



UNIVERSIDAD DE CHILE  
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS  
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

**DETECCIÓN DE OBSOLESCENCIA DE MODELOS MEDIANTE LA  
APLICACIÓN DE TEST DE INFORMACIÓN MUTUA**

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

JOAQUÍN IGNACIO ORTEGA VÁSQUEZ

PROFESOR GUÍA:  
MARCOS ORCHARD CONCHA

PROFESOR CO-GUÍA:  
JORGE SILVA SÁNCHEZ

COMISIÓN:  
ANDRÉS CABA RUTTE

SANTIAGO DE CHILE  
2023

## DETECCIÓN DE OBSOLESCENCIA DE MODELOS MEDIANTE LA APLICACIÓN DE TEST DE INFORMACIÓN MUTUA

Es común en diversas áreas de la ingeniería ajustar modelos en base a datos para representar el comportamiento de un sistema. Específicamente para procesos complejos, el modelamiento en base a datos resulta particularmente útil para captar la dinámica del proceso sin necesariamente poseer las ecuaciones de los fenómenos detrás de este. Lo anterior es potencialmente útil para la detección y diagnóstico de fallas, dado que se puede utilizar un modelo correctamente ajustado como supervisor en tiempo real de un proceso, a esto se le llama *detección y diagnóstico de fallas basada en modelo*. Dado que los modelos son ajustados en condiciones de operación normal del sistema, una falla en este generará la **obsolescencia** del modelo.

Esta memoria tiene como objetivo, detectar la obsolescencia de modelos ajustados a un sistema real mediante la aplicación de una prueba de información mutua. Específicamente en un sistema de estanque cónico, con control de nivel PID. La hipótesis principal del problema propone que, ante condiciones normales de operación de la planta, no existe una dependencia estadística entre una señal residual (equivalente a la resta entre la salida real del sistema y la estimada por el modelo) y las entradas del sistema, mientras que la situación contraria ocurre cuando el sistema opera en falla. La dependencia estadística puede ser medida mediante la estimación de la **información mutua**.

El flujo de trabajo general de esta memoria, consistió en generar datos desde la simulación del sistema, tanto en condiciones normales como en falla, para luego ajustar modelos del tipo red neuronal MLP, que posteriormente eran testeados para diferentes escenarios de falla del sistema. Las predicciones resultantes del modelo eran sometidas al test de información mutua TSP, desarrollado en [1]. Finalmente, se obtiene como resultado la estimación de información mutua entre la señal residual y las entradas del sistema.

Los resultados obtenidos demostraron una capacidad de detección limitada por el nivel de ruido presente en las señales censadas, siendo exitosa (detección de obsolescencia en más del 85 % de los casos simulados) en niveles de ruido medio-bajo y bajo; mientras que fue parcialmente exitosa (sólo hubo detección en un tipo de falla) para niveles de ruido medio-alto y alto.

Se concluye que el test de información mutua resulta adecuado para la detección de obsolescencia de modelos ajustados a este tipo de sistemas, coincidiendo con lo esperado teóricamente y confirmando la hipótesis del problema. Resulta particularmente útil para la detección y diagnóstico de fallas si existe la capacidad de generar modelos correctamente ajustados al fenómeno físico que rige al sistema en cuestión. Sus principales ventajas son la baja tasa de falsos positivos, la alta tasa de detección y la posibilidad de aislamiento de una falla, lo que lo hace un test superior a un análisis simple de residuo.

*(...) ¡Jamás! ¡Jamás, Marge!  
Yo no puedo vivir una vida vacía como tú.  
¡Lo quiero todo!  
Las perturbadoras altas,  
las terribles bajas,  
la gris mediocridad.  
Claro; puedo ofender a algunos recatados  
con mi arrogante paso,  
y mi olor a almizcle...  
¡Oh! ¡Jamás seré el borreguito  
de los llamados “ciudadanos modelo”  
que enrollan su lengua,  
enchinan sus barbas  
y deliberan sobre qué deben hacer  
con ese “Homero Simpson”!*

***Homero Simpson***

# Agradecimientos

En primer lugar, quiero agradecer a mis padres Marisol y Robert, por todo el sacrificio que han hecho para mantener a la familia, realmente no dimensiono el esfuerzo que han hecho desde siempre para poder darnos lo mejor a mí con mi hermana. A mi tía Jeanette, que siempre ha estado apoyándome e incentivándome a ser una persona integral, motivándome en el deporte y en los estudios desde siempre. A mi hermana por todos los momentos de risa, reflexión, y conversaciones científicas que ha compartido conmigo. También a toda mi familia en general, mi abuelita María, mis tías Jeanette, Maritza, Ely, mi tío Miguel, y a todos mis primos. A mi tata Juan, fallecido el 14 de septiembre recién pasado a la edad de 99 años, por todo el amor y enseñanzas que me dio. Un hombre realmente ejemplar, capaz de salir de la pobreza y mantener a una familia junto a mi abuelita Elisa. Desde pequeño lo recuerdo diciéndome que “*el estudio es la única forma de salir adelante*”. De alguna u otra forma, lamento no haber podido hacerle esta dedicatoria en vida.

Gracias infinitas a mi polola María José, por la gran compañía, la enorme paciencia que me ha tenido y por todos los buenos momentos que me ha dado, ha sido una verdadera compañera de vida. Gracias a ella, esta etapa universitaria ha sido sin duda la mejor de mi vida. Agradecer también a toda su familia por recibirme en el sur cada vez que voy.

A mis amigos de la U, del grupo 3R de inducción que luego creció hasta ser el “dieta milagrosa”, siempre recordaré las tardes de almuerzo jugando cartas. Especialmente quiero agradecer al Eitan y la Consu, quienes pude conocer mejor y tener momentos más personales. De igual forma agradecer a mi amiga Sofía Medina, con quien si bien no hablamos mucho, me ha ayudado mucho en momentos difíciles.

A mis amigos del colegio; Cata, Esteban, Fran, Ivo, Josefa y Seba; que si bien no nos hemos podido juntar mucho físicamente, fueron muy gratos los momentos jugando al teléfono durante la pandemia.

Quiero agradecer también a mi profesor guía, Marcos Orchard, por la infinita paciencia y todo el tiempo que se tomó en estas reuniones casi semanales desde el 15 de marzo del 2022, explicándome las veces que fuera necesario para que pudiera comprender con claridad los temas que estábamos tratando. Un excelente profesor y aún mejor persona.

Agradecer también al profesor Jorge Silva, que si bien conocí el segundo semestre, ha tenido una gran disposición a explicar todo lo que le pregunté relacionado al tema. Igualmente al equipo de las reuniones semanales, Camilo y Tomás, quienes me ayudaron a encontrar el norte del problema cuando en su momento fue difuso.

# Tabla de Contenido

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación y Formulación del Problema . . . . .	1
1.2. Hipótesis y Objetivos . . . . .	3
1.2.1. Hipótesis . . . . .	3
1.2.2. Objetivo general . . . . .	3
1.2.3. Objetivos específicos . . . . .	3
1.3. Estructura de la Memoria . . . . .	4
<b>2. Marco Teórico y Estado del Arte</b>	<b>5</b>
2.1. Modelo Fenomenológico de Estanque Cónico . . . . .	5
2.1.1. Simulación de Fallas . . . . .	6
2.2. Control PID . . . . .	7
2.3. Modelos basados en Redes Neuronales artificiales (NNs) . . . . .	8
2.3.1. Perceptrón Multicapa . . . . .	9
2.3.2. El Problema de Regresión . . . . .	10
2.3.3. Error Cuadrático Medio (MSE) . . . . .	11
2.3.4. Predicción <i>Naive</i> . . . . .	11
2.4. Test de Hipótesis . . . . .	12
2.5. Información Mutua . . . . .	12
2.6. Detección y Diagnóstico de Fallas Basado en Modelo . . . . .	13
2.7. Estimación de la Información Mutua . . . . .	13
2.8. Obsolescencia de Modelos ( <i>mismodeling</i> ) Basado en la Información Mutua . . . . .	14
2.8.1. Experimentos Previos (M. Videla, 2022) . . . . .	14
2.8.2. Análisis teórico del problema (C. Ramírez, 2022) . . . . .	16
<b>3. Metodología</b>	<b>18</b>
3.1. Flujo de Trabajo . . . . .	18
3.1.1. Generación de Datos . . . . .	18
3.1.2. Ajuste de Modelos . . . . .	19
3.1.3. Operación del Modelo, Generación del Residuo y Aplicación del Test de Información Mutua . . . . .	20
3.1.4. Análisis de Validez . . . . .	22
3.1.5. Observación Relevante: Obsolescencia $\neq$ Falla . . . . .	23
3.1.6. Análisis de resultados y conclusiones . . . . .	23
3.2. Datos y Modelos Generados . . . . .	24
3.2.1. Nivel de Ruido . . . . .	24
3.2.2. Tiempos de Muestreo . . . . .	24

3.2.3.	Parámetros PID . . . . .	24
3.2.4.	Fallas . . . . .	25
<b>4.</b>	<b>Resultados Experimentales y análisis</b>	<b>26</b>
4.1.	Muestreo cada 25 segundos . . . . .	26
4.1.1.	Filtrado . . . . .	27
4.1.2.	Parámetro $\lambda$ . . . . .	27
4.1.3.	Tamaño de la ventana deslizante . . . . .	27
4.1.4.	Resultados . . . . .	27
4.1.4.1.	Ventana de 720 muestras (5 horas), $\lambda = 7.5 \cdot 10^{-5}$ . . . . .	28
4.1.4.2.	Ventana de 720 muestras, $\lambda = 7.5 \cdot 10^{-6}$ . . . . .	30
4.1.4.3.	Ventana de 720 muestras, $\lambda = 7.5 \cdot 10^{-4}$ . . . . .	31
4.1.4.4.	Ventana de 1440 muestras (10 horas). . . . .	32
4.1.4.5.	Ventana de 2160 muestras (15 horas). . . . .	33
4.1.5.	Análisis y comentarios para los próximos experimentos . . . . .	34
4.2.	Muestreo cada 10 segundos . . . . .	35
4.3.	Muestreo cada 5 segundos . . . . .	39
<b>5.</b>	<b>Discusión de Resultados</b>	<b>43</b>
5.1.	Funcionamiento del Test . . . . .	43
5.1.1.	Detección . . . . .	43
5.1.2.	Sensibilidad a los tipos de falla simulados . . . . .	43
5.1.3.	Aislamiento de falla detectada . . . . .	45
5.2.	Influencia del Ruido . . . . .	45
5.3.	Influencia del Tiempo de Muestreo . . . . .	46
5.4.	Influencia del Lazo de Control del sistema . . . . .	46
<b>6.</b>	<b>Conclusiones y Trabajo Futuro</b>	<b>48</b>
6.1.	Conclusiones Principales . . . . .	48
6.2.	Trabajo Futuro . . . . .	49
	<b>Bibliografía</b>	<b>50</b>
	<b>Anexos</b>	<b>53</b>
A.	Resultados para modelos de desempeño regular . . . . .	53
B.	Resultados con distintas semillas de generación de ruido . . . . .	55
C.	Resultados de aplicar el test con mayor cantidad de autorregresores de $f$ y $h$ . . . . .	56
D.	Herramientas Computacionales . . . . .	59
D.1.	Generación de Datos: Matlab y Simulink . . . . .	59
D.2.	Generación de Referencias: Microsoft Excel . . . . .	60
D.3.	Ordenamiento y Limpieza de Datos: Pandas y NumPy . . . . .	61
D.4.	Generación y Análisis de Modelos: PyTorch y SciKitLearn . . . . .	61
D.5.	Almacenamiento de Datos: Google Drive . . . . .	64
D.6.	Realización del Test y Automatización del proceso: Python en Google Colaboratory . . . . .	65

# Índice de Tablas

2.1.	Valores numéricos de las constantes del estanque cónico . . . . .	6
4.1.	Valores del MSE acorde al nivel de ruido para los modelos con tiempo de muestreo de 25 segundos. . . . .	26
4.2.	Valores del MSE acorde al nivel de ruido para los modelos con tiempo de muestreo de 10 segundos. . . . .	35
4.3.	Valores del MSE acorde al nivel de ruido para los modelos con tiempo de muestreo de 5 segundos. . . . .	39

# Índice de Ilustraciones

1.1.	<i>Pipeline</i> general de la detección de fallas basado en modelos . . . . .	2
2.1.	Diagrama básico de las variables y componentes del sistema en cuestión. . . . .	5
2.2.	Esquema básico de un sistema de control PID. . . . .	7
2.3.	Esquema de una neurona y del proceso de la sinapsis. . . . .	8
2.4.	Esquema de una neurona artificial. . . . .	9
2.5.	Esquema de la arquitectura de un perceptrón multicapa. . . . .	10
2.6.	Ejemplos de subajuste y sobreajuste para el problema de regresión. . . . .	11
2.7.	Resultados de uno de los experimentos desarrollados por M. Videla en 2022 [24].	15
2.8.	Resultados del análisis teórico . . . . .	17
3.1.	Resumen del sistema completo con control de nivel a lazo cerrado . . . . .	18
3.2.	Resumen del <i>loop</i> de entrenamiento . . . . .	20
3.3.	Análisis de ventana deslizante . . . . .	21
3.4.	Simulación con datos sintéticos . . . . .	22
4.1.	Resultados no filtrados . . . . .	28
4.2.	Resultados filtrados . . . . .	29
4.3.	Resultados para el segundo valor de $\lambda$ . . . . .	30
4.4.	Resultados para el tercer valor de $\lambda$ . . . . .	31
4.5.	Resultados para una ventana de 1440 muestras . . . . .	32
4.6.	Resultados para una ventana de 2160 muestras . . . . .	33
4.7.	Resultados para un tiempo de muestreo de 10 (s), con una ventana de 720 muestras	36
4.8.	Resultados para un tiempo de muestreo de 10 (s), con una ventana de 1440 muestras . . . . .	37
4.9.	Resultados para un tiempo de muestreo de 10 (s), con una ventana de 2160 muestras . . . . .	38
4.10.	Resultados para un tiempo de muestreo de 5 (s), con una ventana de 720 muestras	40
4.11.	Resultados para un tiempo de muestreo de 5 (s), con una ventana de 1440 muestras	41
4.12.	Resultados para un tiempo de muestreo de 5 (s), con una ventana de 1440 muestras	42
5.1.	Comparación de afección de las constantes de perturbación equivalentes en los flujos de entrada y salida. . . . .	44
5.2.	Comparación de afección de las constantes de perturbación, relativas a la cons- tante que afecta cada una, en los flujos de entrada y salida. . . . .	45
A.1.	Modelo ajustado por 300 épocas . . . . .	53
A.2.	Modelo ajustado por 1000 épocas . . . . .	54
B.1.	Resultados para falla grave en la bomba, con distintas semillas de generación de ruido. . . . .	55
C.1.	EMI del primer autorregresor de $f$ e $y$ . . . . .	56
C.2.	EMI del segundo autorregresor de $f$ e $y$ . . . . .	56
C.3.	EMI del tercer autorregresor de $f$ e $y$ . . . . .	56

C.4.	EMI del cuarto autorregresor de $f$ e $y$ . . . . .	57
C.5.	EMI del quinto autorregresor de $f$ e $y$ . . . . .	57
C.6.	EMI del sexto autorregresor de $f$ e $y$ . . . . .	57
C.7.	EMI del séptimo autorregresor de $f$ e $y$ . . . . .	57
C.8.	EMI del octavo autorregresor de $f$ e $y$ . . . . .	58
C.9.	EMI del noveno autorregresor de $f$ e $y$ . . . . .	58
C.10.	EMI del décimo autorregresor de $f$ e $y$ . . . . .	58
D.1.	Diagrama de simulink utilizado, notar que abajo a la derecha está el simulador del estanque desarrollado en [5], regido por las ecuaciones de este. . . . .	59
D.2.	Sección modificada correspondiente a la simulación de las ecuaciones del sistema. Notar que las constantes llamadas en el diagrama como “coef_falla_1” y “coef_falla_2” corresponden a los coeficientes de perturbación $\delta_2$ y $\delta_1$ respectivamente. . . . .	60
D.3.	Plantilla generadora de referencias, notar que se logra un texto concatenado que permite ingresar un gran número de cambios de referencia a MATLAB. . . . .	61
D.4.	Almacenamiento de los datos en Google Drive . . . . .	64

# Capítulo 1

## Introducción

### 1.1. Motivación y Formulación del Problema

Es común en diversas áreas de la ingeniería ajustar modelos en base a datos para representar el comportamiento de un sistema (modelamiento). Específicamente para procesos complejos, el modelamiento en base a datos resulta particularmente útil para captar la dinámica del proceso sin necesariamente poseer las ecuaciones de los fenómenos detrás de este.

Los motivos por los cuales es conveniente disponer de modelos que representen a un sistema son variados, siendo uno de estos motivos la supervisión del correcto funcionamiento del sistema mediante la comparación de la respuesta de salida real con la respuesta estimada por el modelo. De esta forma (asumiendo un modelo vigente y funcional, validado por datos históricos del sistema operando en condiciones normales), discrepancias en la señal de salida del modelo con respecto a la del sistema dan indicios de un cambio en el comportamiento de este, lo que puede ser indicio de una falla o de necesidad de mantención. Este principio, de hecho, es el funcionamiento base de todo un área de estudio en el campo del control de sistemas: la detección y diagnóstico de fallas basada en modelos (*Model-based fault-detection and diagnosis*) [2].

En relación con lo anterior, se define la **obsolescencia** de un modelo (*mismodeling*) como el punto en el cual este ya no representa al sistema que está modelando, lo cual es una causal directa de la revisión del sistema (o del modelo) y por lo tanto es importante la capacidad de detectar esta obsolescencia. Lo anterior no resulta trivial cuando las diferencias entre la salida real y la estimada por el modelo son leves y confundibles con el ruido. La idea principal de este método se muestra en la Figura 1.1

Este problema de detección de obsolescencia puede verse como un test de hipótesis, en el cual la hipótesis nula afirma la correcta representación del sistema por parte del modelo, mientras que la hipótesis alternativa afirma lo contrario (i.e. que el modelo ha quedado obsoleto). A continuación se presenta un esquema formal de este test planteado:

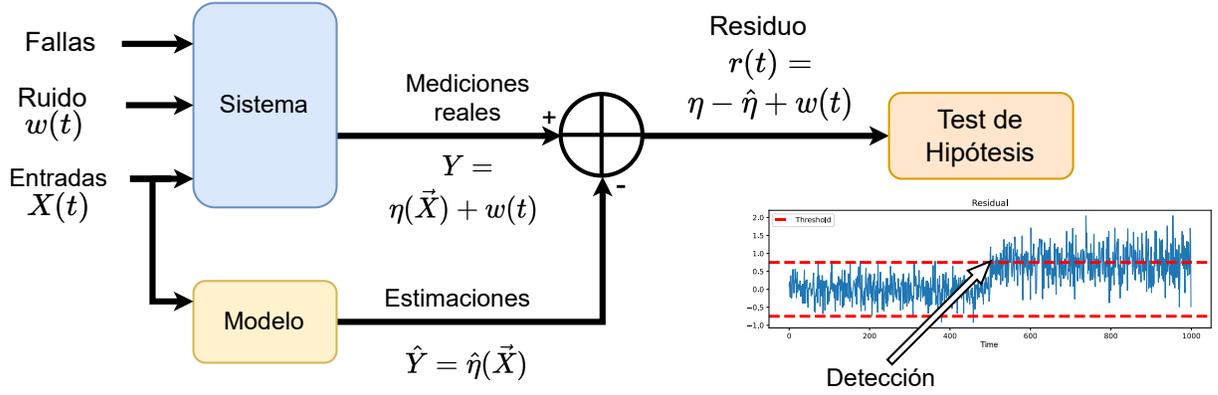


Figura 1.1: *Pipeline* general de la detección de fallas basado en modelos. Modificado de [3, 4]

Sea  $\mathbf{Y}(\vec{x}) = \eta(\vec{x}) + w(t)$  la señal de salida observada del sistema para las entradas dadas en  $\vec{x}$ , con  $\eta(\vec{x})$  el fenómeno real que rige al sistema y  $w(t)$  un ruido aditivo arbitrariamente distribuido. Sea además  $\hat{\mathbf{Y}} = \hat{\eta}(\vec{x})$  la señal de salida del modelo para las mismas entradas (de modo que  $\hat{\mathbf{Y}}$  es un estimador de  $\mathbf{Y}$ ). Se definen las hipótesis nula y alternativa como lo siguiente:

$$H_0 : \mathbf{Y} \sim \hat{\mathbf{Y}} \quad (\text{i.e. el modelo está vigente})$$

$$H_1 : \mathbf{Y} \not\sim \hat{\mathbf{Y}} \quad (\text{i.e. el modelo está obsoleto})$$

Equivalentemente, se puede desarrollar el test anteriormente expuesto mediante la definición del **residuo**, igual a la resta simple entre las señales  $\mathbf{Y}$  e  $\hat{\mathbf{Y}}$ , tal como se muestra en la Figura 1.1. De esta forma, el test se re-formula de la siguiente manera:

$$H_0 : \mathbf{r} \sim w(t)$$

$$H_1 : \mathbf{r} \approx g(\vec{x}) + w(t)$$

Con  $g(\vec{x}) = \mathbf{Y} - \hat{\mathbf{Y}} \neq 0$  una función resultante de la obsolescencia del modelo, que genera una dependencia del residuo con las entradas del sistema.

