{t}=abs(Fx.Stt{t}(1:n0,1:n0));
        F.Rtt{t}=Fx.Stt{t}(1+n0:end,1+n0:end);
    end
end
%% Función que calcula los puntos de la cuadratura de Gauss-Hermite
function [w,X] = gauss_hermite(m)
    J=zeros(m,m);
    for i=1:1:m-1
        J(i,i+1)=sqrt(i/2);
    end
    J=J+J';
    [eigvecs,eigvals] = eig(J);
    X=sqrt(2)*diag(eigvals)';
    w=(eigvecs(1,:)).^2;
end
```

Filtro de Kalman Cuantizado

```
function [F] = quantized_kalman(G)
    % Valores iniciales.
    F.xtmt{1}=G.mm.Icon.xin;
    F.Stmt{1}=G.mm.Icon.Pin;
    % Filtro de Kalman cuantizado
    for t=1:1:G.sim.N
        % Ganancia de Kalman.
        F.K{t}=(F.Stmt{t}*G.mm.C')/(G.mm.R+G.mm.C*F.Stmt{t}*G.mm.C');
        % Actualización de la medición.
        F.xtt{t}=F.xtmt{t}+F.K{t}*(G.sim.yt{t}-Q(G,G.mm.C*F.xtmt{t}+G.mm.D*G.sim.ut{t}));
        F.Stt{t}=(eye(length(G.mm.C))-F.K{t}*G.mm.C)*F.Stmt{t};
        % Actualización temporal.
        F.xtmt{t+1}=G.mm.A*F.xtt{t}+G.mm.B*G.sim.ut{t};
        F.Stmt{t+1}=G.mm.Q+G.mm.A*F.Stt{t}*G.mm.A';
    end
end
%% Función cuantizador binario
function q=Q(G,zt)
   if  zt>=G.sim.threshold
       q=G.sim.v2;
   else
       q=G.sim.v1;
   end
end
```

Filtro de Partículas

```
function [PF] = particle_filter(G)
    n= size(G.mm.A,1);
    %
    xpar = repmat(G.mm.Icon.xin,1,G.sim.Npar)+chol(G.mm.Icon.Pin)*randn(n,G.sim.Npar);
    if ~strcmp(G.mm.fun,'bin')
        nh=chol(G.mm.P)*randn(1,G.sim.Nsamp);
    end
    for t=1:1:G.sim.N
        % Cómputo de los pesos normalizados para la función saturación
        w=zeros(1,G.sim.Npar);
        if strcmp(G.mm.fun,'sat')
            if G.sat.v1~=G.sim.yt{t} && G.sim.yt{t}~=G.sat.v2
                temp1=real(G.mm.phi{1}(repmat(G.sim.yt{t},1,G.sim.Nsamp)-nh));
                Tm1=repmat(temp1,G.sim.Npar,1);
                temp2=real(G.mm.gamma{1}(repmat(G.sim.yt{t},1,G.sim.Nsamp)-nh));
                Tm2=repmat(temp2,G.sim.Npar,1);
                temp3=G.mm.C*xpar+G.mm.D*G.sim.ut{t};
                Tm3=repmat(temp3',1,G.sim.Nsamp);
                Tm4=Tm2-Tm3;
                Tm5=normpdf(Tm4,0,sqrt(G.mm.R));
                w=w+sum(Tm1.*Tm5,2)'/G.sim.Nsamp;
            else
                if G.sim.yt{t}==G.sat.v2  %L
                    a= G.sat.v2;
                    b=Inf;
                end
                if G.sim.yt{t}==G.sat.v1 %l
                    a=-Inf;
                    b=G.sat.v1;
                end
                lim_a = repmat(a,1,G.sim.Npar) - G.mm.C*xpar-G.mm.D*G.sim.ut{t};
                lim_b = repmat(b,1,G.sim.Npar) - G.mm.C*xpar-G.mm.D*G.sim.ut{t};
                w = transpose(mvncdf(lim_b',0,G.mm.R) - mvncdf(lim_a',0,G.mm.R));
            end
        % Cómputo de los pesos normalizados para la función cuantizador binario
        elseif strcmp(G.mm.fun,'bin')
            if G.sim.yt{t}==G.sim.v2
                a= G.sim.threshold;
                b=Inf;
            end
            if G.sim.yt{t}==G.sim.v1
                a=-Inf;
                b=G.sim.threshold;
            end
            lim_a = repmat(a,1,G.sim.Npar) - G.mm.C*xpar-G.mm.D*G.sim.ut{t};
            lim_b = repmat(b,1,G.sim.Npar) - G.mm.C*xpar-G.mm.D*G.sim.ut{t};
            w = transpose(mvncdf(lim_b',0,G.mm.R) - mvncdf(lim_a',0,G.mm.R));
        % Cómputo de los pesos normalizados para la función zona muerta
        elseif strcmp(G.mm.fun,'dead')
            if G.sim.yt{t}<0
                temp1=real(G.mm.phi{1}(repmat(G.sim.yt{t},1,G.sim.Nsamp)-nh));
                Tm1=repmat(temp1,G.sim.Npar,1);
                temp2=real(G.mm.gamma{1}(repmat(G.sim.yt{t},1,G.sim.Nsamp)-nh));
                Tm2=repmat(temp2,G.sim.Npar,1);
                temp3=G.mm.C*xpar+G.mm.D*G.sim.ut{t};
                Tm3=repmat(temp3',1,G.sim.Nsamp);
                Tm4=Tm2-Tm3;
                Tm5=normpdf(Tm4,0,sqrt(G.mm.R));
                w=w+sum(Tm1.*Tm5,2)'/G.sim.Nsamp;
            elseif G.sim.yt{t}>0
```

```