Finalmente, el problema puede verse como un test de independencia del residuo con respecto a las entradas del sistema, lo que permite el desarrollo de un sistema de detección de obsolescencia de modelos basado en la estimación de la dependencia de estas señales. Dado que en sistemas reales y medianamente complejos, calcular dependencias en base a relaciones matemáticas formales y exactas resulta imposible, es que surge la idea de estimar la dependencia estadística entre las señales.

Del problema recién descrito surge la motivación de este trabajo, esto es, detectar la obsolescencia de modelos a través del uso de una prueba de información mutua. Para ello, se necesita un sistema del cual extraer datos para ajustar los modelos que lo representarán, definir el llamado “oráculo” (i.e. datos de la salida real o datos de prueba) para tener la ca-

pacidad de testeo del modelo con respecto al sistema y, por último, se precisa de un algoritmo que estime la dependencia estadística entre las entradas y el residuo.

Con respecto a lo anterior, se eligió como sistema el estanque cónico, particularmente el modelo fenomenológico desarrollado por C. Jáuregui en 2016 [5], basado en el modelamiento fenomenológico de la planta real del estanque cónico con recirculación disponible en el laboratorio de automática del departamento de ingeniería eléctrica de la Universidad de Chile (DIE). Las ventajas de usar este modelo fenomenológico como oráculo para la experimentación de este trabajo son principalmente la no linealidad de la planta, la existencia física de esta (que abre un potencial trabajo futuro de experimentar con datos reales) y que el ajuste de modelos del estilo red neuronal es factible a partir de la generación de datos representativos del sistema.

Para medir la dependencia estadística, se utilizará la estimación de la **información mutua** (MI por sus siglas en inglés) desarrollada en [1] [4], disponible para usar en la librería TSP-IT desarrollada por M. Videla en 2022 para el lenguaje de programación *Python*. A este test de independencia se le llamará **test de información mutua**.

## 1.2. Hipótesis y Objetivos

### 1.2.1. Hipótesis

La hipótesis principal del problema propone que, ante condiciones normales de operación de la planta, y bajo el supuesto de un modelo representativo del sistema dentro del rango de operación definido, las entradas del sistema no deben tener dependencia estadística con la señal residual, mientras que ante la obsolescencia del modelo en cuestión, esta situación cambia, por lo que el residuo tendrá dependencia estadística con al menos una componente de la entrada.

### 1.2.2. Objetivo general

Detectar la obsolescencia de modelos ajustados a un sistema real mediante la aplicación de una prueba de información mutua.

### 1.2.3. Objetivos específicos

- Elegir una planta de la cual extraer datos para generar un oráculo, modelar, supervisar y testear.
- Ajustar y validar un modelo del sistema de la planta anteriormente elegida.
- Simular una falla incipiente o cambio en las condiciones de la planta que gatille la obsolescencia del modelo ajustado.
- Experimentar mediante la prueba de información mutua el comportamiento de la señal de error con respecto a las entradas del sistema.
- Analizar y validar resultados.

## 1.3. Estructura de la Memoria

Esta memoria consta de 6 capítulos más una sección de anexos, los cuales son descritos brevemente a continuación:

- **Introducción:** Se realiza una introducción al problema abordado en este trabajo, las hipótesis, objetivos y el tema de estudio relacionado con éste, además de los fundamentos teóricos básicos que sustentan la hipótesis del problema.
- **Marco Teórico y Estado del Arte:** Se describen brevemente los conceptos teóricos relevantes utilizados en el trabajo, en conjunto al trabajo previo existente en la misma línea del problema.
- **Metodología:** Se expone el flujo de trabajo general utilizado para abordar el problema, y se describen los datos y modelos generados para la experimentación.
- **Resultados Experimentales y Análisis:** Se muestran los diversos resultados obtenidos de manera gráfica, en conjunto a un un breve análisis de estos.
- **Discusión de Resultados:** Se comenta sobre los resultados obtenidos, relacionando los comportamientos de las variables y su relación con la teoría.
- **Conclusiones y Trabajo Futuro:** Se muestran las conclusiones del trabajo realizado y algunas propuestas de trabajo futuro que se consideran relevantes para el crecimiento de la investigación en torno al problema.
- **Anexos:** Se muestran más resultados de experimentos y una descripción de las herramientas computacionales utilizadas, incluidos algunos fragmentos de código destacados.

# Capítulo 2

## Marco Teórico y Estado del Arte

### 2.1. Modelo Fenomenológico de Estanque Cónico

Tal como se adelantó en el capítulo de introducción, el sistema de estanque cónico con el cual se trabaja corresponde al modelado por C. Jáuregui en base al sistema físico presente en el laboratorio de automática del DIE. Por lo que los fundamentos físicos utilizados para el modelamiento fenomenológico son extraídos desde [5].

La base del sistema en cuestión puede resumirse con el esquemático de la Figura 2.1

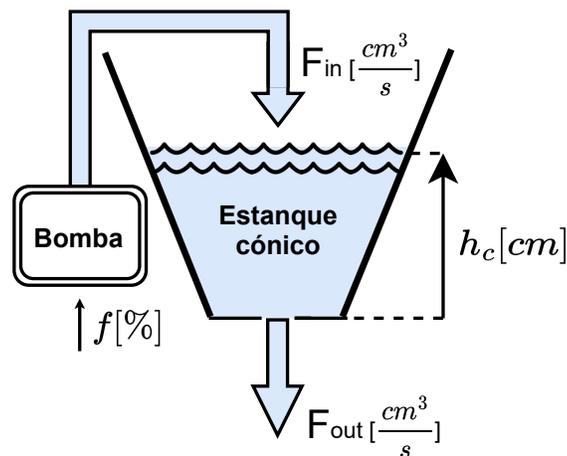


Figura 2.1: Diagrama básico de las variables y componentes del sistema en cuestión. Modificado de [5]

Tal como puede verse en el diagrama, existen múltiples variables del sistema, las cuales se definen a continuación.

- $h_c \geq 0$ : Altura del agua en el estanque cónico.
- $F_{in}$ : Flujo de entrada de agua al estanque cónico.
- $F_{out}$ : Flujo de salida de agua del estanque cónico.
- $f$ : Porcentaje de utilización de la bomba.

- $V$  (no presente en la imagen): Corresponde al volumen de agua dentro del estanque, por lo que está relacionado con  $h_c$

Asimismo, las ecuaciones que relacionan estas variables están bien definidas una vez se conocen ciertos parámetros dependientes de las características físicas del estanque y del motor de la bomba. Los principios de funcionamiento de estas ecuaciones se detallan en [5], y se resumen como lo siguiente:

$$F_{in} = \alpha_1 \cdot f + \alpha_2 \geq 0 \quad (2.1)$$

$$F_{out} = \beta \sqrt{h_c} \quad (2.2)$$

$$V = 0,21h_c^3 + 5,7h_c^2 + 17,1h_c + 290,7 \quad (2.3)$$

$$\frac{\partial V}{\partial t} = \frac{\partial V}{\partial h} \cdot \frac{\partial h}{\partial t} = F_{in} - F_{out} \quad (2.4)$$

Con lo que finalmente se obtiene:

$$\dot{h}_c = \frac{F_{in} - F_{out}}{0,63h_c^2 + 11,4h_c + 17,1} \quad (2.5)$$

Con  $\alpha_1$ ,  $\alpha_2$  parámetros dependientes de las características del motor de la bomba; y  $\beta$  un parámetro dependiente de la geometría del estanque, las propiedades de los fluidos que forman parte del sistema y la aceleración de gravedad. Los valores que acompañan a las distintas potencias de  $h_c$  en la ecuación 2.3 corresponden a parámetros obtenidos de manera experimental y que también dependen de la geometría del estanque utilizado. Las ecuaciones difieren sustancialmente de un estanque cónico teórico debido a que existen inclinaciones en el estanque real que fueron consideradas en este modelamiento [5]. Los valores numéricos usados se muestran en la Tabla 2.1.

Tabla 2.1: Valores numéricos de las constantes del estanque cónico

Parámetro	Valor
$\alpha_1$	5.43 $\left[\frac{cm^3}{s\%}\right]$
$\alpha_2$	-78.23 $\left[\frac{cm^3}{s}\right]$
$\beta$	20.21 $\left[\frac{cm^{5/2}}{s}\right]$

De esta forma, con estas ecuaciones y los valores de los parámetros es posible modelar un estanque cónico con recirculación. El software utilizado para modelar este sistema es MATLAB y Simulink. En este trabajo se modificará el mismo modelo “Sistema\_Estanques.slx” presentado en [5], con el consentimiento del autor.

### 2.1.1. Simulación de Fallas

Para experimentar y generar datos del sistema operando en falla, se generarán dos tipos de perturbaciones en las ecuaciones mismas del estanque y la bomba, que cambiarán el comportamiento general y gatillarán la obsolescencia del modelo ajustado para el sistema anterior. Las perturbaciones se añadirán en las ecuaciones 2.1, 2.2 de la siguiente manera:

$$F_{in} = (\alpha_1 + \delta_1) \cdot f + \alpha_2 \geq 0 \quad (2.6)$$

$$F_{out} = (\beta + \delta_2)\sqrt{h_c} \quad (2.7)$$

Con  $\delta_1$  y  $\delta_2 \in \mathbb{R}$  los coeficientes de perturbación. Así, la ecuación 2.6 corresponderá a una falla en la bomba para  $\delta_1 \neq 0$  (i.e. se requerirá de más o menos valor de la variable  $f$  para obtener el mismo valor de  $F_{in}$  en condiciones normales); mientras que la ecuación 2.7 corresponderá a una falla en la salida del agua para  $\delta_2 \neq 0$  (e.g. fugas, tapones, cambios en el flujo de salida esperado)

## 2.2. Control PID

Como siguiente concepto fundamental, se tiene el control PID, el cual corresponde a una de las técnicas de control más utilizadas en la industria. El nombre corresponde a la sigla “Proporcional, Integral y Derivativo” [6]. En el esquema de la Figura 2.3 se muestra el principio de funcionamiento del control PID, esto es, la implementación en lazo cerrado, en la cual el controlador tiene como señal de entrada una retroalimentación generada por la diferencia entre la referencia y la salida del sistema, lo que se conoce como “error”.

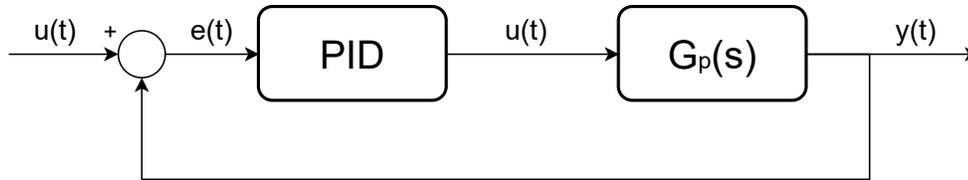


Figura 2.2: Esquema básico de un sistema de control PID, el bloque PID corresponde al bloque de control, mientras que el bloque  $G_p(s)$  corresponde a la planta haciendo referencia a la función de transferencia de esta. Las señales  $r$ ,  $e$ ,  $u$  e  $y$  en función del tiempo corresponden a la entrada o referencia, el error, la salida del control PID y la señal de salida respectivamente. Extraído de [7]

Con respecto a cada componente de los controladores PID (esto es P, I y D por separado), estas son acciones que se toman para operar con la señal de error, y están indicadas como  $K_p$ ,  $K_i$  y  $K_d$  en la ecuación 2.8:

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (2.8)$$

Con  $u(t)$  la señal de salida del control PID,  $e(t)$  el error y  $t$  el tiempo.

Tal como puede inferirse de la ecuación, cada componente P, I y D tendrá un efecto particular en la señal de salida del controlador. Conocer cualitativamente estas características es de mucha utilidad, puesto que permite estimar el comportamiento de una señal de salida y descartar la configuración de parámetros sin la necesidad imperiosa de simular esta. Estas características se resumen en lo siguiente:

- $K_p$ : Tal como su nombre y posición en la ecuación lo indica, esta acción es proporcional

al error actual (i.e. no tiene “memoria”). Sus efectos en el control son que, ante un mayor valor de  $K_p$  se tiene una respuesta más rápida (menor tiempo de subida) y un mejor error de estado estacionario (error en el régimen permanente), pero con un mayor sobrepaso (esto es, una mayor sobre-oscilación provocada por el “paso” de la señal de salida por encima de la referencia) [5, 7].

- $K_i$ : Esta acción considera a la integral del error entre  $t = 0$  hasta el tiempo actual, por lo que sí considera una “memoria” de los errores pasados. Elimina el error de estado estacionario, pero al incrementar este valor se incrementa el sobrepaso. Esta acción es por naturaleza desestabilizadora, por lo que siempre va acompañada de la acción proporcional [5, 7].
- $K_d$ : Esta acción considera a la derivada del error con respecto al tiempo en el instante actual, por lo que representa la tendencia al cambio y por lo tanto considera una proyección hacia el futuro. Reduce el sobrepaso pero tiene poca robustez al ruido. [5, 7].

Finalmente, en Simulink existe el bloque “PID”, el cual permite setear el trío de componentes  $K_p$ ,  $K_i$  y  $K_d$ , realizando la tarea que compete a este tipo de controlador.

### 2.3. Modelos basados en Redes Neuronales artificiales (NNs)

Esta sección está basada en [8] y se limita a definir y explicar a modo general el funcionamiento de una red neuronal. Esto pues el concepto de red neuronal es utilizado en diversos campos sumamente amplios que no compaginan con el uso que se le dará en este trabajo.

Una red neuronal, como su nombre lo indica, es una red formada por neuronas. Particularmente las redes neuronales artificiales toman el nombre desde el concepto biológico de neurona, la cual se define como la unidad estructural y funcional del sistema nervioso, especializada en la comunicación rápida con otras neuronas mediante la sinapsis [9]. En la Figura 2.3 se muestra un esquema básico de las componentes de una neurona y del proceso de sinapsis.

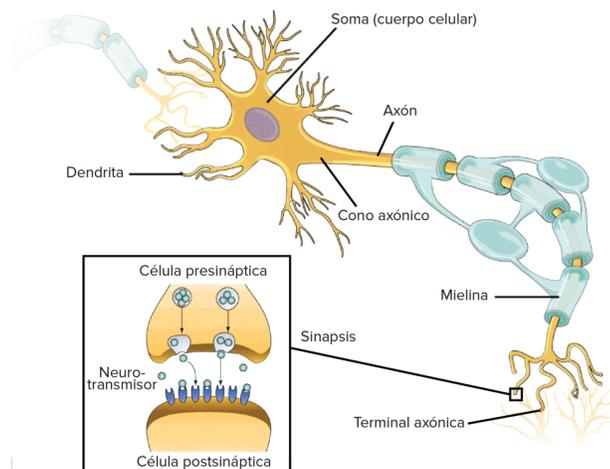


Figura 2.3: Esquema de una neurona y del proceso de la sinapsis. Extraído de [10]

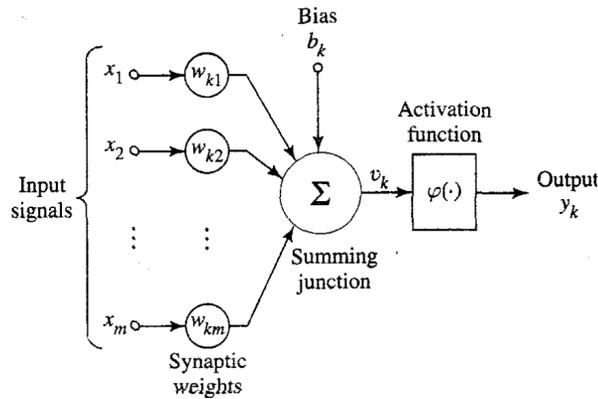


Figura 2.4: Esquema de una neurona artificial. Extraído de [8]

De esta forma, el concepto de neurona artificial genera un modelo matemático regido por el principio que una neurona tiene múltiples entradas y una salida, en donde las entradas corresponden a las señales ponderadas por un valor llamado peso sináptico, estas pueden ser la salida de otra neurona, lo que permite el avance de la red hacia adelante. La operación realizada en esta unidad corresponde a la combinación lineal de las entradas por los pesos sinápticos, más la adición de un valor constante llamado sesgo. Lo que distingue a la neurona de un operador lineal simple es la función de activación, la cual consiste en una pseudo regla de decisión que, ante la combinación lineal de las distintas entradas con el sesgo, decide si enviar o no la señal (dependiendo de qué función de activación se elija, pueden haber valores difusos en el borderline), similar al principio biológico del todo o nada [8]. En la Figura 2.4 se muestra un esquema de la arquitectura de la neurona artificial. En las ecuaciones 2.9 y 2.10 se muestra el mismo esquema en términos matemáticos.

$$v_k = \sum_{j=1}^m w_{kj} x_j \quad (2.9)$$

$$y_k = \phi(v_k + b_k) \quad (2.10)$$

Con  $v_k$  la salida de la combinación lineal de las entradas con el sesgo,  $x_j$  la  $j$ -ésima señal de entrada,  $w_{kj}$  el peso sináptico asociado a la  $j$ -ésima entrada,  $b_k$  el sesgo y  $y_k$  la señal de salida.

Finalmente, volviendo al inicio, se puede definir de manera básica la **Red Neuronal** como un conjunto de capas formadas por distinto número de neuronas cada una. La principal característica de estas redes es que, mediante un conjunto de ejemplos, una función de pérdida a minimizar (*loss function*) y el algoritmo de propagación hacia atrás (*backpropagation*), se pueden reajustar los valores de los pesos y el sesgo, con lo que es posible ajustar una red a distintos escenarios, y ha demostrado una gran versatilidad en el tipo de problema a resolver [8].

### 2.3.1. Perceptrón Multicapa

Una de las arquitecturas de redes las más utilizadas (y precisamente la utilizada en este trabajo) corresponde al **perceptrón multicapa** (MLP), la cual corresponde a una red neu-

ronal formada por capas en la cual la salida de cada neurona es entrada de cada neurona de la capa siguiente. El MLP consta de al menos tres capas: La capa de entrada, las capas ocultas y la capa de salida, las cuales pueden verse en el esquema de la arquitectura del MLP de la Figura 2.5.