                temp1=real(G.mm.phi{2}(repmat(G.sim.yt{t},1,G.sim.Nsamp)-nh));
                Tm1=repmat(temp1,G.sim.Npar,1);
                temp2=real(G.mm.gamma{2}(repmat(G.sim.yt{t},1,G.sim.Nsamp)-nh));
                Tm2=repmat(temp2,G.sim.Npar,1);
                temp3=G.mm.C*xpar+G.mm.D*G.sim.ut{t};
                Tm3=repmat(temp3',1,G.sim.Nsamp);
                Tm4=Tm2-Tm3;
                Tm5=normpdf(Tm4,0,sqrt(G.mm.R));
                w=w+sum(Tm1.*Tm5,2)'/G.sim.Nsamp;
            else
                a=G.dead.v1;
                b=G.dead.v2;
                lim_a = repmat(a,1,G.sim.Npar) - G.mm.C*xpar-G.mm.D*G.sim.ut{t};
                lim_b = repmat(b,1,G.sim.Npar) - G.mm.C*xpar-G.mm.D*G.sim.ut{t};
                w = transpose(mvncdf(lim_b',0,G.mm.R) - mvncdf(lim_a',0,G.mm.R));
            end
        % Cómputo de los pesos normaizados para la función monótona a trozos
        else
            for i=1:1:G.sim.M
                temp1=real(G.mm.phi{i}(repmat(G.sim.yt{t},1,G.sim.Nsamp)-nh));
                Tm1=repmat(temp1,G.sim.Npar,1);
                temp2=real(G.mm.gamma{i}(repmat(G.sim.yt{t},1,G.sim.Nsamp)-nh));
                Tm2=repmat(temp2,G.sim.Npar,1);
                temp3=G.mm.C*xpar+G.mm.D*G.sim.ut{t};
                Tm3=repmat(temp3',1,G.sim.Nsamp);
                Tm4=Tm2-Tm3;
                Tm5=normpdf(Tm4,0,sqrt(G.mm.R));
                w=w+sum(Tm1.*Tm5,2)'/G.sim.Nsamp;
            end
        end
        w = w/sum(w);

        % Cómputo del estado estimado y su covarianza del error de estimación
        PF.xtt{t} = sum(w.*xpar,2);
        PF.Stt{t}=zeros(n,n);
        for i=1:1:G.sim.Npar
            PF.Stt{t}= PF.Stt{t} + w(i)*(xpar(:,i)-PF.xtt{t})*(xpar(:,i)-PF.xtt{t})';
        end
        % Resampling
        index = resampling(w);
        xpar = xpar(:,index);
        PF.xpres{t} = xpar;
        % Predecir particulas
        xpar = G.mm.A*xpar + G.mm.B*G.sim.ut{t} + chol(G.mm.Q)*randn(n,G.sim.Npar);
    end
end
%% Función resampling
function i=resampling(q)
    qc=cumsum(q);
    M=length(q);
    u=([0:M-1]+rand(1))/M;
    i=zeros(1,M);    k=1;
    for j=1:M
        while (qc(k)<u(j))
            k=k+1;
        end
        i(j)=k;
    end
end
```

Filtro suma de gaussianas

```
function [F,yx]=gaussian_sum_filter(G)
```

```
Icon.mu{1}=G.mm.Icon.xin;
Icon.Sigma{1}=G.mm.Icon.Pin;
Icon.w{1}=1;
TU = Icon;
% Guarda la PDF de predicción al instante t=1.
F.timeup{1} = TU;
% Guarda las estadisticas de predicción al instante t=1.
F.xtmt{1}=TU.mu{1};
F.Stmt{1}=TU.Sigma{1};
%
n=size(G.mm.A,1);
p=size(G.mm.C,1);
Mt_tm1=length(TU.w);
% Obtiene los puntos de la cuadratura de Gauss-Legendre.
q = quadGaussLegendre(G.sim.L);
wtau=q.Weights;
xtau=q.Points;
% Calcula el modelo p(yt|xt).
[yx] = model_pytxt(G,wtau,xtau);
Kj=G.sim.M*G.sim.L;
% Filtro Suma de Gaussianas.
for t=1:1:G.sim.N
    xij=yx.xij{t};
    betaj=yx.betaj{t};
    % Actualización de la medición.
    ell=0;
    for k=1:1:Mt_tm1
        for j=1:1:Kj
            ell=ell+1;
            my=xij(j)-G.mm.C*TU.mu{k}-G.mm.D*G.sim.ut{t};
            Zy=G.mm.R+G.mm.C*TU.Sigma{k}*G.mm.C';
            K=(TU.Sigma{k}*G.mm.C')/Zy;
            vz(:,:,ell)=(eye(n)-K*G.mm.C)*TU.Sigma{k};
            md(:,ell)=TU.mu{k}+K*my;
            ex=-0.5*(my'*(Zy\my));
            ps(ell)=log(betaj(j))+log(TU.w{k})-0.5*log(det(2*pi*Zy))+ex;
        end
    end
    % Normalización de los pesos.
    ps=softmax(ps);
    %
    Mtt=G.sim.L*G.sim.M*Mt_tm1;
    % Guarda la PDF de filtraje al instante t.
    MU=[];
    MU.w=num2cell(ps,1);
    MU.mu=num2cell(md,1);
    for s=1:1:Mtt
        MU.Sigma{s}=vz(:,:,s);
    end
    F.measup{t} = MU;
    % Actualización temporal.
    for ell=1:1:Mtt
        k=ell;
        m(:,k)=G.mm.A*MU.mu{ell}+G.mm.B*G.sim.ut{t};
        S(:,:,k)=G.mm.A*MU.Sigma{ell}*G.mm.A'+G.mm.Q;
        w(k)=MU.w{ell};
    end
    % Guarda la PDF de predicción al instante t.