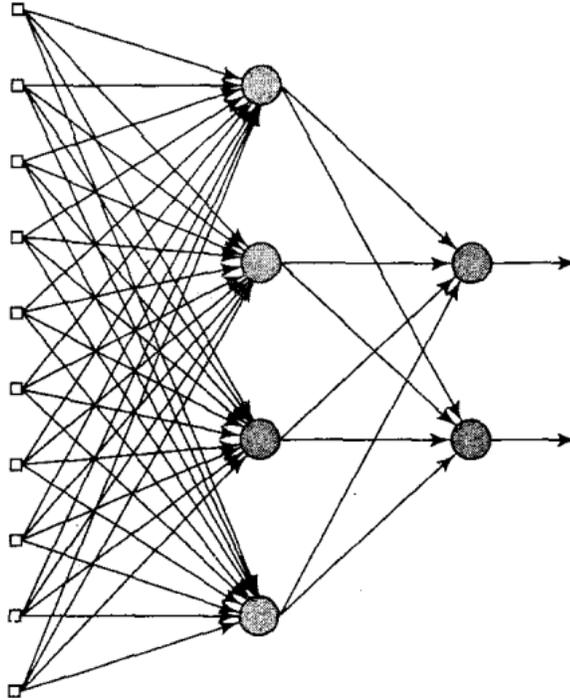


Figura 2.5: Esquema de la arquitectura de un perceptrón multicapa. Extraído de [8]

La popularidad del MLP se debe al buen desempeño que tiene en los problemas tanto de clasificación como de regresión (sin embargo, no ha demostrado buen desempeño en el problema de extrapolación).

### 2.3.2. El Problema de Regresión

El problema de regresión consiste en cómo modelar una o más variables dependientes en base a un set de variables de predicción. El objetivo es indicar un valor continuo de la variable dependiente dadas las variables de predicción [11–13].

En la línea de las redes neuronales y modelos en base a datos, con ejemplos representativos y adecuado entrenamiento, una red de este tipo puede ajustarse correctamente a un proceso (al menos en el rango de operación de los datos con los que fue entrenada). Si el entrenamiento es insuficiente o excesivo, ocurrirá el “subajuste” (*underfitting*) y “sobreajuste” (*overfitting*), tal como se muestra en la Figura 2.6 [14].

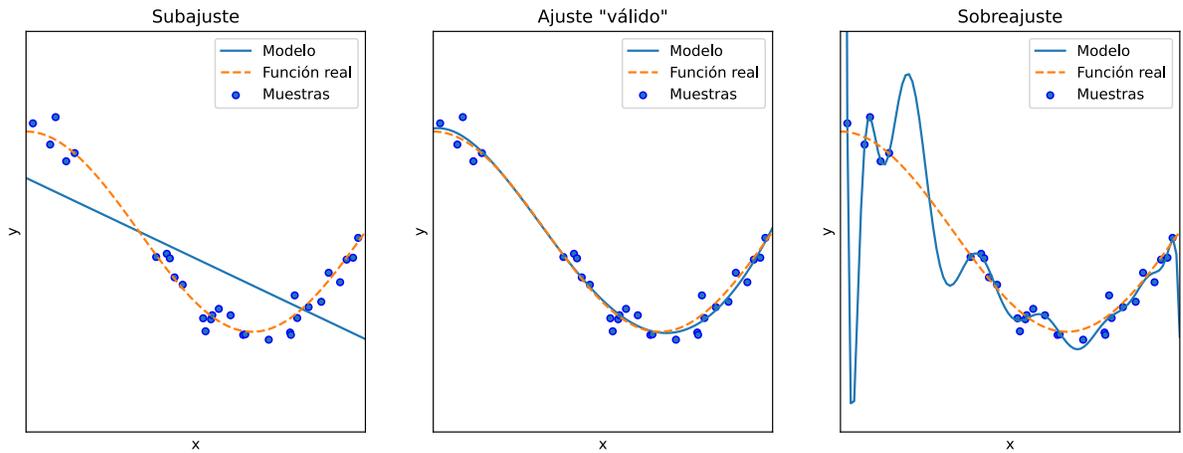


Figura 2.6: Ejemplos de subajuste y sobreajuste para el problema de regresión. Modificado de [14]

### 2.3.3. Error Cuadrático Medio (MSE)

Específicamente para el problema de regresión mediante redes neuronales, la función de pérdida más utilizada corresponde al error cuadrático medio (MSE por sus siglas en inglés), se define como el promedio de los cuadrados de los errores de cada estimación respecto de su valor real [11, 15], formalmente:

$$L_{mse} = \frac{1}{n} \sum (y - \hat{y})^2 \quad (2.11)$$

Con  $n$  el número de observaciones,  $y$  el valor real de la observación e  $\hat{y}$  la estimación del valor de  $y$ .

### 2.3.4. Predicción *Naive*

Por otro lado, para regresiones con datos en el tiempo (similar a un *forecasting*), es necesario el uso de autorregresores, es decir, observaciones de tiempo anterior que proporcionan información sobre el estado del fenómeno que se quiere estimar. Al estar presentes estos autorregresores, es común el sobreajuste de la red mediante la toma del valor del instante inmediatamente anterior al que se quiere estimar, obviando la información proporcionada por el resto de las variables exógenas. A esta mecánica de predicción se le llama *predicción naive* o “ingenua” [16]. Dado que es evidentemente una mecánica incorrecta, es necesario tener presente que ante menores tiempos de muestreo, el MSE de la predicción naive será menor, por lo que hay más probabilidad que la red tome ese camino. En consecuencia, previo al entrenamiento de una red para problemas de este tipo, es necesario utilizar el MSE de la predicción *naive* (al cual se le llama *naive MSE* en este trabajo, definido formalmente en la ecuación 2.12), para compararlo posteriormente al MSE de la red, el cual debe ser sustancialmente menor.

$$\text{naive MSE} = \frac{1}{n} \sum_{t=0}^n (y_{t-1} - y_t)^2, \text{ con } y_{-1} = y_0 \quad (2.12)$$

## 2.4. Test de Hipótesis

El test o contraste de hipótesis corresponde a un método en el área de la estadística, el cual se utiliza para decidir si los datos disponibles son suficientes para rechazar o no una hipótesis. Esta subsección se basa en [17] y describe los fundamentos básicos y la notación de un test de hipótesis.

Los principales conceptos en un test de hipótesis son: la **hipótesis nula**, denotada por  $H_0$  y la **hipótesis alternativa**, denotada por  $H_1$ :

- $H_0$  es la hipótesis que supone un punto de partida y afirma un estado de la población que desea ser evaluado. Usualmente impone un valor a cierta condición (por ejemplo: “El valor de la media de  $X$  es igual a  $\mu_0$ ”). El test usualmente surge cuando hay sospechas acerca de que  $H_0$  es incorrecta, por lo que uno de los objetivos es rechazar  $H_0$ .
- $H_1$  es la hipótesis que contradice  $H_0$ , y se formula proponiendo que el parámetro que se suponía en  $H_0$  es mayor, menor o simplemente distinto al valor impuesto (por ejemplo, en la misma línea del párrafo anterior: “La media de  $X$  es mayor a  $\mu_0$ ”).

El paso que sigue a la formulación del test es definir un criterio de decisión, para el que se define un **nivel de significancia**, que corresponderá al criterio de juicio sobre el cual se toma una decisión con respecto al valor establecido en una hipótesis nula. Equivalentemente, el nivel de significancia corresponde a la probabilidad de rechazar  $H_0$  cuando esta es verdadera. Usualmente en ciencias se utiliza un 5% de nivel de significancia [17].

## 2.5. Información Mutua

La información mutua es un concepto fundamental en la teoría de la información, y corresponde a la medida que representa la dependencia mutua de dos variables aleatorias, esto es, a nivel cualitativo, la cantidad de información que se obtiene de una variable aleatoria al observar la otra [18].

Formalmente, la información mutua entre dos variables aleatorias  $X$  e  $Y$  se define según la ecuación 2.13 para variables aleatorias discretas con probabilidad conjunta  $p(x, y)$ : y por la ecuación 2.14 para variables aleatorias continuas con función de distribución de probabilidad conjunta  $f(x, y)$ : [18]

$$\mathbf{MI}(X; Y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \quad (2.13)$$

$$\mathbf{MI}(X; Y) = \int \int f(x, y) \log \frac{f(x, y)}{f(x)f(y)} dx dy \quad (2.14)$$

De la definición puede inferirse que la información mutua entre variables no correlacionadas (i.e. variables independientes) es cero. Lo cual también puede verse cualitativamente al notar que al haber dos variables independientes, conocer algo sobre una no da información sobre la otra.

## 2.6. Detección y Diagnóstico de Fallas Basado en Modelo

Si se desea clasificar un área a la cual este trabajo pertenece, esta sería la detección de fallas basada en modelos. Tal como se anticipaba en el capítulo anterior (ver Figura 1.1), esta área de la detección y diagnóstico de fallas se basa en el principio de comparación de las señales de salida de un sistema y la salida de un modelo matemático ajustado a este para las mismas entradas, el cual supone un correcto funcionamiento. Esta comparación resulta no trivial (motivo por el cual existe esta área) y resulta aún más crítica cuando se está analizando un sistema en tiempo real. Es por este motivo que la proposición de diversas técnicas que garanticen soluciones en corto tiempo y demuestren efectividad resulta valiosa [19] (detección temprana de obsolescencia de modelos en el caso de este trabajo). La estructura principal de un sistema de detección de fallas basado en modelo las siguientes etapas:

- **Generación de residuo:** Se le llama residuo o señal residual a la función de las señales de entrada y salida en tiempo real que aporta a la detección de una falla (usualmente corresponde a la señal de error, equivalente a la resta entre la salida del sistema y la del modelo). Por su propia definición esta señal es independiente de las entradas del sistema en operación normal (i.e. cuando no hay falla); además el valor de esta es igual a cero en condiciones ideales y es notablemente mayor cuando hay una falla presente detectada. El problema es que no existen condiciones ideales y por lo tanto la señal de error usualmente es ruidosa, con lo que una falla incipiente puede pasar desapercibida hasta que exista una diferencia notable que puede desestabilizar el sistema. Dado lo anterior, la detección temprana de fallas es un desafío en sistemas reales. [2, 20, 21]
- **Toma de decisión:** Esta etapa consiste en el análisis de la señal de residuo para determinar la probabilidad de una falla, aplicando una regla de decisión. La acción más simple es imponer un límite de tolerancia para el valor del error y decidir si hay o no falla con este valor, lo cual no es ideal puesto que carece de fundamentos estadísticos y puede dejar pasar fallas (falso negativo) o bien detectar fallas donde no las hay (falso positivo). Dado esto, lo más usual es basarse en diversos métodos de la teoría estadística para el rápido análisis de la señal. [2, 20]

## 2.7. Estimación de la Información Mutua

Tal como se mencionó en la sección 2.5, la información mutua es un concepto fundamental en la teoría de la información; por otro lado, la aplicación real de la medición de la información mutua dista de la definición de esta, pues en casos reales solo puede existir un muestreo de poblaciones que tienen algún comportamiento, por lo que no es posible conocer la probabilidad o distribución conjunta real entre las variables a analizar. Es por esto que es necesaria una técnica de estimación de este valor.

Algunas de las propuestas más recientes sobre la estimación de la información mutua son las siguientes:

- *Mutual Information Neural Estimator (MINE)*: Un estimador basado en redes neuronales, disponible en la librería PyTorch. Fue desarrollado el año 2018 por M. Belghazi et al. [22]

- *scalable Mutual Information Estimation using Dependence Graphs*: Una técnica basada en grafos de dependencia cuyo resultado final demostró una menor complejidad computacional. Fue desarrollada en 2019 por M. Noshad et al. [23]
- *Data-Driven Representations for Testing Independence: Modeling, Analysis and Connection With Mutual Information Estimation*: De por sí no es un desarrollo para la estimación de la información mutua, pero relaciona el test de independencia basado en particiones estructuradas por árboles (TSP) con el problema de estimación de información mutua. Fue desarrollado por M. González, J. Silva, M. Videla y M. Orchard en 2022 [1]. Es precisamente en este estudio en el cual M. Videla se basa para el desarrollo de la librería TSP de python, utilizada en este trabajo para el test de información mutua.

## 2.8. Obsolescencia de Modelos (*mismodeling*) Basado en la Información Mutua

La idea principal de este trabajo es desarrollada previamente por el Information and Decision System Group de la Universidad de Chile en conjunto con el Centro Avanzado de Ingeniería Eléctrica y Electrónica de la Universidad Técnica Federico Santa María en [4]. En esta sección se muestran dos experimentos relevantes que fueron útiles para el desarrollo del trabajo principal de esta memoria. Los experimentos preliminares que demuestran el potencial de la librería TSP-IT fueron desarrollados por M. Videla en enero de 2022 [24], mientras que el análisis teórico que muestra la factibilidad de aplicación del test en el sistema de estanque cónico fueron desarrollados por C. Ramírez en 2022 [25].

### 2.8.1. Experimentos Previos (M. Videla, 2022)

Estos experimentos consistieron en el análisis de distintas señales de error para modelos ajustados a datos de sistemas lineales y autorregresivos. En estos experimentos, Videla logra demostrar el gran potencial de la librería y del test como tal, ya que ante perturbaciones en el sistema, los resultados indican que el test es capaz de detectar estos cambios y, además, dar indicios de qué entrada es la que está relacionada a la perturbación, lo que es extremadamente bueno para la hipótesis de este trabajo. Esto puede verse mejor en los gráficos de la Figura 2.7, en la cual se puede ver un incremento de la información mutua ante la presencia de perturbaciones, y además se ve una diferencia importante de este valor para las distintas entradas.

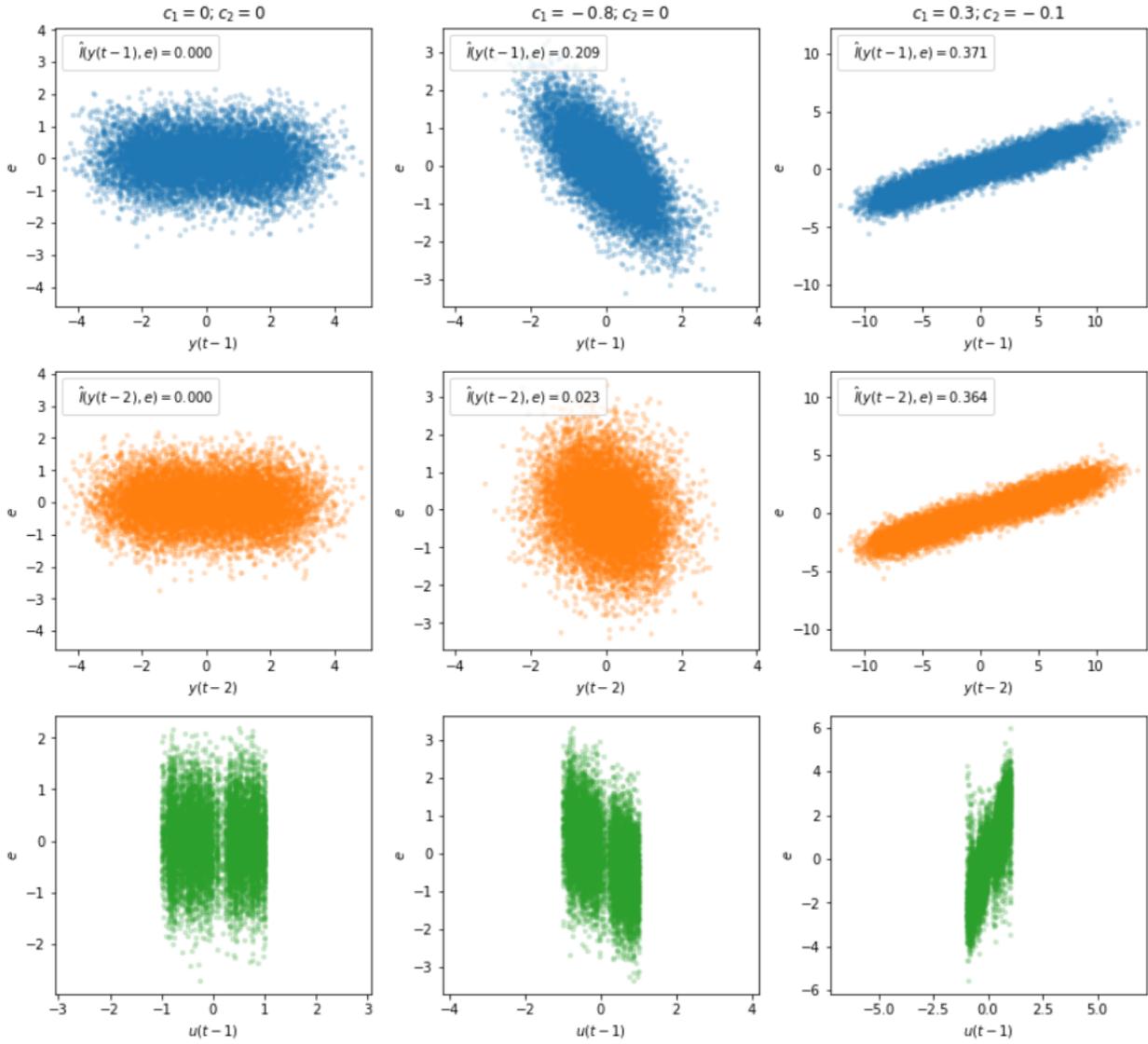


Figura 2.7: Resultados de uno de los experimentos desarrollados por M. Videla en 2022 [24]. En los gráficos se muestra la proyección de los valores de la señal de error en un modelo autorregresivo, con respecto a las diversas entradas de cada modelo. En la primera columna no hay perturbaciones, mientras que en las dos siguientes sí. En las dos primeras filas puede además verse la estimación de la información mutua entre el error y las variables de entrada. Nótese cómo se incrementa la información mutua ante la presencia de perturbaciones.

## 2.8.2. Análisis teórico del problema (C. Ramírez, 2022)

Los experimentos teóricos realizados corresponden a un análisis de sensibilidad del test para distintos valores de los coeficientes de falla mostrados en las ecuaciones 2.6 y 2.7. Para operar en tiempo discreto, se pueden reformular las ecuaciones para obtener una aproximación a ecuación de diferencia [25]:

$$\dot{h}_c = \frac{F_{in} - F_{out}}{0,63h_c^2 + 11,4h_c + 17,1} \implies (h_t - h_{t-1}) = \left( \frac{(\alpha_1 + \delta_1)f_{t-1} + \alpha_2 - (\beta + \delta_2)\sqrt{h_{t-1}}}{0,63h_{t-1}^2 + 11,4h_{t-1} + 17,1} \right) \Delta t \quad (2.15)$$

$$y_t = h_t + w_t \quad (2.16)$$

Con  $y_t$  la observación en tiempo  $t$ ,  $h_t$  la altura real del agua en el tiempo  $t$ , y  $w_t$  un ruido aditivo libremente distribuido en el instante  $t$ . Con estas ecuaciones, la expresión teórica del residuo (asumiendo un modelo con ajuste perfecto del fenómeno) está dada por:

$$r_t = \frac{\delta_1 f_{t-1} - \delta_2 \sqrt{h_{t-1}}}{0,63h_{t-1}^2 + 11,4h_{t-1} + 17,1} + w_t \quad (2.17)$$

Esta expresión muestra que existe dependencia entre los coeficientes de falla y el residuo, además se puede ver que cuando  $\delta_1 = 0$ , la dependencia del residuo con respecto a  $f_{t-1}$  desaparece y solo existe dependencia con respecto a  $h_{t-1}$ , por lo que en escenarios de independencia de estas dos variables, el aislamiento de una falla es factible.

Con estas expresiones, se estimó la información mutua para 2000 muestras para distintas combinaciones del par  $(\delta_1, \delta_2)$  y distintos niveles de magnitud del ruido  $w_t$ . Los resultados se muestran en los mapas de calor de la Figura 2.8; en ella, se muestra en la escala de colores el valor de la información mutua estimada, en el eje x se muestran los valores de  $\delta_2$ , mientras que en el eje y se muestran los valores de  $\delta_1$ . En la primera columna se muestra la información mutua del residuo con respecto al porcentaje de utilización de la bomba  $f$  en tiempo  $t$ , mientras que en las columnas 2 y 3 se muestra la información mutua del residuo con respecto a  $f$  y  $h$  respectivamente, para el tiempo  $(t-1)$ . Cada fila corresponde a un orden de magnitud del ruido diferente (dado por la desviación estándar  $\sigma$ ), siendo  $\sigma = 1, 0.1, 0.01, 0.001$  y  $0.0001$  para las filas 1, 2, 3, 4 y 5 respectivamente. Notar que estos resultados muestran una fuerte dependencia del orden de ruido, siendo imposible la detección de obsolescencia para un ruido alto (primera fila). Notar también que la detección es más sensible a variaciones de  $\delta_1$  que de  $\delta_2$ .