    TM=[];
    TM.w=num2cell(w,1);
    TM.mu=num2cell(m,1);
    for s=1:1:Mtt
```

```
            TM.Sigma{s}=S(:,:,s);
        end
        F.timeup{t+1}=TM;
        % Reducción de Gaussianas.
        if Mtt>=G.sim.Gmin
            [w,m,S]=kullbackL(G.sim.minG,G.sim.maxG,0.5,w,m,S);
        end
        Mt_tm1=length(w);
        % PDF de predicción reducida para la siguiente iteración.
        TU=[];
        TU.w=num2cell(w,1);
        TU.mu=num2cell(m,1);
        for s=1:1:Mt_tm1
            TU.Sigma{s}=S(:,:,s);
        end
        % Guarda las estadisticas de filtraje al instante t.
        [x,P]=gmm2muP(MU);
        F.xtt{t}=x;
        F.Stt{t}=P;
        % Guarda las estadisticas de predicción al instante t.
        [x,P]=gmm2muP(TM);
        F.xtmt{t+1}=x;
        F.Stmt{t+1}=P;
        %
        clear ps md vz w m S;
    end
end

%% Función que calcula la media y covarianza dada una GMM.
function [x,P]=gmm2muP(Data)
    alphai=Data.w;
    mui=Data.mu;
    Pi=Data.Sigma;
    n=size(Pi{1},1);
    x=zeros(n,1);
    P=zeros(n,n);
    tam=length(alphai);
    if tam==1
        x=mui{1};
        P=Pi{1};
    else
        for i=1:1:tam
            x=x+alphai{i}*mui{i};
        end
        for i=1:1:tam
            P=P+alphai{i}*(Pi{i}+(mui{i}-x)*(mui{i}-x)');
        end
    end
end
%% Función que genera los puntos de la cuadratura de Gauss-Legendre.
% Ethan Kubatko (2021). quadGaussLegendre.
% (https://www.mathworks.com/matlabcentral/fileexchange/94560-quadgausslegendre),
% MATLAB Central File Exchange. Retrieved September 20, 2021.
function Q = quadGaussLegendre(n,varargin)
    vn = @(x)validateattributes(x,{'numeric'},{'scalar','integer','>=',1});
    vD = @(x)validateattributes(x,{'numeric'},{'numel',2,'increasing'});
    ip = inputParser;
    ip.addRequired('n',vn); ip.addParameter('Domain',[-1,1],vD)
    ip.parse(n,varargin{:}); ip.Results; Domain = ip.Results.Domain;
    % Compute the weights and points.
    % Define the coefficients of the three-term recurrence relationship.
    a = @(n)(2*n+1)./(n+1);
```

```
    b = @(n)0;
    c = @(n)n./(n+1);
    % Constructe the symmetric tridiagonal matrix
    A = -b(0:n-1)./a(0:n-1); B = sqrt(c(1:n-1)./(a(0:n-2).*a(1:n-1)));
    J = diag(B,1) + diag(A) + diag(B,-1);
    % Compute the eigenvalues and eigenvectors.
    [V,D] = eig(J,'vector');
    % Save (sorted) points and weights.
    [Q.Points,I] = sort(D);
    Q.Weights = (2*V(1,I).^2)';
    % Note: The next three lines insure zero is zero and the points and weights
    % are perfectly symmetric.
    Q.Points(abs(Q.Points)<10*eps) = 0;
    Q.Points(ceil(end/2)+1:end) = -flipud(Q.Points(1:floor(end/2)));
    Q.Weights(ceil(end/2)+1:end) = flipud(Q.Weights(1:floor(end/2)));
    % Transformation of points and weights if [a,b]~=[-1,1].
    if ~isequal(Domain,[-1,1])
        Q.Points  = (Domain(2)-Domain(1))/2*Q.Points + (Domain(1)+Domain(2))/2;
        Q.Weights = (Domain(2)-Domain(1))/2*Q.Weights;
    end
    % Assign properties.
    Q.Properties.Degree = 2*n-1;
    Q.Properties.Type   = 'Gauss-Legendre';
    Q.Properties.Domain = Domain;
end
%% Función que implementa el algoritmo de reducción de Sumas de Gaussianas.
function [pso,med,var]=kullbackL(LI,LS,LD,ps,md,vz)
    N=length(ps);
    k=N;  B=Inf*ones(N,N);
    B = boundS(ps,md,vz,B);
    while (k>LS || (k>LI && min(min(B))<LD))
        [iaste,jaste]=find(B==min(min(B)));  iast=iaste(1); jast=jaste(1);
        [w,mu,P] = merged(ps(iast),ps(jast),md(:,iast),md(:,jast),vz(:,:,iast),vz(:,:,jast));
        ps(iast)=w;    md(:,iast)=mu;    vz(:,:,iast)=P;
        ps(jast)=[];   md(:,jast)=[];    vz(:,:,jast)=[];
        B(jast,:)=[];    B(:,jast)=[];
        B = bound(ps,md,vz,iast,B);
        k=k-1;
    end
    pso=ps;     med=md;     var=vz;
end
function [wij,muij,Pij] = merged(psi,psj,mdi,mdj,vzi,vzj)
    wij=psi+psj;
    wiIij = (psi)/(psi+psj);
    wjIij = (psj)/(psi+psj);
    muij = wiIij*mdi+wjIij*mdj;
    Pij = wiIij*vzi + wjIij*vzj + wiIij*wjIij*(mdi-mdj)*(mdi-mdj)';
end
function B = boundS(ps,md,vz,B)
    L=size(B,1);
    for i=1:1:L-1
        for j=i+1:1:L
            [wij,~,Pij] = merged(ps(i),ps(j),md(:,i),md(:,j),vz(:,:,i),vz(:,:,j));
            B(i,j)=0.5*(wij*log(det(Pij))-ps(i)*log(det(vz(:,:,i)))-ps(j)*log(det(vz(:,:,j))))
                ;
        end
    end
end
function B = bound(ps,md,vz,iast,B)
    N=size(B,1);
    for j=1:1:N
        if iast<j
```

```
            [wij,~,Pij] = merged(ps(iast),ps(j),md(:,iast),md(:,j),vz(:,:,iast),vz(:,:,j));
            B(iast,j)=0.5*(wij*log(det(Pij))-ps(iast)*log(det(vz(:,:,iast)))-ps(j)*log(det(vz
                (:,:,j))));
        end
    end
    jiast=iast;
    for i=1:1:N
        if jiast>i
            [wij,~,Pij] = merged(ps(i),ps(jiast),md(:,i),md(:,jiast),vz(:,:,i),vz(:,:,jiast));
            B(i,jiast)=0.5*(wij*log(det(Pij))-ps(i)*log(det(vz(:,:,i)))-ps(jiast)*log(det(vz
                (:,:,jiast))));
        end
    end
end
%% Función que permite normalizar los pesos en el algoritmo GSF
% P. Blanchard, D. J. Higham, and N. J. Higham.
% Accurately computing the log-sum-exp and softmax functions.
% IMA J. Numer. Anal., Advance access, 2020.
function [sm,lse] = softmax(x)
    if ~isvector(x), error('Input x must be a vector.'), end
    n = length(x);
    e = zeros(1,n);
    [xmax,k] = max(x); a = xmax;
    s = 0;
    for i = 1:n
        e(i) = exp(x(i)-xmax);
        if i ~= k
           s = s + e(i);
        end
    end
    if nargout > 1
       lse = a + log1p(s);
    end
    sm = e/(1+s);
end
```

Modelo de $p(y_t|x_t)$

```
function [yx] = model_pytxt(G,wtau,xtau)
    % Función saturación
    if strcmp(G.mm.fun,'sat')
        [yx] = hw_pytxt_sat(G,wtau,xtau);
    % Función cuantizador binario
    elseif strcmp(G.mm.fun,'bin')
        [yx] = hw_pytxt_bin(G,wtau,xtau);
    % Función zona muerta
    elseif strcmp(G.mm.fun,'dead')
        [yx] = hw_pytxt_dead(G,wtau,xtau);
    % Función monótona a trozos
    else
        [yx] = hw_pytxt_anyg(G,wtau,xtau);
    end
end
%% Cálculo del modelo p(yt|xt) para la función monótona a trozos
function [yx] = hw_pytxt_anyg(G,wtau,xtau)
    ldtau=(xtau)./(1-xtau.^2);
    fac=(1+xtau.^2)./((1-xtau.^2).^2);
    gamma=G.mm.gamma;
    phi=G.mm.phi;
    for t=1:1:G.sim.N
        j=0;
        betaj=zeros(1,G.sim.L*G.sim.M);
        xij=zeros(1,G.sim.L*G.sim.M);
```

```
        for tau0=1:1:G.sim.L
            for i=1:1:G.sim.M
                j=j+1;
                hieval=real(phi{i}(G.sim.yt{t}-ldtau(tau0)));
                xij(j)=real(gamma{i}(G.sim.yt{t}-ldtau(tau0)));
                betaj(j)=wtau(tau0)*hieval*normpdf(ldtau(tau0),0,sqrt(G.mm.P))*fac(tau0);
            end
        end
        yx.betaj{t}=betaj;
        yx.xij{t}=xij;
    end
end
%% Cálculo del modelo p(yt|xt) para la función saturación
function [yx] = hw_pytxt_sat(G,wtau,xtau)
    ldtau=(xtau)./(1-xtau.^2);
    fac=(1+xtau.^2)./((1-xtau.^2).^2);
    gamma=G.mm.gamma;
    phi=G.mm.phi;
    for t=1:1:G.sim.N
        if G.sat.v1~=G.sim.yt{t} && G.sim.yt{t}~=G.sat.v2
            j=0;
            betaj=zeros(1,G.sim.L*G.sim.M);
            xij=zeros(1,G.sim.L*G.sim.M);
            for tau0=1:1:G.sim.L
                for i=1:1:G.sim.M
                    j=j+1;
                    hieval=real(phi{i}(G.sim.yt{t}-ldtau(tau0)));
                    xij(j)=real(gamma{i}(G.sim.yt{t}-ldtau(tau0)));
                    betaj(j)=wtau(tau0)*hieval*normpdf(ldtau(tau0),0,sqrt(G.mm.P))*fac(tau0);
                end
            end
        else
            if G.sim.yt{t}==G.sat.v2
                xij=G.sat.v2+((1-xtau)./(1+xtau));
                betaj=(2*wtau./((1+xtau).^2))';
            end
            if G.sim.yt{t}==G.sat.v1
                xij=G.sat.v1-((1-xtau)./(1+xtau));
                betaj=(2*wtau./((1+xtau).^2))';
            end
        end
        yx.betaj{t}=betaj;
        yx.xij{t}=xij;
    end
end
%% Cálculo del modelo p(yt|xt) para la función zona muerta
function [yx] = hw_pytxt_dead(G,wtau,xtau)
    ldtau=(xtau)./(1-xtau.^2);
    fac=(1+xtau.^2)./((1-xtau.^2).^2);
    gamma=G.mm.gamma;
    phi=G.mm.phi;
    for t=1:1:G.sim.N
        betaj=zeros(1,G.sim.L);
        xij=zeros(1,G.sim.L);
        if G.sim.yt{t}<0
            for j=1:1:G.sim.L
                hieval=real(phi{1}(G.sim.yt{t}-ldtau(j)));
                xij(j)=real(gamma{1}(G.sim.yt{t}-ldtau(j)));
                betaj(j)=wtau(j)*hieval*normpdf(ldtau(j),0,sqrt(G.mm.P))*fac(j);
            end
        elseif G.sim.yt{t}>0
            for j=1:1:G.sim.L
```

```
                hieval=real(phi{2}(G.sim.yt{t}-ldtau(j)));
                xij(j)=real(gamma{2}(G.sim.yt{t}-ldtau(j)));
                betaj(j)=wtau(j)*hieval*normpdf(ldtau(j),0,sqrt(G.mm.P))*fac(j);
            end
        else
            xij=((G.dead.v2-G.dead.v1)/2)*xtau + (G.dead.v1+G.dead.v2)/2;
            betaj=wtau*(G.dead.v2-G.dead.v1)/2;
        end
        yx.betaj{t}=betaj;
        yx.xij{t}=xij;
    end
end
%% Cálculo del modelo p(yt|xt) para la función cuantizador binario
function [yx] = hw_pytxt_bin(G,wtau,xtau)
   for t=1:1:G.sim.N
        if G.sim.yt{t}==G.sim.v2
            yx.xij{t}=G.sim.threshold+((1-xtau)./(1+xtau));
            yx.betaj{t}=2*wtau./((1+xtau).^2);
        end
        if G.sim.yt{t}==G.sim.v1
            yx.xij{t}=G.sim.threshold-((1-xtau)./(1+xtau));
            yx.betaj{t}=2*wtau./((1+xtau).^2);
        end
    end
end
```

Bibliografía

Abramowitz, M., Stegun, I. A., and Romer, R. H. (1988). Handbook of mathematical functions with formulas, graphs, and mathematical tables.