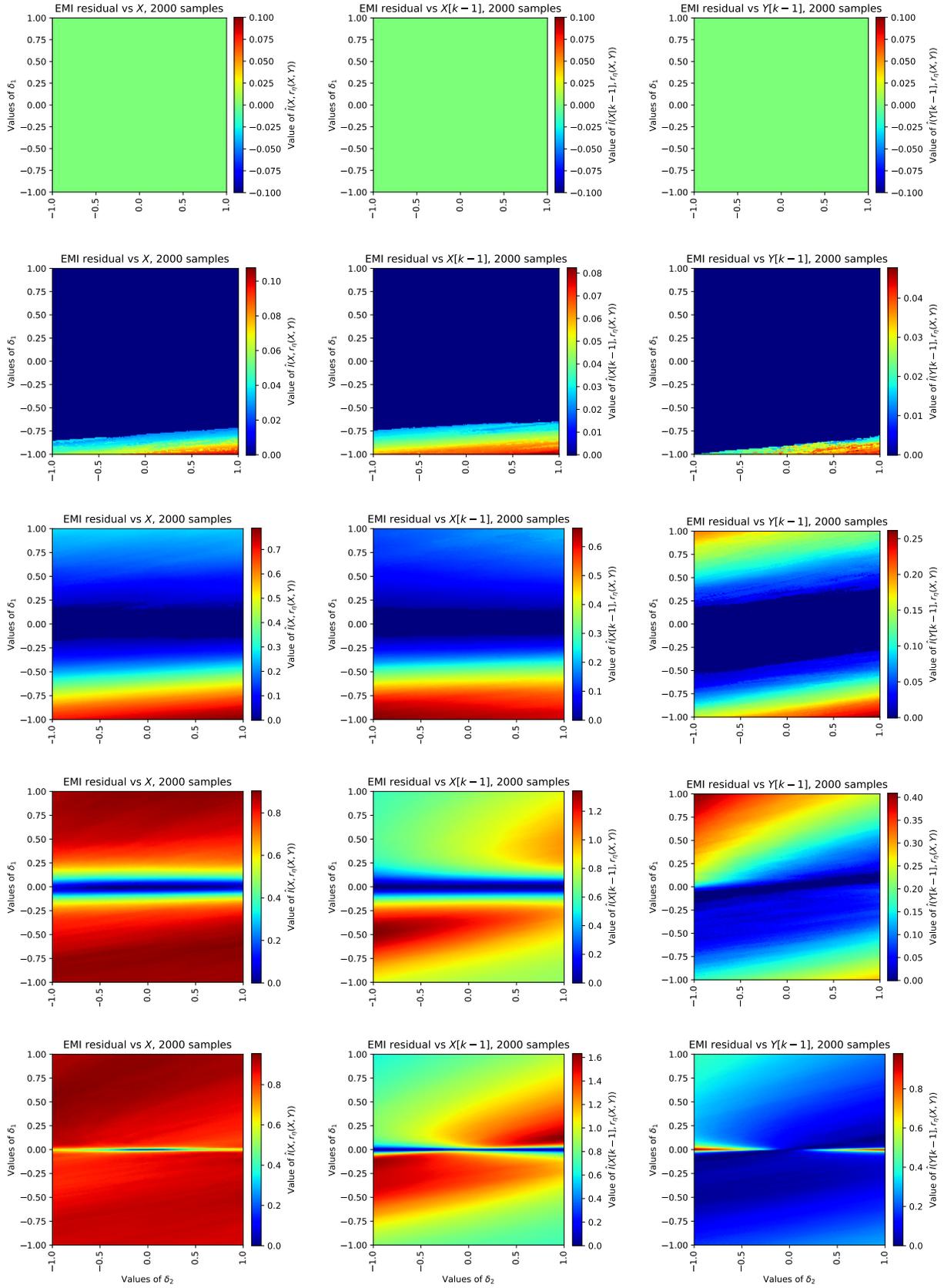


Figura 2.8: Resultados del análisis teórico. Extraído de [25]

# Capítulo 3

## Metodología

### 3.1. Flujo de Trabajo

#### 3.1.1. Generación de Datos

Para la generación de datos, tal como se anticipa en el capítulo anterior, se utilizaron los software MATLAB y Simulink, mediante los cuales se realizó la simulación del sistema de estanque cónico con control de nivel PID. Para tener bajo control los tiempos de muestreo (i.e. operar con control a tiempo discreto) se utilizó un retenedor de orden cero (ZOH por sus siglas en inglés), herramienta que permite setear un valor sensado por un tiempo definido. El esquema de funcionamiento del sistema completo con el control de nivel es resumido en la Figura 3.1. El rango de operación del sistema con el control de nivel es de entre  $[15 - 60]$  [cm], esto es debido a que a niveles más bajos, la bomba es incapaz de mantener un punto de operación estable, mientras que a niveles más altos se arriesgan desbordes [5].

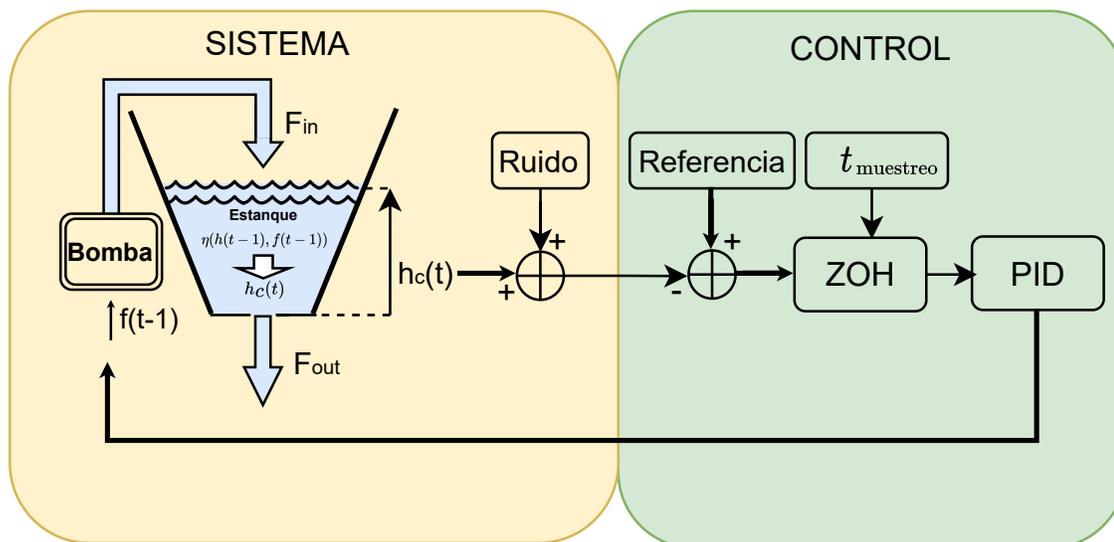


Figura 3.1: Resumen del sistema completo con control de nivel a lazo cerrado. Se muestra al estanque modelado por un fenómeno de tiempo discreto  $\eta$  dependiente del porcentaje de utilización de la bomba  $f$  y de la altura del agua  $h$  en un instante  $(t - 1)$  para dar sitio al siguiente nivel de agua  $h(t)$ .

De esta manera, se pueden generar grandes volúmenes de datos para distintos escenarios, controlando las variables de referencia, ruido, tiempo de muestreo, altura inicial y parámetros PID. El archivo resultante posterior a la realización de una simulación es un *.csv* de cuatro columnas: ‘tiempo’, ‘referencia’, ‘hc’, ‘f\_pid’; correspondientes a (como su nombre lo dice) el tiempo (en segundos), el valor de la referencia (en centímetros), el valor de altura de agua medido para ese instante de tiempo (en centímetros), y el valor del porcentaje de utilización de la bomba calculado por el controlador PID (en porcentaje); respectivamente. Todos estos datos están equiespaciados según el tiempo de muestreo definido al inicio de la simulación.

### 3.1.2. Ajuste de Modelos

Los modelos generados están basados en una arquitectura de red neuronal *fully connected*, que recibe como entrada los dos primeros autorregresores de  $h$  y  $f$  (esto es,  $h(t-1)$ ,  $h(t-2)$ ,  $f(t-1)$  y  $f(t-2)$ ), para predecir el valor en tiempo actual de  $h$ . La red entrenada generadora de todos los modelos a evaluar tiene 3 capas ocultas de 200, 50 y 10 neuronas. La función a optimizar durante el entrenamiento es el error cuadrático medio (MSE), valor que tendrá un límite óptimo (desconocido) generado por la presencia de ruido en la señal, ya que un ajuste casi perfecto del modelo (MSE  $\approx 0$ ) implicaría un sobreajuste de la red. Para evitar lo anterior, se reserva un dataset del 20% del tamaño del set de entrenamiento (no necesariamente un subconjunto de este) como set de validación, guardando exclusivamente los modelos que mejoren el MSE tanto del set de entrenamiento como del set de validación (ver Figura 3.2). De esta forma se controla que la disminución de la función de pérdida sea resultado de una “comprensión” correcta del fenómeno más que un sobreajuste a los datos de entrenamiento. Cabe destacar que para cada tiempo de muestreo distinto, así como para distintas configuraciones del controlador PID, es necesario ajustar un nuevo modelo, dado que la red tiene el objetivo de “captar” lo mejor posible la dinámica del controlador usado para generar los datos.

El modelo se entrena por 10000 épocas. Finalizada esta etapa, basta con ingresar los datos correspondientes a los autorregresores de  $h$  y  $f$ , para generar una predicción de  $h$  en el tiempo actual. A pesar de tener un conjunto de validación, es necesario evaluar el modelo mediante un set de prueba, correspondiente a un conjunto de datos de operación del sistema en condiciones normales, para los mismos rangos de operación que el set de entrenamiento, pero con distintos cambios de referencia y generación aleatoria de ruido (con distinta semilla de generación).

Finalmente, para efectos de este trabajo, se define como **modelo suficientemente válido** aquel que logre un MSE menor al 5% del MSE de la predicción *naive* para el conjunto de validación. Esta definición surge para evitar que la red tome una vía de ajuste equivocada como lo es la predicción *naive*. Tal como se menciona en el capítulo anterior, el costo para alcanzar un error cuadrático medio que cumpla con este criterio es mayor para tiempos de muestreo menores, dado que la predicción *naive* es cada vez mejor. De la misma manera, es posible que para ciertos niveles de ruido altos no sea posible cumplir con este criterio, ya que el MSE óptimo podría ser un valor entre el MSE del modelo ingenuo y el 5% de este.

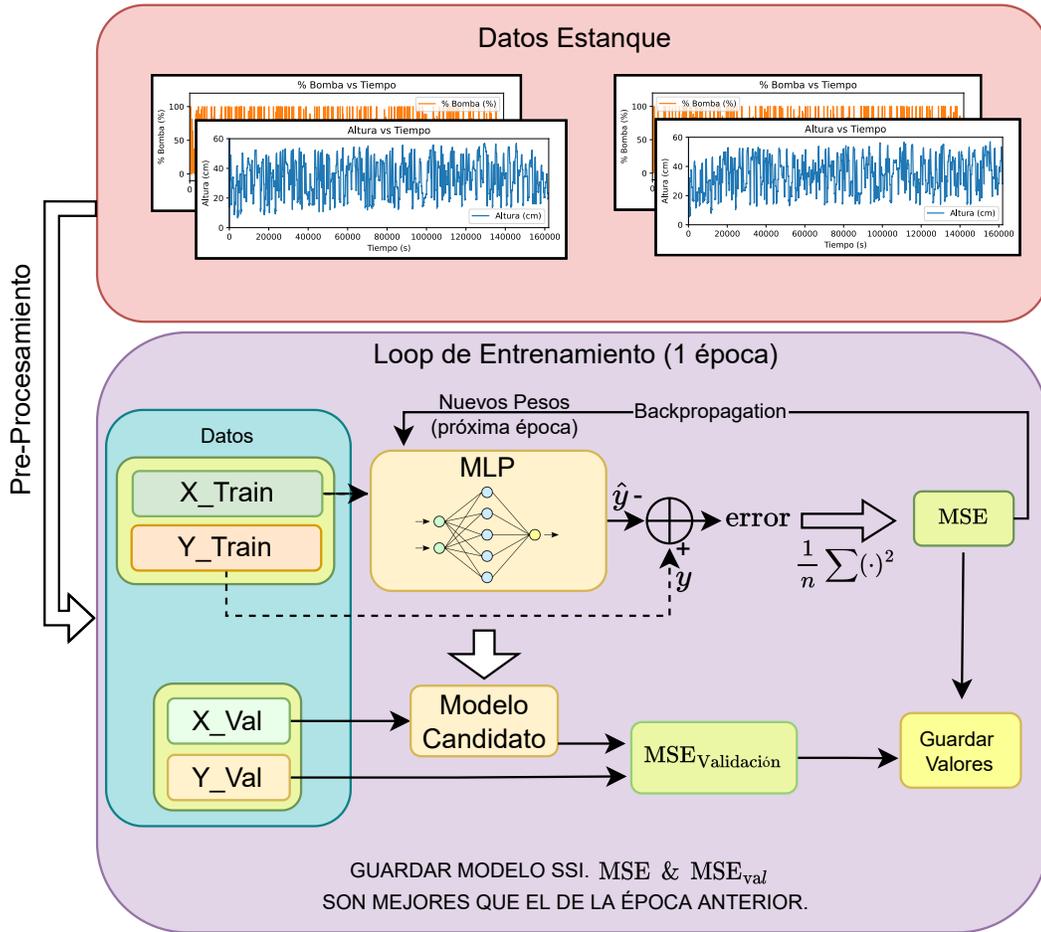


Figura 3.2: Esquema del *loop* de entrenamiento para una época. Notar que el modelo se guarda dependiendo del valor del MSE logrado tanto para el dataset de entrenamiento como para el de validación.

### 3.1.3. Operación del Modelo, Generación del Residuo y Aplicación del Test de Información Mutua

Una vez se posee el modelo ajustado y validado para los criterios definidos, este puede operar como supervisor del sistema, recibiendo como entradas las dos últimas muestras de  $f$  y  $h$ , para posteriormente estimar el valor siguiente de  $h$ . Con esta predicción, se puede comparar una a una la data real (oráculo) contra las estimaciones, para generar de esta forma el residuo, tal como se muestra en la ecuación 3.1:

$$r_i = y_i - \hat{y}_i \quad (3.1)$$

En donde  $y_i$  e  $\hat{y}_i$  corresponden a la  $i$ -ésima muestra tomada y estimada respectivamente (correctamente compaginadas a su tiempo correspondiente), con lo que  $r_i$  es la  $i$ -ésima fila del vector residuo  $\vec{r}$ , cuyo largo es igual al total de muestras. Dada la rápida respuesta del modelo ante una entrada ( $\ll 1[s]$ ), la supervisión en tiempo real es factible, al menos para tiempos de muestreo superiores a  $1[s]$ .

Para la obtención de la información mutua estimada (EMI), se utiliza el análisis mediante ventana deslizante, tal como se muestra en la Figura 3.3, en la cual se define una ventana con el tamaño suficiente para albergar una cantidad de datos que permita una estimación verosímil de la información mutua. Para efectos de este trabajo, se experimentará con ventanas  $\geq 150$  muestras. Cabe destacar que en los casos borde (es decir, al inicio del proceso) se rellenará la ventana con valores nulos, de manera que se tenga en consideración solo la data existente.

Sumado a lo anterior, es necesario destacar que el algoritmo TSP requiere de tres parámetros como entrada, estos son [1, 24]:

- $1\_bn \in (0, 1/3)$ : Exponente de la **aproximación del *threshold*** del criterio de refinamiento de las particiones estructuradas por árbol (TSP).
- $w\_bn \geq 0$ : Factor de peso (ponderación) de la **aproximación del *threshold*** del criterio de refinamiento de las TSP.
- $lambda > 0$ : Factor de regularización de las TSP (llamado  $\alpha$  en el paper principal [1]).

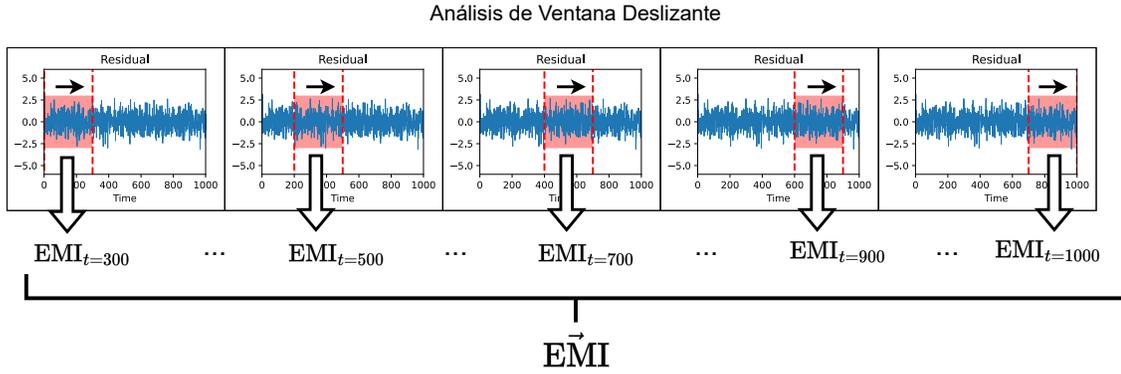


Figura 3.3: Análisis de ventana deslizante. Se tiene el residuo como serie temporal con una ventana de tamaño fijo deslizándose continuamente a través de esta señal. Por cada paso que da la ventana se estima la información mutua de estos datos y se guarda el valor. En la imagen se le llama EMI a la estimación de información mutua.

En donde la “aproximación del *threshold* del criterio de refinamiento” antes mencionada corresponde a la estimación del parámetro  $b_n$  del algoritmo TSP; este parámetro “*hace fuertemente consistente para detección de independencia a un esquema  $\phi$  si  $b_n \approx n^{-l}$  (con  $n$  el número de muestras y  $l \in (0, 1/3)$ )*” [1], y es utilizado en el algoritmo como  $b_n = w \cdot n^{-1}$  por construcción (Ver Teorema 3 del paper principal).

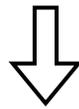
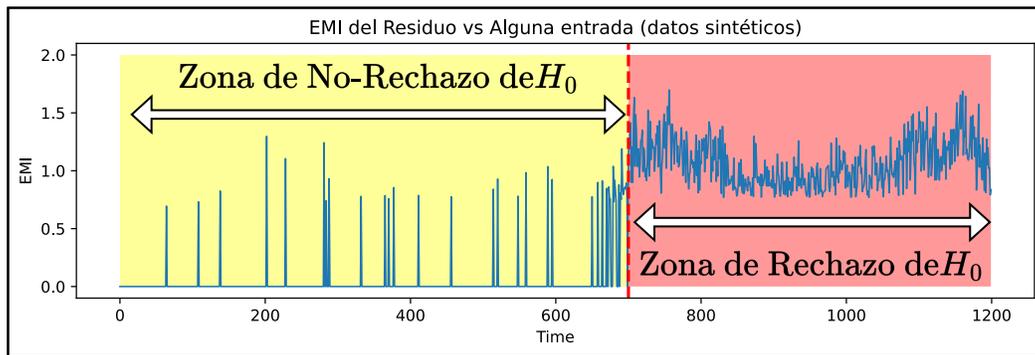
La elección de estos parámetros para la tarea que se desea realizar no es trivial y es posible que no se esté operando con la óptima configuración. La elección de los parámetros  $1\_bn$  y  $w\_bn$  está basada en las opciones sugeridas en el análisis realizado por M. Videla en [26], esto es,  $(1\_bn, w\_bn) = (0.001, 0.1)$ .