Agüero, J. C., Tang, W., Yuz, J. I., Delgado, R., and Goodwin, G. C. (2012). Dual time-frequency domain system identification. *Automatica*, 48(12):3031–3041.

Ahmed, A., Naqvi, S. A. A., Watling, D., and Ngoduy, D. (2019). Real-Time Dynamic Traffic Control Based on Traffic-State Estimation. *Transportation Research Record*, 2673(5):584–595.

Ahwiadi, M. and Wang, W. (2020). An Adaptive Particle Filter Technique for System State Estimation and Prognosis. *IEEE Transactions on Instrumentation and Measurement*, 69(9):6756–6765.

Albornoz, R., Carvajal, R., and Agüero, J. C. (2019). A novel bayesian filtering method for systems with quantized output data. In *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–7.

Alspach, D. L. and Sorenson, H. W. (1972). Nonlinear bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448.

Anderson, B. D. O. and Moore, J. B. (1979). *Optimal Filtering*. Prentice-Hall, Inc.

Anderson, B. D. O. and Moore, J. B. (2007). *Optimal control: linear quadratic methods*. Courier Corporation.

Andrieu, C. and Doucet, A. (2002). Particle filtering for partially observed Gaussian state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):827–836.

Arasaratnam, I., Haykin, S., and Elliott, R. J. (2007). Discrete-time nonlinear filtering algorithms using gauss–hermite quadrature. *Proceedings of the IEEE*, 95(5):953–977.

Balenzuela, M. P., Dahlin, J., Bartlett, N., Wills, A. G., Renton, C., and Ninness, B. (2019). Accurate Gaussian Mixture Model Smoothing using a Two-Filter Approach. *Proceedings of the IEEE Conference on Decision and Control*, 2018-Decem(Cdc):694–699.

Bhattacharjee, A., Sengupta, A., and Sutradhar, A. (2010). Nonparametric modeling of glucose-insulin process in IDDM patient using Hammerstein-Wiener model. In *2010 11th International Conference on Control Automation Robotics & Vision*, pages 2266–2271.

Billings, S. A. (2013). *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer.

Bittner, G., Orellana, R., Carvajal, R., and Agüero, J. C. (2019). Maximum Likelihood identification for Linear Dynamic Systems with finite Gaussian mixture noise distribution. In *IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies, CHILECON 2019*.

Blanchard, P., Higham, D. J., and Higham, N. J. (2020). Accurately computing the log-sum-exp and softmax functions. *IMA Journal of Numerical Analysis*.

Bonvini, M., Sohn, M. D., Granderson, J., Wetter, M., and Piette, M. A. (2014). Robust on-line fault detection diagnosis for HVAC components based on nonlinear state estimation techniques. *Applied Energy*, 124.

Brown, M. and Harris, C. J. (1994). *Neurofuzzy adaptive modelling and control*. Prentice Hall.

Bucy, R. S. and Joseph, P. D. (1968). *Filtering for Stochastic Processes with Applications to Guidance*. John Wiley & Sons, Inc.

Cedeño, A. L., Albornoz, R., Carvajal, R., Godoy, B. I., and Agüero, J. C. (2021). A Two-Filter Approach for State Estimation Utilizing Quantized Output Data. *Sensors*, 21(22):7675.

Cedeño, A. L., Orellana, R., Carvajal, R., and Agüero, J. C. (2020). EM-based identification of static errors-in-variables systems utilizing Gaussian Mixture models. *IFAC-PapersOnLine*, 53(2):863–868.

Cessenat, M. (2018). *Mathematical Modelling of Physical Systems*. Springer International Publishing.

Chang, X. H. and Liu, Y. (2020). Robust H Filtering for Vehicle Sideslip Angle With Quantization and Data Dropouts. *IEEE Transactions on Vehicular Technology*, 69(10):10435–10445.

Chiang, K. W., Duong, T. T., Liao, J. K., Lai, Y. C., Chang, C. C., Cai, J. M., and Huang, S. C. (2012). On-Line Smoothing for an Integrated Navigation System with Low-Cost MEMS Inertial Sensors. *Sensors*, 12(12).