El parámetro  $lambda$ , en cambio, tiene mucho peso en cuanto a la seguridad del no-rechazo de la hipótesis nula, esto es, marca un umbral bien definido de información mutua

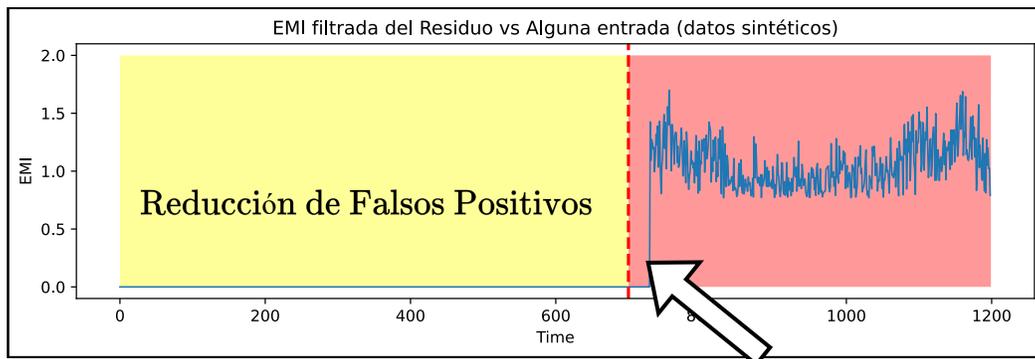
estimada por debajo del cual se estimará  $EMI = 0$ . Para los experimentos realizados se realizó una variación del parámetro  $\lambda$  obteniendo distintos resultados, obteniendo en el caso óptimo una nula tasa de falsos positivos en varios experimentos.

### 3.1.4. Análisis de Validez

Finalizada la realización del test, se observarán alzas en los valores de la información mutua estimada (detección de obsolescencia). Ante estas detecciones, es necesario demostrar que son debido a un correcto funcionamiento del test y no producto del azar. Para ello se realizará otro test de hipótesis, que planteará como hipótesis nula que una detección del test corresponde a un falso positivo (y por lo tanto que el modelo sigue vigente), mientras que la hipótesis alternativa sostendrá que esta detección es cierta y por lo tanto se debe a una obsolescencia del modelo.



Filtrar si  $\exists EMI_i = 0 \in \vec{EMI}_{\text{last } \frac{1}{3}} \cdot (\text{EMI window size})$



Aumento de Falsos Negativos

Figura 3.4: Simulación de la metodología de filtrado con datos sintéticos, notar que alzas intermitentes cuasi-aleatorias en la EMI no son suficientes para concluir detección de obsolescencia. Notar además que en el inicio de la zona roja (primeros valores) aún no se tienen los datos suficientes para decidir rechazar  $H_0$ , lo que genera un retardo en la detección respecto al tiempo presente, equivalente al número de muestras del filtro multiplicado por el tiempo de muestreo.

La forma más simple de abordar este problema, es mediante un análisis cualitativo hacia el futuro, esto es, ante un alza en la información mutua estimada, tener un cierto número de muestras adyacentes que fortalezcan la hipótesis alternativa (o bien, ayuden a rechazarla). De esta forma, se considerarán  $n$  muestras hacia atrás que estén por sobre una  $EMI = 0$  como suficientes para rechazar la hipótesis nula y decidir que se está frente a una obsolescencia del modelo; en donde  $n = \frac{1}{3}$  de las muestras tomadas por la ventana deslizante en el desarrollo del test de información mutua. En caso contrario, se decidirá que no existe información suficiente para concluir la detección de obsolescencia (no se rechaza la hipótesis nula).

Pese a la simplicidad del mecanismo, este permite reducir sustancialmente el error tipo 1 (i.e. falsos positivos), comúnmente causado por “confusiones” generadas por la dinámica del ruido en la ventana de datos que se está analizando, y que desaparecen conforme esta ventana cambia el escenario. La situación anterior no ocurre ante la obsolescencia efectiva del modelo, pues la MI estimada se mantendría por sobre 0 hasta que se solucione el problema, por lo que, dejando de lado fallas intermitentes, este método no invisibiliza casos en los que hay una objetiva detección, salvo cuando no se tienen aún las 30 muestras completas (ver Figura ). Los puntos en contra de lo anterior son claros, y corresponden a que ante casos borde (i.e. cuando la detección es difusa o intermitente), siempre se decidirá no rechazar  $H_0$ , por lo que aumenta la tasa de error tipo 2 (falsos negativos).

### 3.1.5. Observación Relevante: Obsolescencia $\neq$ Falla

Es fundamental tener presente los supuestos planteados previo a detectar una falla mediante este mecanismo, esto pues, en la hipótesis del problema se indica que la detección se da ante un modelo representativo del sistema en cuestión, por lo que un modelo inadecuadamente ajustado estará obsoleto incluso para cuando el sistema opere bajo condiciones normales, y, por lo tanto, la detección no será la deseada sino que acusará sobre un modelo mal entrenado. Es por este motivo que en la sección 3.1.2 se define un mínimo MSE para tener un modelo válido, sin embargo, esta definición no garantiza un modelo que capte la dinámica del fenómeno en cuestión, por lo que este sistema basado en la información mutua también resultaría útil en la evaluación de modelos (con los suficientes datos del sistema operando en condiciones normales conocidas).

### 3.1.6. Análisis de resultados y conclusiones

Finalizado un experimento y pasado por el filtro de validación, este se considerará como exitoso si logra detectar una falla en tiempo real (dentro del margen de tolerancia de la ventana deslizante definida) y si sostiene la detección para el sistema operando continuamente en falla (es decir, que la EMI no decaiga a 0 durante la operación en falla).

Si para algún cierto grupo de experimentos se logra además el aislamiento de una falla (situación que es anticipada por la expresión teórica del residuo, explicado en la sección 2.8.2), el test gana valor, en el sentido que permite orientar la revisión del sistema hacia un sector por sobre otro, decisión que no es trivial y que es más compleja que el problema de simple detección.

## 3.2. Datos y Modelos Generados

En esta sección se muestran las distintas configuraciones de experimentos a realizar. Respecto a los datos, se considerarán distintos niveles de ruido, tiempos de muestreo y parámetros del controlador PID; mientras que respecto al test, se experimentará con distintos valores de  $\lambda$  y del tamaño de la ventana deslizante.

### 3.2.1. Nivel de Ruido

Tal como se muestra en la sección 2.8.2, el orden de magnitud del ruido de medición del nivel de agua afectaría negativamente en la efectividad del test, por lo que es relevante experimentar con los mismos ordenes de ruido mostrados en esa sección, esto es, un ruido gaussiano aditivo con  $\mu = 0$  y  $\sigma = \{10^{-1}; 10^{-2}; 10^{-3}; 10^{-4}\}$  [cm]. Se considerará un ruido despreciable para la variable  $f$ , es decir, se entrenará el modelo con la misma señal  $f$  que le indique el controlador PID a la bomba.

### 3.2.2. Tiempos de Muestreo

Debido a que se requiere de un modelo causal capaz de “entender” el fenómeno detrás de los datos, es necesario que el tiempo de muestreo del controlador PID sea el mismo que el de los datos que irán de entrada a la red neuronal; en consecuencia, el control de nivel debe ser en tiempo discreto. Esto conlleva al dilema sobre qué tiempo de muestreo resulta ideal, dado que desde el punto de vista del control de sistemas se logrará un mejor control ante un tiempo de muestreo menor, mientras que desde el punto de vista del ajuste de modelos, un valor pequeño de esta variable puede dar lugar a un peor ajuste de la red, y desde el punto de vista del test de información mutua, un valor muy grande del tiempo de muestreo generaría mayor retardo en la detección.

Dado lo anterior, se experimentará con distintos tiempos de muestreo: 5, 10 y 25 [s]; rango limitado inferiormente por el alto costo computacional que requeriría ajustar una red para la obtención de un modelo suficientemente representativo con un tiempo de muestreo menor; y acotado superiormente por el tiempo de reacción del controlador ante el constante vaciado del estanque (aunque esto realmente depende de los parámetros PID, se eligió la menor cota que permite un correcto control de nivel para los valores PID utilizados en los experimentos).

### 3.2.3. Parámetros PID

Los parámetros PID utilizados en estos experimentos tienen como base los valores encontrados mediante la sintonización PSO hecha por C. Jáuregui en [5]. Dado que el la combinación sugerida para un control óptimo posee una gran ganancia proporcional que fuerza a la bomba a trabajar en rangos de 0 a 100 % entre un instante de tiempo y otro, es que se optó por reducir la ganancia proporcional a  $\frac{1}{6}$  de esta, sacrificando precisión en el régimen permanente, pero ganando varianza en la variable  $f$ , lo que es útil para el análisis de señales por medio del test que se está implementando, y además facilita el ajuste de modelos del tipo red neuronal MLP. Los parámetros finales utilizados son  $P = 17.88$ ;  $I = 9.41 \cdot 10^{-5}$  y  $D = 4.44$ .

### 3.2.4. Fallas

Las fallas modeladas serán de dos tipos, tal como se anticipó en la sección 2.1, estas son, falla en la bomba y en la salida del agua, modeladas por las constantes de perturbación ( $\delta_1, \delta_2$ ). Para cada una de ellas, se modelan tres fallas, catalogadas según su gravedad: leve, media y grave. Estas son impuestas según la variación del valor nominal de la constante afectada por el factor de perturbación. De esta manera, una falla leve será aquella que varíe el 1 % del valor nominal de  $\alpha_1$  y  $\beta$ ; mientras que las fallas media y grave corresponderán a aquellas en que el valor nominal de estas constantes se vea afectado en un 5 y 10 % respectivamente.

# Capítulo 4

## Resultados Experimentales y análisis

En este capítulo se mostrarán los resultados experimentales para los tiempos de muestreo de 25, 10 y 5 segundos.

**Todos los resultados corresponden a una operación por 90 horas del estanque,** con cambios de referencia aleatorios dentro del rango de operación definido en 3.1.1 cada 400 segundos. De estas 90 horas, se simulan las primeras 45 en estado normal y las últimas 45 operando en falla constante (i.e. valores de  $\delta_1$  o  $\delta_2 \neq 0$ , sin variación en el tiempo). Cabe destacar que los gráficos presentados están en la misma escala en el eje  $X$ , mas no en el eje  $Y$ , esto pues, ante cada experimento cambia el nivel de información mutua estimada y por lo tanto la visualización de detección no sería intuitiva si se incluyesen imágenes con una escala estándar. Se utilizó la misma semilla de generación de las componentes aleatorias para que los resultados sean comparables.

### 4.1. Muestreo cada 25 segundos

En esta sección se muestran los resultados para un tiempo de muestreo de 25 segundos, en esta primera instancia se define la estructura de los experimentos siguientes, y se realiza un análisis previo de los parámetros intrínsecos al test, estos son, el valor de `lambda` y el tamaño de la ventana deslizante. En la Tabla 4.1 se muestran los distintos valores del MSE en el set de prueba, tanto del modelo ingenuo como del modelo utilizado para cada experimento según su orden de ruido.

Tabla 4.1: Valores del MSE acorde al nivel de ruido para los modelos con tiempo de muestreo de 25 segundos. Valor calculado en el set de prueba respectivo.

Orden del Ruido	<i>Naive</i> MSE	Test MSE
$10^{-1}$	22.8	0.28
$10^{-2}$	22.81	0.029
$10^{-3}$	22.73	0.0079
$10^{-4}$	23.25	0.0053

### 4.1.1. Filtrado

Se mostrarán resultados sin filtrar para posteriormente compararlos con los resultados filtrados según el criterio definido en el análisis de validez. Se mostrarán los efectos del filtrado solo para un set de experimentos para evitar redundancia. Posterior a ello se mostrarán solo resultados filtrados.

### 4.1.2. Parámetro `lambda`

Como se menciona en la sección 3.1.3, el algoritmo posee un parámetro llamado `lambda`, correspondiente al parámetro de regularización, y que permite establecer una mínima información mutua estimada para el rechazo de la hipótesis nula. Es por ello que resulta relevante experimentar con variaciones de este parámetro, puesto que ayuda a filtrar resultados cuya información mutua alcanzada es relativamente pequeña respecto a la esperada en una falla.

Dado que realizar una experimentación sistemática de varios valores de `lambda` elevaría fuertemente el número de experimentos a realizar (y por lo tanto el tiempo de ejecución), es que se realiza esta variación solo en el caso de  $t_{\text{muestreo}} = 25[s]$ , en el cual se analiza y se decide el valor de `lambda` para los siguientes experimentos a realizar. Concretamente se experimentó con `lambda`  $\in \{7.5 \cdot 10^{-5}; 7.5 \cdot 10^{-6}; 7.5 \cdot 10^{-4}\}$  (siendo el primero, el valor sugerido, y los restantes correspondientes a variaciones en un orden de magnitud para evaluar la sensibilidad del parámetro). Posterior a este análisis, todos los resultados expuestos son con `lambda` =  $7.5 \cdot 10^{-5}$  a menos que se indique lo contrario.

### 4.1.3. Tamaño de la ventana deslizante

Finalmente, en la misma línea de lo anterior, se experimentará con distintos tamaños de la ventana deslizante, (pero para el primer valor de `lambda` para evitar resultados redundantes). En particular, se utilizarán ventanas de 720, 1440 y 2160 muestras para cada tiempo de muestreo, con el objetivo de concluir si más datos implican directamente mejores resultados.

### 4.1.4. Resultados

El esquema de los resultados mostrados en esta memoria consiste en exponer en una sola Figura varias realizaciones del experimento para distinta gravedad de falla y distinto nivel de ruido. Esto es debido a que ayuda a la visualización y comparación de los experimentos cuando el número de estos es elevado (se realizaron más de 100 experimentos). En las figuras individuales, se puede ver una **línea vertical azul**, correspondiente al instante en el cual se gatilla la falla correspondiente, mientras que a la vez se visualiza una **línea vertical amarilla**, correspondiente al paso de tiempo equivalente a una (1) ventana deslizante desde el inicio de la falla.

#### 4.1.4.1. Ventana de 720 muestras (5 horas), $\lambda = 7.5 \cdot 10^{-5}$

#### Resultados sin filtrar

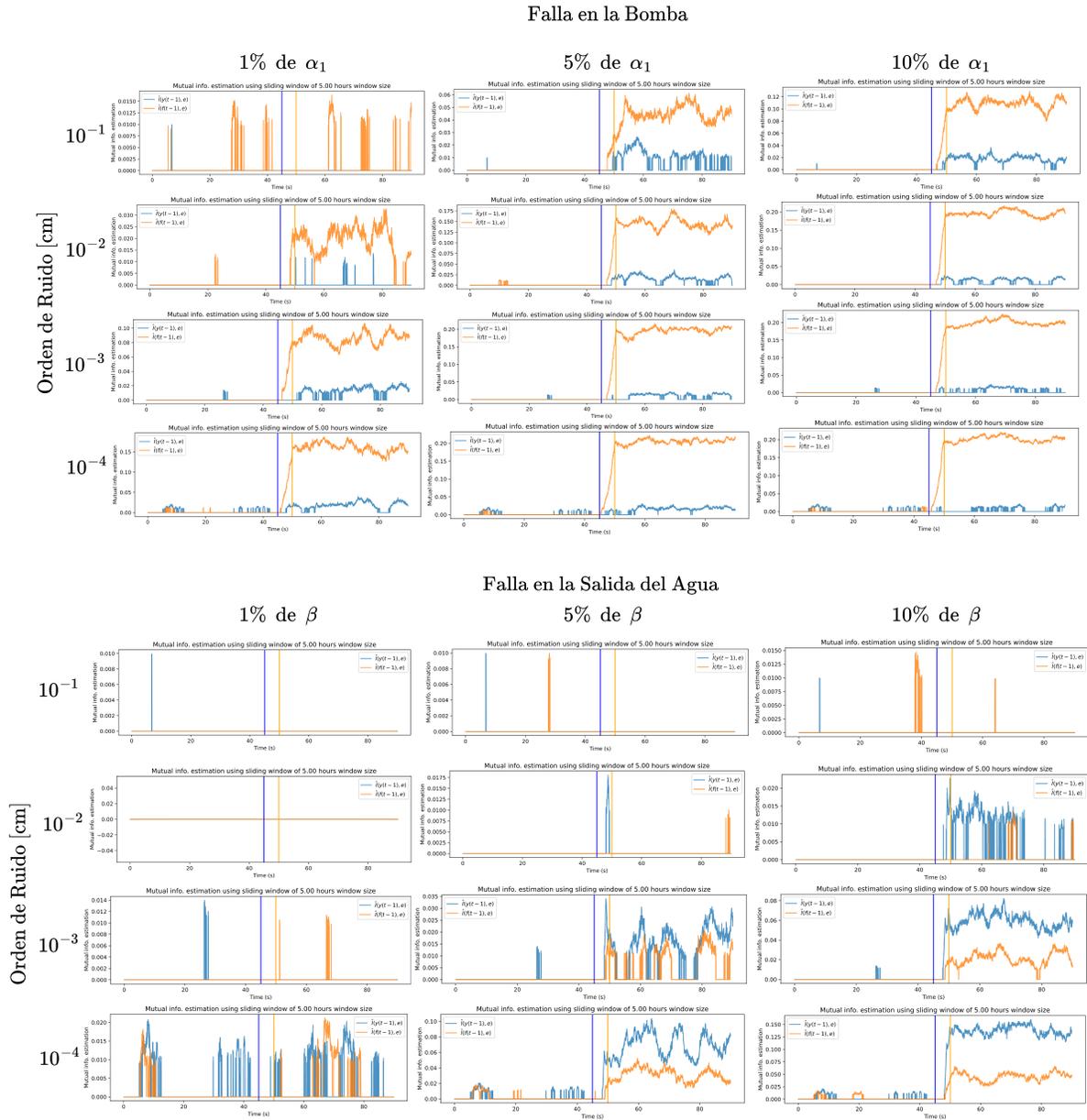


Figura 4.1: Resultados para un tiempo de muestreo de 25 [s], sin filtrar, se muestran fallas en la bomba y en la salida del agua, clasificadas por su gravedad y nivel de ruido del sistema.

# Resultados filtrados

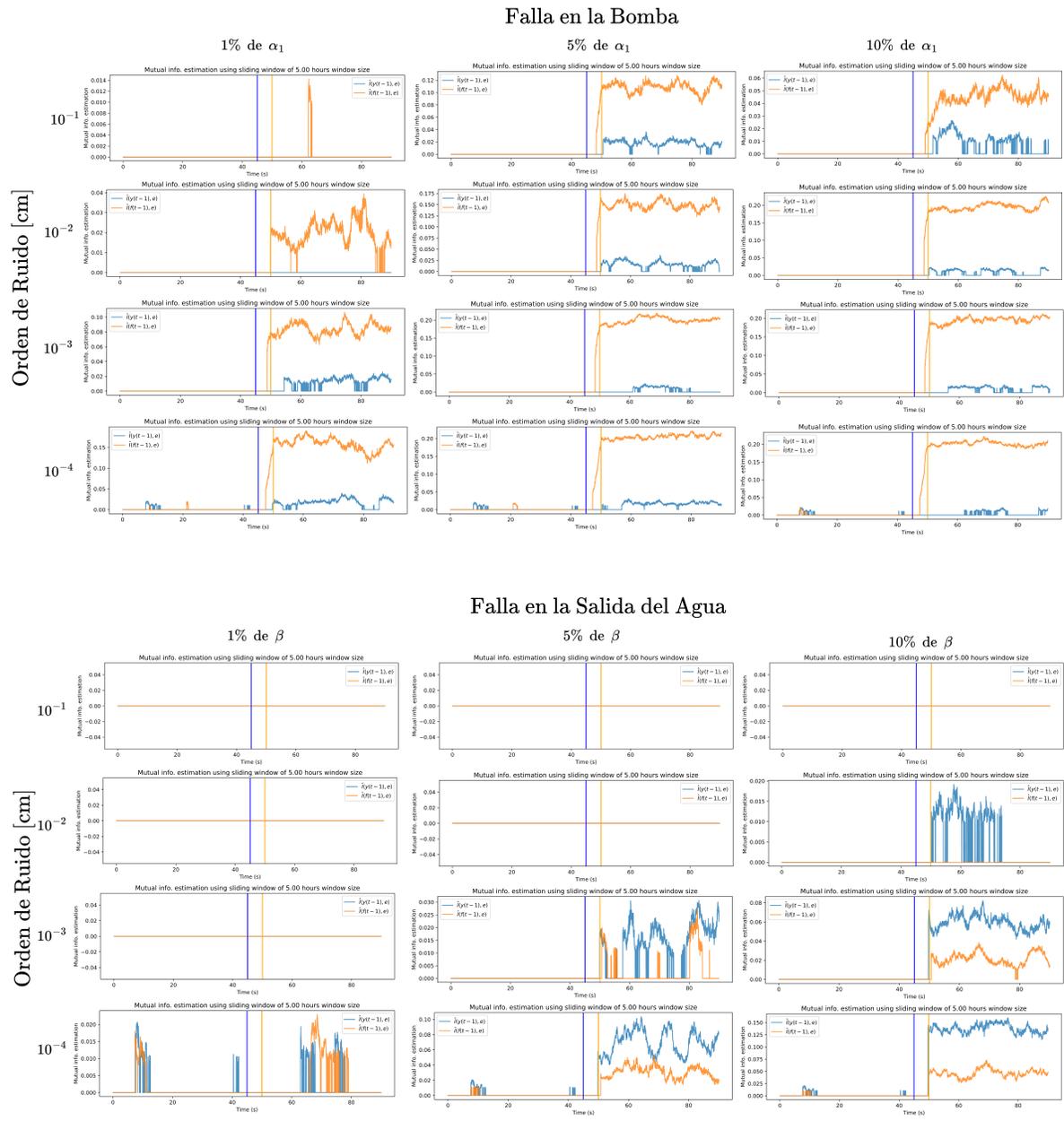


Figura 4.2: Resultados filtrados. Notar la reducción de los *peaks* aislados de información mutua presentes en la Figura 4.1

#### 4.1.4.2. Ventana de 720 muestras, $\lambda = 7.5 \cdot 10^{-6}$

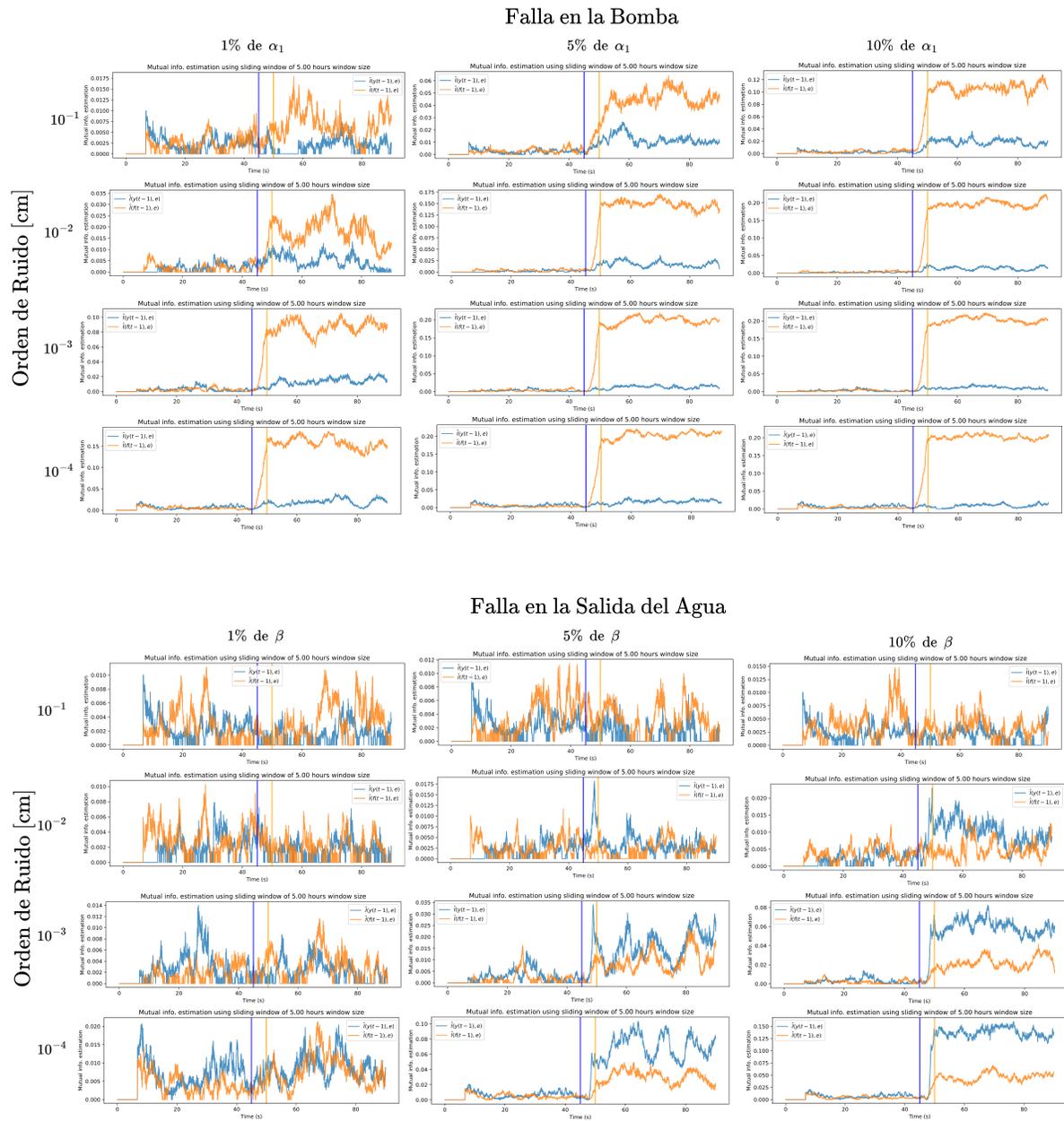


Figura 4.3: Resultados para un valor de  $\lambda$  un orden de magnitud menor que en la Figura 4.2. Notar que el filtrado no es efectivo en la reducción de falsos positivos dado que el test está constantemente estimando valores de la información mutua distintos de cero y sólo aumenta el valor de esta posterior a algunas fallas.

### 4.1.4.3. Ventana de 720 muestras, $\lambda = 7.5 \cdot 10^{-4}$

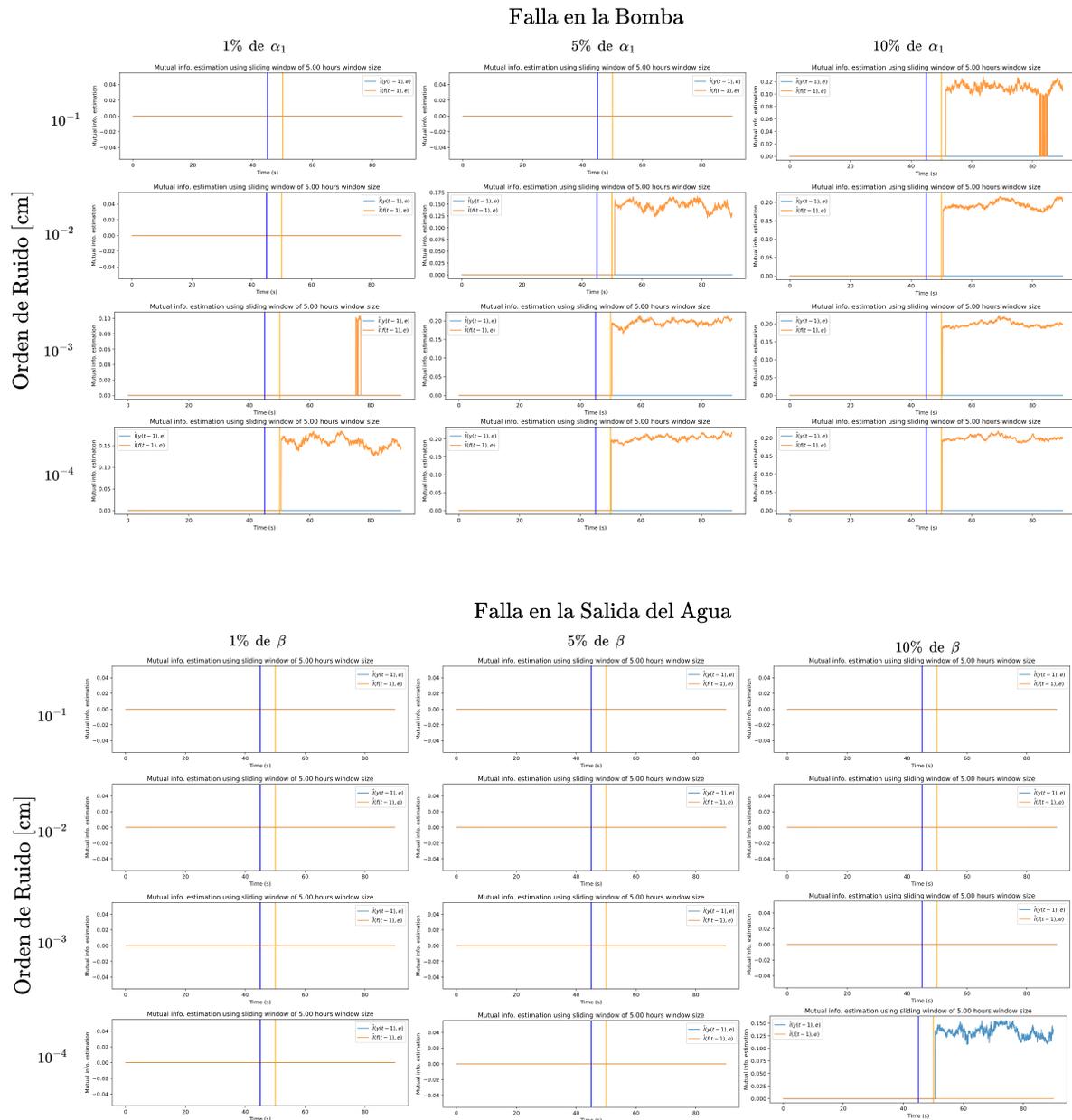


Figura 4.4: Resultados para un valor de lambda un orden de magnitud mayor que en la Figura 4.2. Notar que el umbral mínimo de información mutua es mucho más agresivo que antes, lo que genera una incapacidad de detección de obsolescencia en muchos casos.

#### 4.1.4.4. Ventana de 1440 muestras (10 horas).

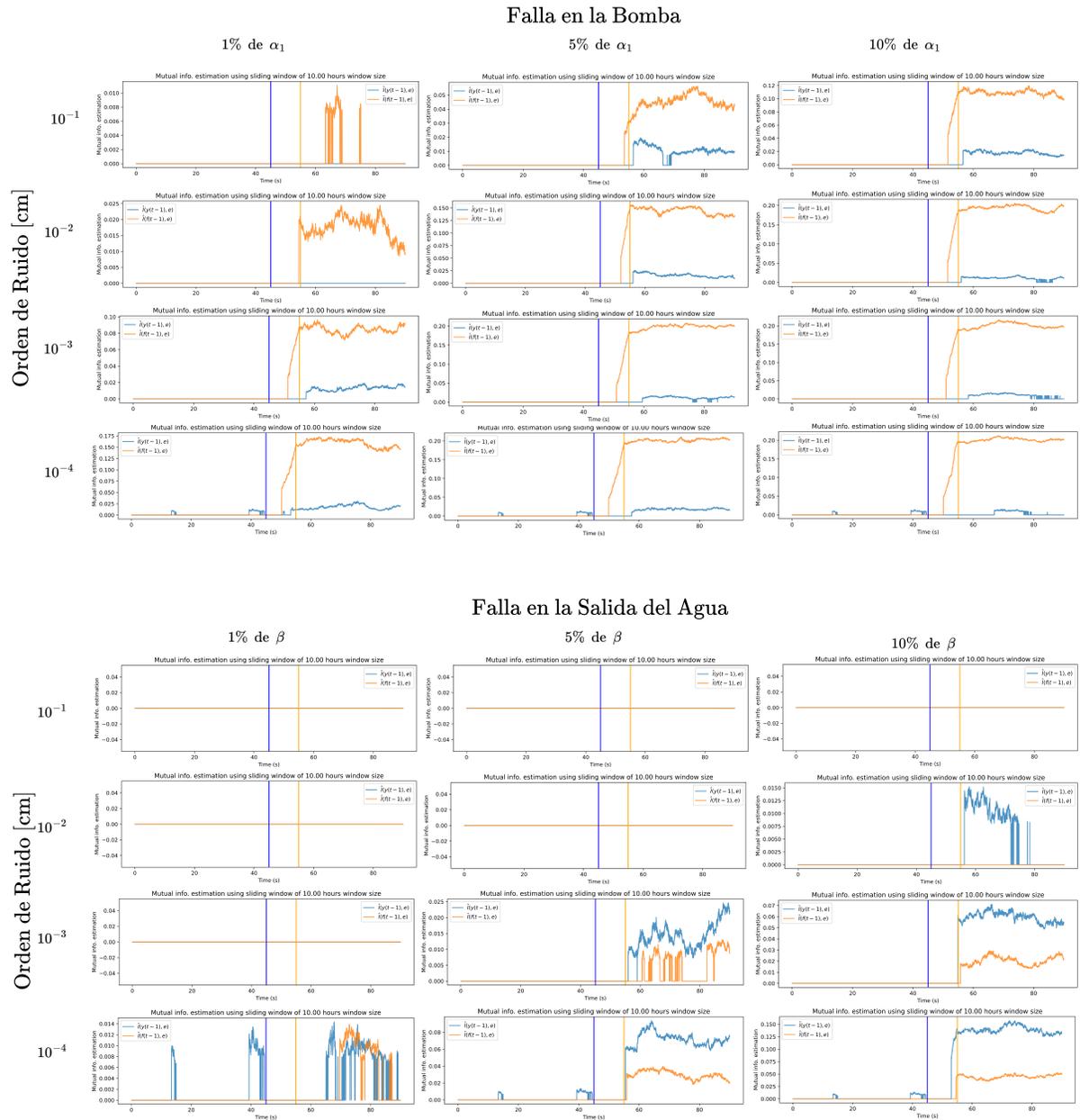


Figura 4.5: Resultados para una ventana deslizante de 1440 muestras. Se pueden observar resultados menos ruidosos, pero con similar proporción de detección efectiva. Se puede ver también que se retarda la detección dado que la ventana de análisis es mayor.

#### 4.1.4.5. Ventana de 2160 muestras (15 horas).

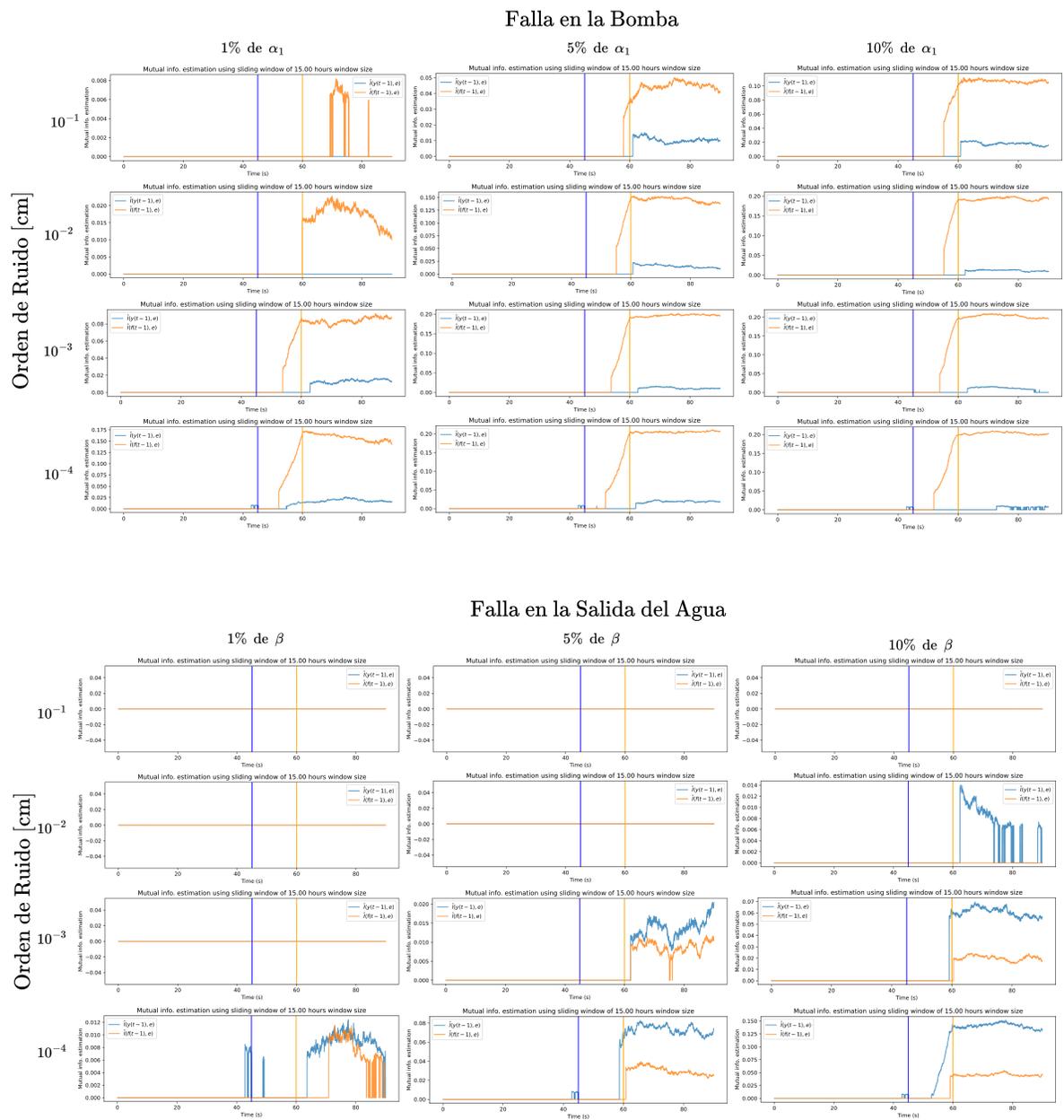


Figura 4.6: Resultados para una ventana deslizante de 2160 muestras. Se puede observar una estimación de la información mutua más suave, segura y constante en el tiempo, bajo el costo de una detección hasta dos veces más tardía que en los experimentos con 720 muestras.

#### 4.1.5. Análisis y comentarios para los próximos experimentos

Previo al análisis de las figuras, resultan notables los resultados del MSE de los modelos en la Tabla 4.1, en ella puede verse lo que se anticipaba teóricamente, esto es, que a mayor nivel de ruido hay un peor ajuste de la red. Aún así, se logran modelos suficientemente válidos acorde al criterio definido del 5% del *naive* MSE. Es esperable que la situación anterior se complique en los próximos experimentos debido al mejor desempeño del modelo ingenuo.

Como puede verse en las figuras, el test ha demostrado efectividad para la mayoría de los escenarios impuestos, siendo fallido en los casos de baja gravedad de la falla, alto nivel de ruido y cuando la falla está en la salida del agua (tal como se anticipaba en el análisis teórico mostrado en la sección 2.8.2, que sugiere una menor sensibilidad del residuo respecto al valor de  $\delta_2$ ). Se puede ver que la técnica de filtrado logra reducir las alzas particulares en la información mutua estimada, lo que reduce los falsos positivos, pero a la vez invalida los casos en los que esta alza de información mutua estimada es correspondiente al tramo de operación en falla. Esto es adecuado debido a que, pese a tener detecciones particulares, no resulta suficientemente verosímil que estas sean producto de un correcto funcionamiento del test, por consiguiente, la detección es invalidada por el filtro.

Se puede ver que el valor de **lambda** de  $7.5 \cdot 10^{-5}$  es (de los tres con los que se experimentó) el que permite una mejor relación de detección versus falsos positivos, debido a que logra establecer un umbral de información mutua estimada que filtra gran parte de las estimaciones mayores a 0 en la zona de operación normal; esto no implica que sea el valor óptimo, puesto que el algoritmo soporta un rango continuo, por lo que encontrar el valor óptimo requeriría de técnicas de búsqueda más costosas computacionalmente.