Cohen, H. (2011). *Numerical approximation methods*. Springer.

Corbetta, M., Sbarufatti, C., Giglio, M., and Todd, M. D. (2018). Optimization of nonlinear, non-Gaussian Bayesian filtering for diagnosis and prognosis of monotonic degradation processes. *Mechanical Systems and Signal Processing*, 104:305–322.

Cosme, E., Verron, J., Brasseur, P., Blum, J., and Auroux, D. (2012). Smoothing problems in a Bayesian framework and their linear Gaussian solutions. *Monthly Weather Review*, 140(2):683–695.

Crassidis, J. L. and Junkins, J. L. (2012). *Optimal Estimation of Dynamic Systems*. Chapman and Hall/CRC.

Curiac, D. (2016). Towards wireless sensor, actuator and robot networks: Conceptual framework, challenges and perspectives. *Journal of Network and Computer Applications*, 63:14–23.

Davis, P. F. and Rabinowitz, P. (1984). Methods of numerical integration.

Ding, D., Han, Q. L., Ge, X., and Wang, J. (2021). Secure State Estimation and Control of Cyber-Physical Systems: A Survey. *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, 51(1):176–190.

Doucet, A., Godsill, S., and Andrieu, C. (2000a). On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208.

Doucet, A., Godsill, S., and Andrieu, C. (2000b). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, 10(3):197–208.

Evans, D. L., Drew, J. H., Glen, A. G., and Leemis, L. M. (2008). *Computational Probability: Algorithms and Applications in the Mathematical Sciences Series, 2008*. Springer.

Friedland, B. (2012). *Control system design: an introduction to state-space methods*. Courier Corporation.

Frühwirth, S. (2006). *Finite Mixture and Markov Switching Models*, volume 796. Springer.

Frühwirth, S., Celeux, G., and Robert, C. P. (2019). *Handbook of Mixture Analysis*. CRC press.

Gelb, A., Kasper, J., Nash, R., Price, C., and Sutherland, A. (1974). *Applied optimal estimation*. Cambridge, MA: MIT Press.

Gençay, R., Selçuk, F., and Whitcher, B. J. (2001). *An introduction to wavelets and other filtering methods in finance and economics*. Elsevier.

Gersho, A. and Gray, R. M. (2012). *Vector Quantization and Signal Compression*, volume 159. Springer Science & Business Media.

Gibson, S. and Ninness, B. (2005). Robust maximum-likelihood estimation of multivariable dynamic systems. *Automatica*, 41(10):1667–1682.

Giri, F. and Bai, E. W. (2010). *Block-oriented Nonlinear System Identification*, volume 1. Springer.

Golub, G. H. and Welsch, J. H. (1969). Calculation of Gauss quadrature rules. *Mathematics of computation*, 23(106):221–230.

Gómez, J. C. and Sad, G. D. (2020). A State Observer from Multilevel Quantized Outputs. In *2020 Argentine Conference on Automatic Control (AADECA)*, pages 1–6.

Goodwin, G. C., Graebe, S. F., and Salgado, M. E. (2001). *Control system design*. Upper Saddle River, NJ: Prentice Hall,.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993a). Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F - Radar and Signal Processing*, 140(2):107–113.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993b). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET.

Grewal, M. S. and Andrews, A. P. (2014). *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons.

Grimble, M. J. and Majecki, P. (2020). *Nonlinear Industrial Control Systems*. Springer.

Gruyitch, L. T. (2019). *Control of Linear Systems Observability and Controllability of General Linear Systems*. CRC Press.

Guo, S. and Han, L. (2018). *Stability and Control of Nonlinear Time-varying Systems*. Springer.

Hale, N. and Townsend, A. (2013). Fast and accurate computation of Gauss–Legendre and Gauss–Jacobi quadrature nodes and weights. *SIAM Journal on Scientific Computing*, 35(2):A652–A674.

Han, Y., Cui, M., and Liu, S. (2020). Optimal Sensor and Relay Nodes Power Scheduling for Remote State Estimation with Energy Constraint. *Sensors*, 20(4).

Harville, D. A. (1998). *Matrix Algebra from a Statistician's Perspective*. Taylor & Francis.

Hendricks, E., Jannerup, O., and Haase, P. (2008). *Linear Systems Control Deterministic and Stochastic Methods*. Springer Science & Business Media.

Hostettler, R. (2015). A two filter particle smoother for wiener state-space systems. In *2015 IEEE Conference on Control Applications (CCA)*, pages 412–417. IEEE.

Hu, T. and Lin, Z. (2001). *Control systems with actuator saturation: analysis and design*. Springer Science & Business Media.

Huang, B. and Wang, Q. G. (2006). Overview of Emerging Bayesian Approach To Nonlinear System Identification. *SICOP Round Tables on Nonlinear Model Identification, International Workshop on Solving Industrial Control and Optimization Problems*, 1(780).

Huang, C., Shen, B., Zou, L., and Shen, Y. (2021). Event-Triggering State and Fault Estimation for a Class of Nonlinear Systems Subject to Sensor Saturations. *Sensors*, 21(4).

Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory*. Courier Corporation.

Jeong, H., Park, B., Park, S., Min, H., and Lee, S. (2019). Fault detection and identification method using observer-based residuals. *Reliability Engineering & System Safety*, 184:27–40.

Ji, X., Yin, Z., Zhang, Y., Wang, M., Zhang, X., Zhang, C., and Wang, D. (2021). Real-time robust forecasting-aided state estimation of power system based on data-driven models. *International Journal of Electrical Power & Energy Systems*, 125:106412.