Respecto al tamaño de la ventana deslizante, se puede ver que resulta útil para tener una mayor certeza del valor de la información mutua entre las entradas, bajo el costo de aumentar el período en que no hay detección pese a existir una falla. También logró una reducción de los falsos positivos (ver gráficos de falla en la salida del agua, para falla leve con orden de ruido de  $10^{-4}$ , en las figuras 4.2, 4.5 y 4.6, en las cuales hay una paulatina desaparición de los falsos positivos conforme aumenta el tamaño de la ventana). Aumentar el tamaño de la ventana no demostró un beneficio sustancial en el aumento de detección, pero dado a que se alcanza una estimación más suave y sostenida, permite abordar el problema de aislamiento de la falla, esto pues, para fallas en la bomba se eleva la MI del residuo  $c/r$  a  $f$  de mayor forma que la MI con respecto a  $y$ , mientras que se da el caso contrario en las fallas en la salida del agua.

Debido a lo anterior, se tendrá en consideración para los próximos experimentos, un valor fijo de **lambda** =  $7.5 \cdot 10^{-5}$ ; y un valor variable del tamaño de la ventana deslizante correspondientes a 720, 1440 y 2160 muestras

## 4.2. Muestreo cada 10 segundos

En esta sección se muestran los resultados obtenidos para un tiempo de muestreo de 10 segundos. Al igual que en la sección anterior, se muestran en la Tabla 4.2. Puede verse que en esta ocasión el modelo correspondiente a un nivel de ruido de  $10^{-1}$  no fue capaz de cumplir con el criterio establecido de modelo válido, por lo que se tendrá bajo especial atención el desempeño del test en este nivel de ruido.

Tabla 4.2: Valores del MSE acorde al nivel de ruido para los modelos con tiempo de muestreo de 10 segundos. Valor calculado en el set de prueba respectivo. Se muestra en rojo el valor que no logró el MSE objetivo.

Orden del Ruido	<i>Naive</i> MSE	Test MSE
$10^{-1}$	2.51	0.19
$10^{-2}$	3.36	0.023
$10^{-3}$	2.35	0.006
$10^{-4}$	2.33	0.004

En las figuras 4.7, 4.8 y 4.9 se pueden ver los experimentos realizados con una ventana deslizante de 720, 1440 y 2160 muestras respectivamente, equivalentes a un tiempo de 2, 4 y 6 horas.

Para fallas en la bomba, se logra apreciar para todo tiempo de muestreo un alza significativa de la información mutua estimada entre el residuo y el porcentaje de utilización de la bomba posterior a la falla en la mayoría de los casos, no así con la MI entre el residuo y el nivel de agua, el cual se mantiene en cero para los casos de ruido alto y oscila pseudo-aleatoriamente en los casos de ruido muy bajo. Se puede además ver un incremento en las falsas detecciones cuando se tiene un muy bajo nivel de ruido.

Para fallas en la salida del agua, se puede ver que la detección de la falla leve no se cumple de manera satisfactoria en ningún experimento para ninguna ventana deslizante; las fallas medias y graves son detectadas con dificultad para el caso de la ventana de 720 muestras, ya que las señales resultantes son ruidosas. Este problema se resuelve ante el aumento del tamaño de la ventana, ya que puede verse que para 1440 y 2160 muestras un alcance estable de la EMI posterior a la falla. Cabe destacar que en **todos** los casos de detección exitosa, el valor de la EMI aumenta significativamente a medida que se reduce el ruido y/o a medida que sube la gravedad de la falla.

# Ventana de 720 muestras (2 horas)

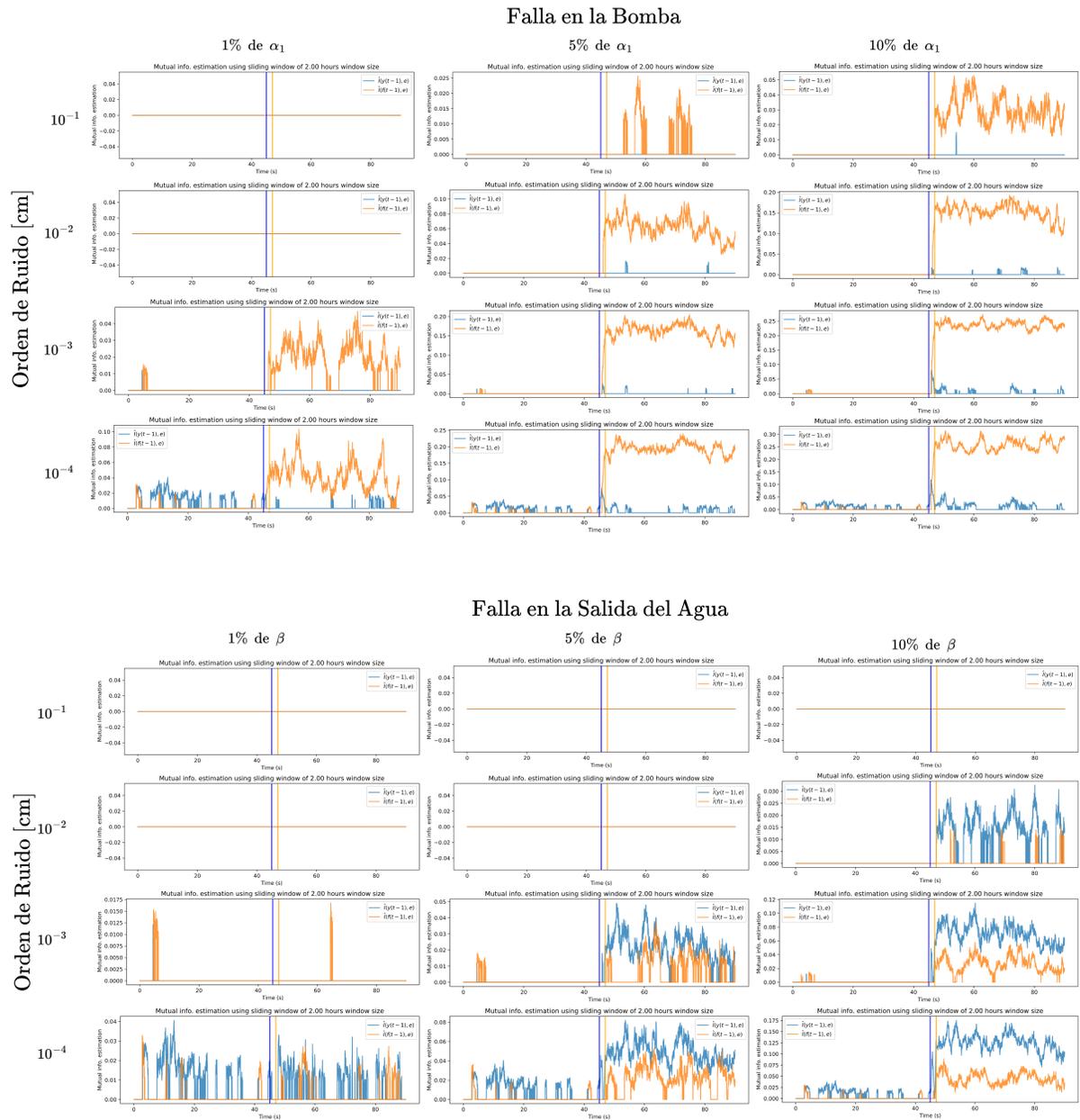


Figura 4.7: Resultados para un tiempo de muestreo de 10 (s), con una ventana de 720 muestras

# Ventana de 1440 muestras (4 horas)

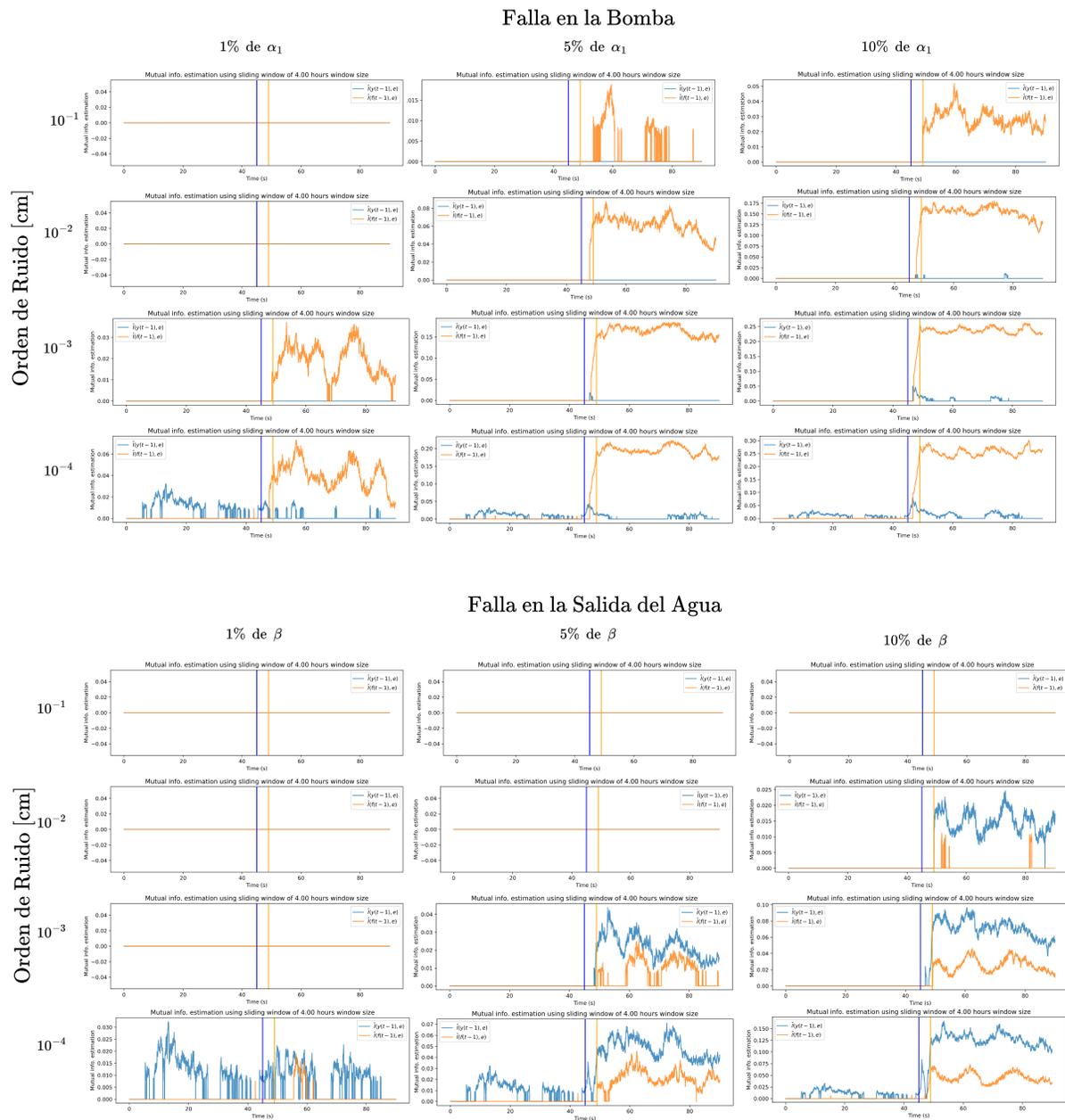


Figura 4.8: Resultados para un tiempo de muestreo de 10 (s), con una ventana de 1440 muestras

# Ventana de 2160 muestras (6 horas)

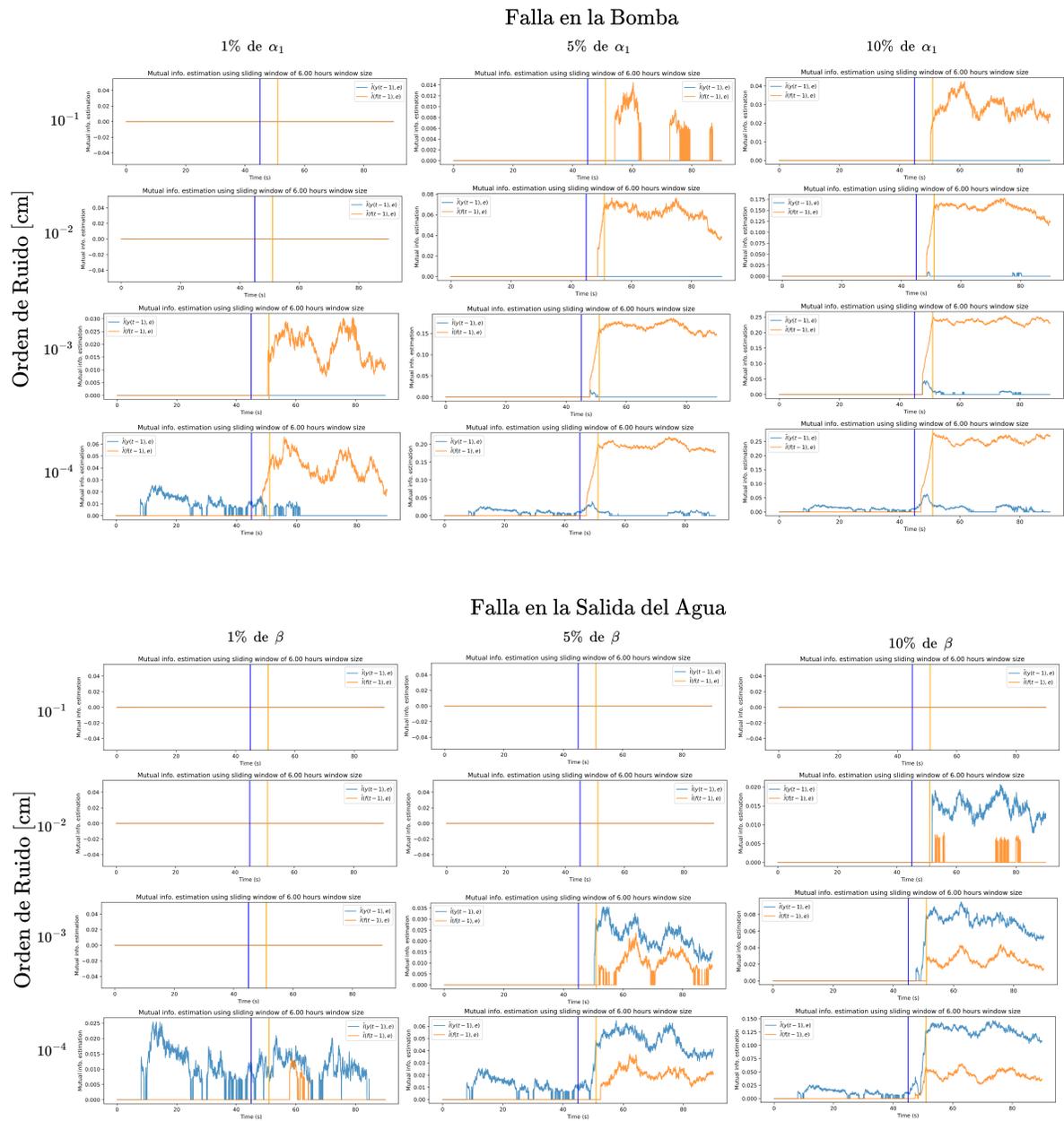


Figura 4.9: Resultados para un tiempo de muestreo de 10 (s), con una ventana de 2160 muestras

### 4.3. Muestreo cada 5 segundos

Finalmente, se muestran los resultados para un tiempo de muestreo de 5 horas. Tal como se preveía de la sección del marco teórico, se puede ver cómo el *naive* MSE es sustancialmente menor que en los experimentos anteriores, mientras que el MSE de test para cada modelo no disminuye con la misma fuerza.

Tabla 4.3: Valores del MSE acorde al nivel de ruido para los modelos con tiempo de muestreo de 5 segundos. Valor calculado en el set de prueba respectivo. Se muestran en rojo los valores del MSE que no lograron cumplir con el criterio establecido de mínimo MSE.

Orden del Ruido	<i>Naive</i> MSE	Test MSE
$10^{-1}$	0.55	0.19
$10^{-2}$	0.36	0.021
$10^{-3}$	0.34	0.0023
$10^{-4}$	0.34	0.00053

Los resultados se muestran en las figuras 4.10, 4.11 y 4.12; correspondientes a realizaciones del experimento con ventana deslizante de 720, 1440 y 2160 muestras respectivamente, equivalentes en tiempo a ventanas de 1, 2 y 3 horas.

En la Figura 4.10 se logran apreciar resultados sumamente ruidosos, con alzas de la EMI erráticas, prácticamente nula detección en el caso de falla en la salida del agua, y gran oscilación en el caso de falla en la bomba. Estas complicaciones se ven mejoradas conforme aumenta el tamaño de la ventana, pues en las figuras 4.11 y 4.12 el comportamiento ruidoso de la Figura anterior se ve asentado, haciendo evidente la detección. Notar que al igual que en los experimentos con tiempo de muestreo de 10 segundos, en el caso de menos ruido se eleva levemente la EMI del residuo *c/r* al nivel de agua en momentos previos a la falla.

# Ventana de 720 muestras (1 hora)

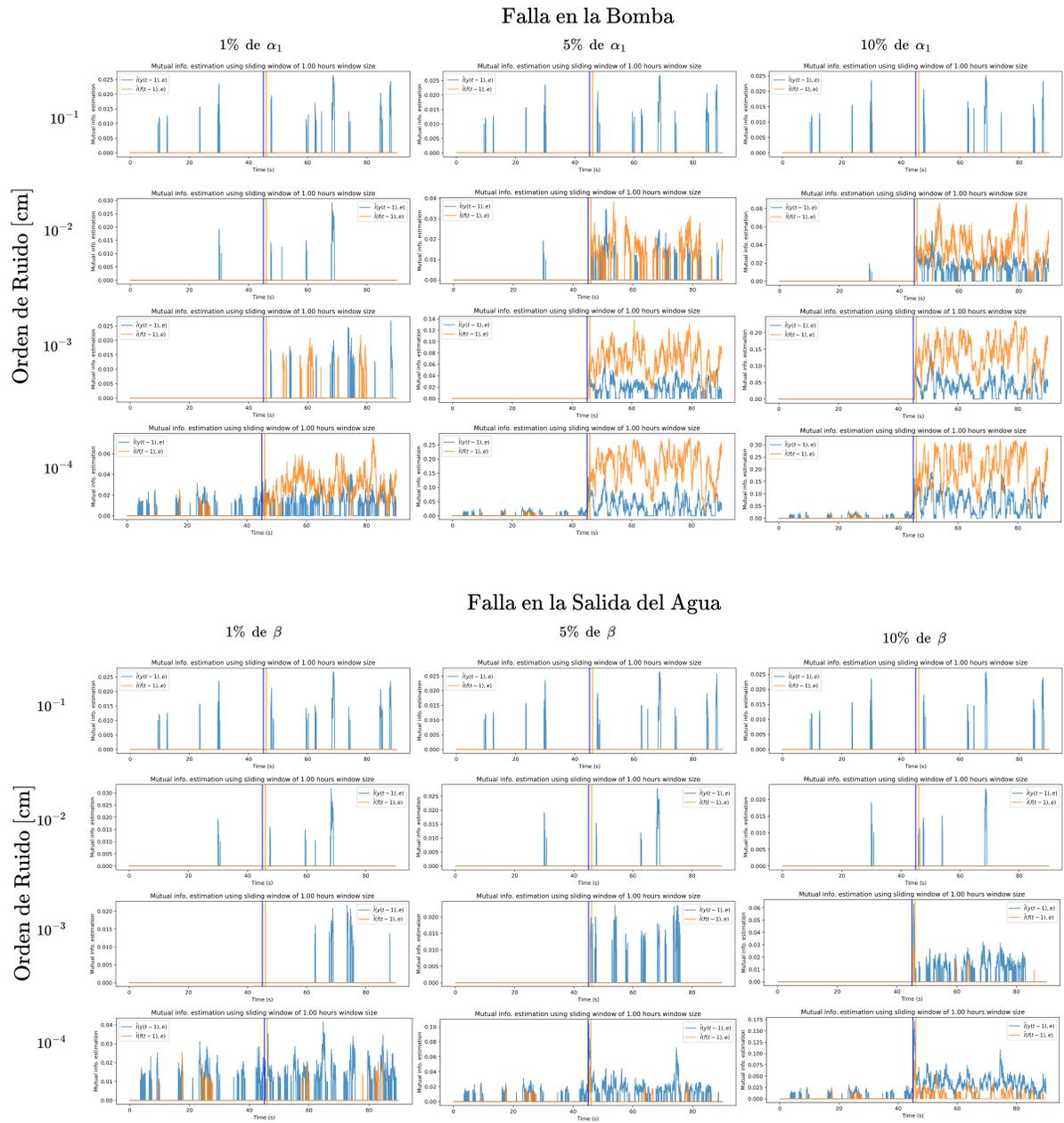


Figura 4.10: Resultados para un tiempo de muestreo de 5 (s), con una ventana de 720 muestras