Ju, Y., Wei, G., Ding, D., and Liu, S. (2019). Event-triggered distributed fault detection over sensor networks in finite-frequency domain. *IET Control Theory & Applications*, 13(14):2261–2269.

Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. International Society for Optics and Photonics.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, Transactions of the ASME*, 82(1):35–45.

Kalman, R. E. and Bucy, R. S. (1961). New Results in Linear Filtering and Prediction Theory. *Journal of Basic Engineering*, 83(1):95–108.

Kaltiokallio, O., Hostettler, R., Yiğitler, H., and Valkama, M. (2021). Unsupervised Learning in RSS-Based DFLT Using an EM Algorithm. *Sensors*, 21(16).

Kamen, E. W. and Su, J. K. (1999). *Introduction to Optimal Estimation*. Advanced Textbooks in Control and Signal Processing. Springer London.

Kapila, V. and Grigoriadis, K. (2002). *Actuator saturation control*. CRC Press.

Kitagawa, G. (1994a). The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, 46(4):605–623.

Kitagawa, G. (1994b). The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, 46(4):605–623.

Kitagawa, G. and Gersch, W. (1996). *Smoothness priors analysis of time series*, volume 116. Springer Science & Business Media.

Krommer, A. R. and Ueberhuber, C. W. (1998). Computational Integration.

Ławryńczuk, M. (2015). Nonlinear predictive control for Hammerstein–Wiener systems. *ISA Transactions*, 55:49–62.

Lax, P. D. and Shea, M. (2014). *Calculus with Applications*. Springer.

Leong, A. S., Dey, S., and Nair, G. N. (2013). Quantized Filtering Schemes for Multi-Sensor Linear State Estimation: Stability and Performance Under High Rate Quantization. *IEEE Transactions on Signal Processing*, 61(15):3852–3865.

Leong, A. S., Dey, S., and Quevedo, D. E. (2018). Transmission scheduling for remote state estimation and control with an energy harvesting sensor. *Automatica*, 91:54–60.

Li, S., Sauter, D., and Xu, B. (2011). Fault isolation filter for networked control system with event-triggered sampling scheme. *Sensors*, 11(1):557–572.

Li, Y. and Lin, Z. (2018). *Stability and performance of control systems with actuator saturation*. Springer.

Liu, J., Liu, Y., Dong, K., Ding, Z., and He, Y. (2019). A Novel Distributed State Estimation Algorithm with Consensus Strategy. *Sensors*, 19(9).

Lo, J. (1972). Finite-dimensional sensor orbits and optimal nonlinear filtering. *IEEE Transactions on information theory*, 18(5):583–588.

Luo, X. and Song, Y. (2018). Data-driven predictive control of Hammerstein–Wiener systems based on subspace identification. *Information Sciences*, 422:447–461.

Mansouri, M., Tolouei, H., and Shoorehdeli, M. A. (2011). Identification of Hammerstein-Wiener ARMAX systems using Extended Kalman Filter. In *2011 Chinese Control and Decision Conference (CCDC)*, pages 1110–1114.

McLachlan, G. and Peel, D. (2004). *Finite Mixture Models*. Wiley Series in Probability and Statistics. Wiley.

McLaughlin, D. (2002). An integrated approach to hydrologic data assimilation: interpolation, smoothing, and filtering. *Advances in Water Resources*, 25(8):1275–1286.

Mellodge, P. (2015). *A practical approach to dynamical systems for engineers*. Woodhead Publishing.

Mengersen, K., Robert, C., and Titterington, M. (2011). *Mixtures: Estimation and Applications*. Wiley Series in Probability and Statistics. Wiley.

Msechu, E. J. and Giannakis, G. B. (2012). Sensor-Centric Data Reduction for Estimation With WSNs via Censoring and Quantization. *IEEE Transactions on Signal Processing*, 60(1):400–414.

Nadimi, E. S., Green, O., Blanes-Vidal, V., Larsen, J. J., and Christensen, L. P. (2012). Hammerstein-Wiener model for the prediction of temperature variations inside silage stack-bales using wireless sensor networks. *Biosystems Engineering*, 112(3):236–247.

Nemati, F., Safavi Hamami, S. M., and Zemouche, A. (2019). A nonlinear observer-based approach to fault detection, isolation and estimation for satellite formation flight application. *Automatica*, 107:474–482.

Noshad, Z., Javaid, N., Saba, T., Wadud, Z., Saleem, M. Q., Alzahrani, M. E., and Sheta, O. E. (2019). Fault Detection in Wireless Sensor Networks through the Random Forest Classifier. *Sensors*, 19(7).

Ogunfunmi, T. (2007). *Adaptive nonlinear system identification: The Volterra and Wiener model approaches*. Springer Science & Business Media.

Oppenheim, A. V. (1969). *Papers on Digital Signal Processing*. The MIT Press.

Orellana, R., Carvajal, R., Agüero, J. C., and Goodwin, G. C. (2020). Model error modelling using a stochastic embedding approach with gaussian mixture models for fir systems. *IFAC-PapersOnLine*, 53(2):845–850.

Orellana, R., Carvajal, R., Escárate, P., and Agüero, J. C. (2021). On the Uncertainty Identification for Linear Dynamic Systems Using Stochastic Embedding Approach with Gaussian Mixture Models. *Sensors*, 21(11):3837.

Papoulis, A. and Pillai, S. U. (1989). *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill.

Pês, B. D. S., Oroski, E., Guimarães, J. G., and Bonfim, M. J. C. (2019). A Hammerstein–Wiener Model for Single-Electron Transistors. *IEEE Transactions on Electron Devices*, 66(2):1092–1099.

Povey, D., Burget, L., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., Kumar, N., Karafiát, M., Rastrow, A., Rose, R. C., Schwarz, P., and Thomas, S. (2010). Subspace gaussian mixture models for speech recognition. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4330–4333.

Rahmathullah, A. S., Svensson, L., and Svensson, D. (2014). Two-filter gaussian mixture smoothing with posterior pruning. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE.

Rana, M. M. and Li, L. (2015). An Overview of Distributed Microgrid State Estimation and Control for Smart Grids. *Sensors*, 15(2).

Rostami Shahrbabaki, M., Safavi, A. A., Papageorgiou, M., and Papamichail, I. (2018). A data fusion approach for real-time traffic state estimation in urban signalized links. *Transportation Research Part C: Emerging Technologies*, 92:525–548.

Runnalls, A. R. (2007). Kullback-Leibler approach to Gaussian mixture reduction. *IEEE Transactions on Aerospace and Electronic Systems*, 43(3):989–999.

Salhi, H. and Kamoun, S. (2016). Combined Parameter and State Estimation Algorithms for Multivariable Nonlinear Systems Using MIMO Wiener Models. *Journal of Control Science and Engineering*, 2016:9614167.

Särkkä, S. (2013). *Bayesian filtering and smoothing*, volume 3. Cambridge University Press.

Schizas, I. D., Giannakis, G. B., Roumeliotis, S. I., and Ribeiro, A. (2008). Consensus in Ad Hoc WSNs With Noisy Links—Part II: Distributed Estimation and Smoothing of Random Signals. *IEEE Transactions on Signal Processing*, 56(4):1650–1666.

Schön, T. B., Wills, A., and Ninness, B. (2011). System identification of nonlinear state-space models. *Automatica*, 47(1):39–49.

Schoukens, M. and Tiels, K. (2017). Identification of block-oriented nonlinear systems starting from linear approximations: A survey. *Automatica*, 85:272–292.

Schreiter, T., van Lint, H., Treiber, M., and Hoogendoorn, S. (2010). Two fast implementations of the Adaptive Smoothing Method used in highway traffic state estimation. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1202–1208.

Simon, D. (2006). Optimal state estimation Kalman, H-infinity, and nonlinear approaches.

Singh, A. K. and Bhaumik, S. (2015). Higher degree cubature quadrature kalman filter. *International Journal of Control, Automation and Systems*, 13(5):1097–1105.

Slotine, J. E. and Li, W. (1991). *Applied Nonlinear Control*, volume 199. Prentice hall Englewood Cliffs, NJ.

Söderström, T. (2002). *Discrete-Time Stochastic Systems Estimation and Control*. Springer Science & Business Media.

Song, Y., Xie, C., and Chen, J. (2010). Medical image segmentation using characteristic function of Gaussian mixture models. *Proceedings - 2010 3rd International Conference on Biomedical Engineering and Informatics, BMEI 2010*, 1(Bmei):375–379.

Stoer, J. and Bulirsch, R. (2002). *Introducction to Numerical Analysis*. Texts in Applied Mathematics. Springer New York.

Stone, L. D., Streit, R. L., Corwin, T. L., and Bell, K. L. (2013). *Bayesian multiple target tracking*. Artech House.

Stroud, A. and Secrest, D. (1966). Gaussian quadrature formulas.

Titterington, D. M., Smith, A. F. M., and Makov, U. E. (1985). Statistical Analysis of Finite Mixture Distributions.

Wan, E. A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee.

Wang, L. Y., Yin, G. G., Zhang, J., and Zhao, Y. (2010). *System identification with quantized observations*. Springer.

Wang, X., Zhu, F., and Ding, F. (2020). The modified extended Kalman filter based recursive estimation for Wiener nonlinear systems with process noise and measurement noise. *International Journal of Adaptive Control and Signal Processing*, 34(10):1321–1340.

Wen, C., Wang, Z., Liu, Q., and Alsaadi, F. E. (2018). Recursive Distributed Filtering for a Class of State-Saturated Systems With Fading Measurements and Quantization Effects. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(6):930–941.

Widrow, B. and Kollár, I. (2008). *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge University Press.

Williams II, R. L. and Lawrence, D. A. (2007). *Linear State-Space Control Systems*. John Wiley & Sons.

Wills, A., Schön, T. B., Ljung, L., and Ninness, B. (2013). Identification of Hammerstein–Wiener models. *Automatica*, 49(1):70–81.

Xu, K., Yang, H., and Zhu, C. (2019). A novel extreme learning Machine-based Hammerstein-Wiener model for complex nonlinear industrial processes. *Neurocomputing*, 358:246–254.

Yang, H., Xia, Y., and Geng, Q. (2019). *Analysis and synthesis of delta operator systems with actuator saturation*, volume 193. Springer.

Yuz, J. I., Alfaro, J., Agüero, J. C., and Goodwin, G. C. (2011). Identification of continuous-time state-space models from non-uniform fast-sampled data. *IET Control Theory and Applications*, 5(7):842–855.

Zhang, L., Liang, H., Sun, Y., and Ahn, C. K. (2021). Adaptive Event-Triggered Fault Detection Scheme for Semi-Markovian Jump Systems With Output Quantization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(4):2370–2381.

Zhao, J., Netto, M., Huang, Z., Yu, S. S., Gómez-Expósito, A., Wang, S., Kamwa, I., Akhlaghi, S., Mili, L., Terzija, V., Meliopoulos, A. P. S., Pal, B., Singh, A. K., Abur, A., Bi, T., and Rouhani, A. (2021). Roles of Dynamic State Estimation in Power System Modeling, Monitoring and Operation. *IEEE Transactions on Power Systems*, 36(3):2462–2472.

Zhu, Y. (2002). Estimation of an N–L–N Hammerstein–Wiener model. *Automatica*, 38(9):1607–1614.