# Ventana de 1440 muestras (2 horas)

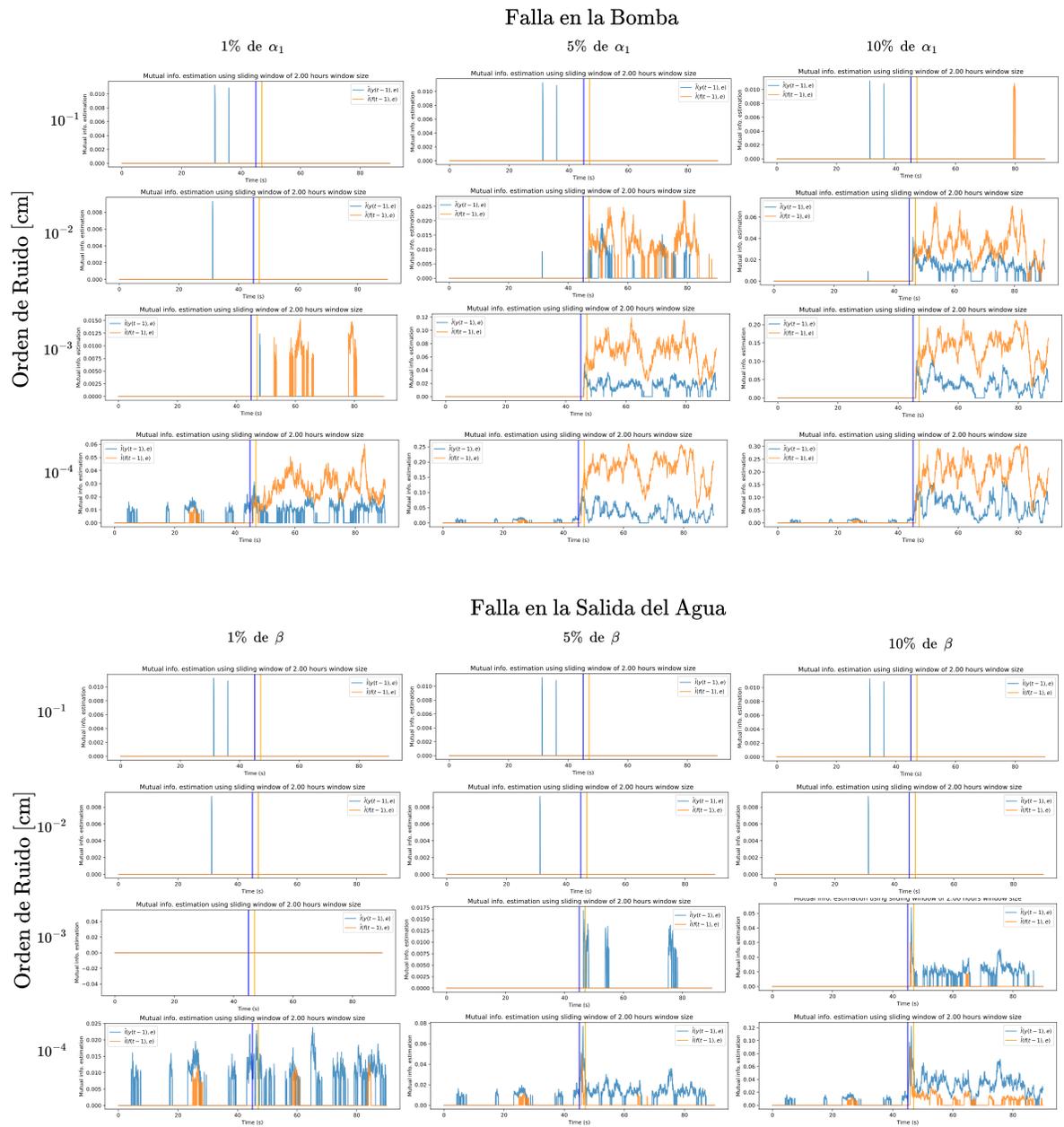


Figura 4.11: Resultados para un tiempo de muestreo de 5 (s), con una ventana de 1440 muestras

# Ventana de 2160 muestras (3 horas)

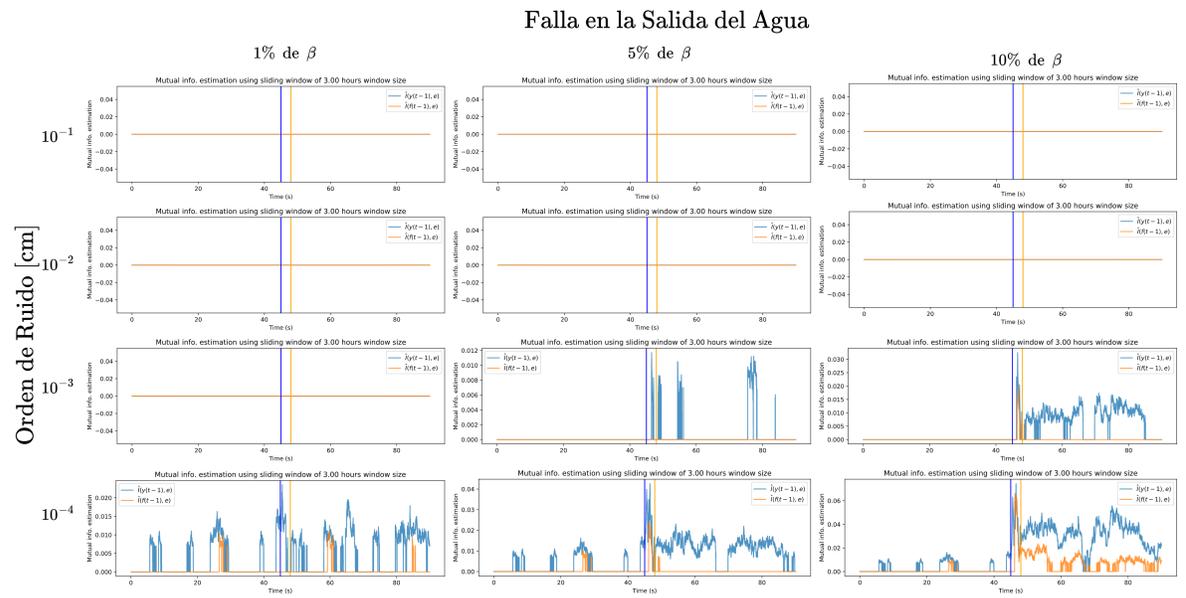
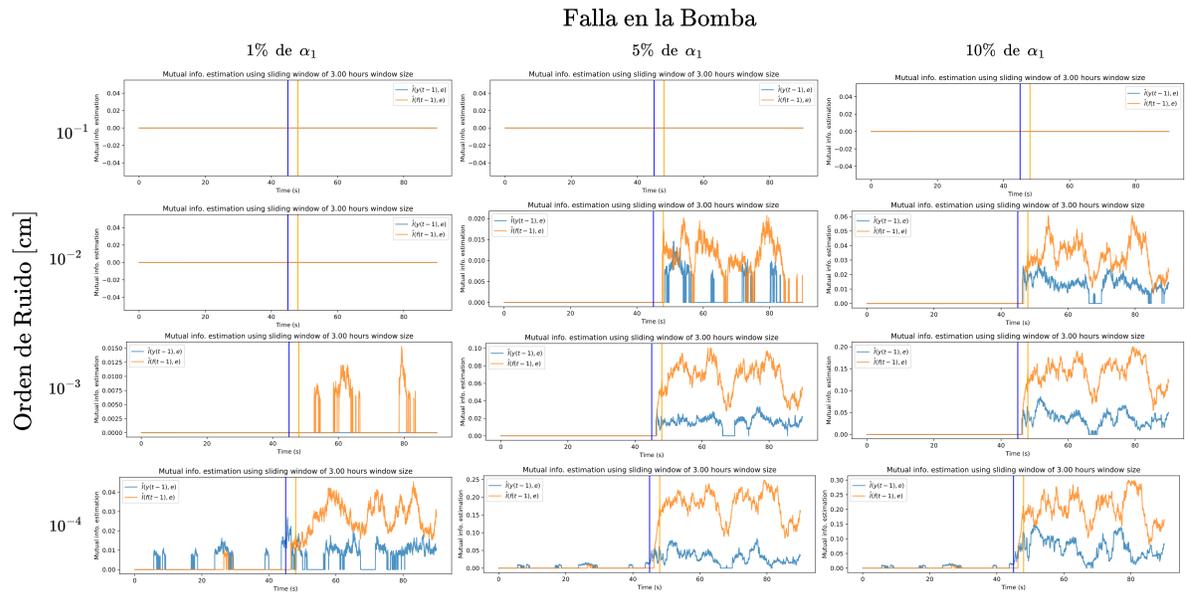


Figura 4.12: Resultados para un tiempo de muestreo de 5 (s), con una ventana de 1440 muestras

# Capítulo 5

## Discusión de Resultados

En este capítulo se realiza la discusión de los resultados, comentando los comportamientos de las variables en los experimentos realizados, y realizando una comparación con los resultados esperados en base a la teoría, para finalmente dar paso a las conclusiones acerca del cumplimiento de los objetivos y la prueba de la hipótesis principal.

### 5.1. Funcionamiento del Test

#### 5.1.1. Detección

Al observar los resultados, resulta inmediato que, al menos para niveles de ruido de medios a bajos (i.e. menores o iguales a  $10^{-2}$  [cm]) y para fallas medias y graves, el alza de información mutua inmediatamente posterior al comienzo de la falla es sistemática; lo que da un sustento sólido para inferir que el alza de la EMI es resultado directo de la obsolescencia del modelo. Dado que los resultados expuestos fueron simulados con la misma semilla de generación de ruido para que fueran comparables, es que resulta posible (aunque poco verosímil) una relación con el ruido y el alza de la EMI posterior a la falla, sospecha que es rápidamente rechazada al cambiar la semilla generación de ruido y observar comportamientos similares del test (ver resultados extra en los anexos).

#### 5.1.2. Sensibilidad a los tipos de falla simulados

Al comparar los resultados obtenidos con el análisis teórico, se puede verificar que la sensibilidad del test respecto a la constante de perturbación  $\delta_1$  es efectivamente mayor que respecto a  $\delta_2$ , haciendo más fácil la detección de falla en la bomba que fallas en la salida del agua. El motivo de este comportamiento en el análisis teórico es sencillo de explicar, y es debido a que una unidad de discrepancia del parámetro  $\alpha_1$  afecta el flujo del agua mucho más que una unidad de discrepancia de  $\beta$ . Por ejemplo, para un porcentaje de utilización  $f = 30\%$ , se tiene un flujo de entrada teórico ( $\delta_1 = 0$ ) de  $84.67 \left[\frac{cm^3}{s}\right]$ , valor que desciende a  $54.67 \left[\frac{cm^3}{s}\right]$  si se impone  $\delta_1 = -1$ . Por otro lado, para una altura de 20 [cm] de agua en el estanque, se tiene un flujo de salida teórico de  $90.4 \left[\frac{cm^3}{s}\right]$ , valor que desciende a  $85.9 \left[\frac{cm^3}{s}\right]$  si  $\delta_2 = -1$ , variación aproximadamente seis veces menor. Un ejemplo completo de lo anterior puede verse en la Figura 5.1, en la cual resulta evidente la gran diferencia de flujos causada por un mismo valor de  $\delta$  en las ecuaciones de flujo de entrada y flujo de salida. Por

consiguiente, dado que los resultados obtenidos en el análisis teórico comparan variaciones equivalentes de  $\delta_1$  y  $\delta_2$ , es esperable una detección más temprana en la falla en la bomba. Sin embargo, lo anterior fue considerado en los experimentos realizados, utilizando variaciones inherentes al valor nominal de las constantes por sobre valores equivalentes con el objetivo de una comparación más justa. Dado lo anterior, existe la posibilidad de que los motivos que explican la diferencia de sensibilidad del test en los distintos tipos de falla del análisis teórico no sean los mismos que los de los resultados experimentales.

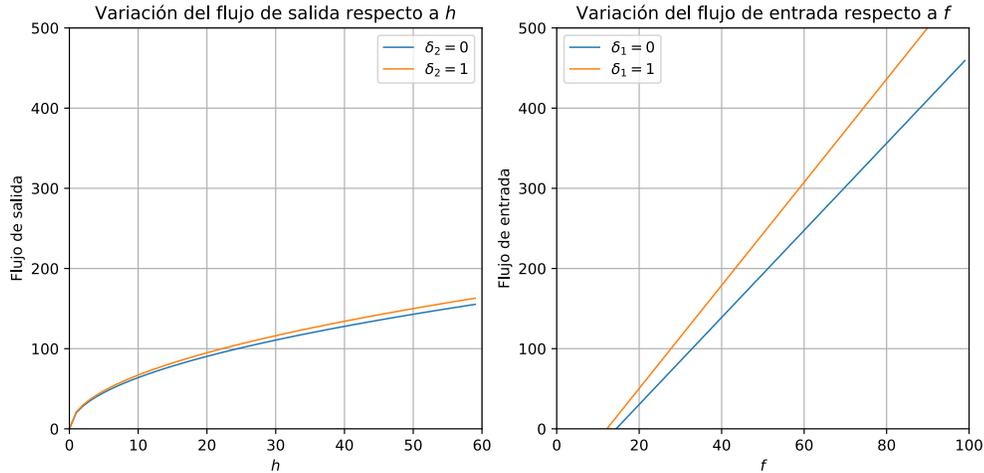


Figura 5.1: Comparación de afección de las constantes de perturbación equivalentes en los flujos de entrada y salida.

De lo anterior, se desprenden dos hipótesis factibles que explicarían esta conducta en los resultados experimentales:

- Que la gravedad de las fallas simuladas no es comparable y por lo tanto no es adecuado usar el mismo porcentaje de variación respecto del valor nominal de  $\alpha_1$  y  $\beta$  como fue usado en estos experimentos.
- Que dado que el flujo de entrada depende linealmente de  $f$ , mientras que el flujo de salida depende de la raíz cuadrada de  $h$ , se generaría una mayor tolerancia en las discrepancias de  $\beta$  por sobre las de  $\alpha_1$ .

Las hipótesis anteriores pueden probarse abordando el problema mediante un análisis gráfico del mismo tipo que el de la Figura 5.1, pero esta vez utilizando los valores de  $\delta_1$  y  $\delta_2$  relativos a la constante que están afectando. El análisis puede verse en la Figura 5.2, en ella, se evidencia que el problema de las discrepancias para las distintas fallas está directamente relacionado con el lento crecimiento de la función del flujo de salida dependiente de la altura del agua en comparación con el crecimiento lineal de la función de flujo de entrada respecto a  $f$ . Esto en conjunto a los rangos de operación de  $h$ , generan que la falla en el flujo de salida tenga mayor tolerancia al valor de  $\delta_2$  referido a la gravedad de la falla, haciendo que la falla leve definida en esta memoria por  $\delta_2 = 1\% \beta$  sea posiblemente demasiado leve para la sensibilidad del test, imposibilitando su satisfactoria detección. Este análisis justifica parcialmente las dos hipótesis planteadas, pero queda pendiente un análisis experimental con diferentes

variaciones en los valores de  $\delta_1$  y  $\delta_2$  que muestren gravedades de falla similares, esperando resultados de detección similares.

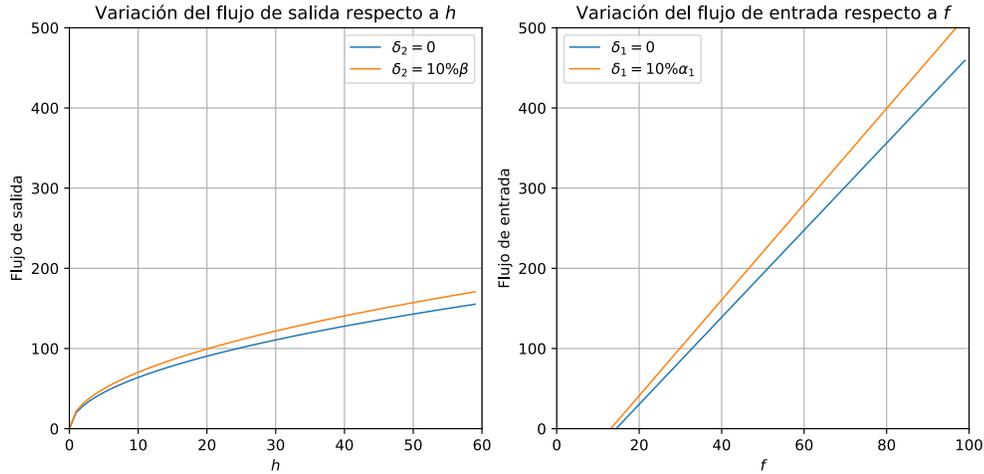


Figura 5.2: Comparación de afección de las constantes de perturbación, relativas a la constante que afecta cada una, en los flujos de entrada y salida.

### 5.1.3. Aislamiento de falla detectada

Al analizar cada uno de los conjuntos de experimentos para los distintos tiempos de muestreo y distintos tamaños de ventana, es posible notar que el alza de la EMI respecto a  $f$  por sobre  $y$  está siempre relacionada con una falla en la bomba, mientras que el caso opuesto está relacionado (aunque de menor forma) con una falla en la salida del agua. De esta manera, además de existir detección de falla, se pueden tomar los resultados del test como evidencia para inferir el origen de la falla (aislamiento). La expresión teórica del residuo indicaba que, ante fallas en la bomba existiría dependencia estadística del residuo con respecto a  $f$  y  $h$ , mientras que una falla en la salida del agua no tendría dependencia con  $f$ . Los resultados obtenidos muestran que, en general, en ambos tipos de falla existirá dependencia estadística del residuo con respecto a ambas variables, pero en una **proporción** sustancialmente distinta dependiendo del tipo de falla. Esto puede explicarse debido a que la expresión del residuo mostrada asume un modelo perfectamente ajustado al fenómeno, lo que no es posible de generar a nivel práctico y ni siquiera es posible acercarse con una red MLP simple y recursos limitados, lo que trae como consecuencia estas diferencias con la teoría.

## 5.2. Influencia del Ruido

Respecto al rol del ruido en la experimentación, se puede observar que en este caso sí se cumple lo esperado del análisis teórico. El nivel de ruido más alto complica severamente el desempeño del test debido a que su amplitud supera al menos en un orden de magnitud a la componente del residuo que teóricamente tendría dependencia estadística con las entradas, por lo que en muchos casos de ruido alto esta dependencia fue imposible de detectar.

Para niveles de ruido medios, el test se comporta relativamente bien, pudiendo detectar satisfactoriamente las fallas de gravedad media y alta para el caso de falla en la bomba, y fallas de gravedad alta en el caso de falla en el flujo de salida del agua.

Para niveles de ruido teóricos (i.e. ruido muy bajo), se comienzan a estimar leves valores información mutua en sectores que debiese ser 0. Dado que esto ocurre en varios experimentos de manera sistemática, es válido inferir que el nivel de ruido bajo tiene participación en esto. En efecto, ante un nivel tan bajo de ruido, cualquier discrepancia del modelo con el fenómeno real podría generar dependencia del residuo con alguna entrada, por lo que se requeriría de un modelo mejor ajustado para solventar estos efectos (o bien aumentar el valor de  $\lambda$  para aumentar el umbral de EMI mínimo, sacrificando sensibilidad de detección).

Por consiguiente, para un problema real de este estilo, la indicación sería comprar un sensor con precisión de al menos  $\pm 10^{-2}$  [cm], en conjunto con un modelo lo mejor ajustado posible, con fin de facilitar la detección de obsolescencia.

### 5.3. Influencia del Tiempo de Muestreo

Los resultados obtenidos dan evidencia de una disminución de precisión (resultados más ruidosos) conforme disminuye el tiempo de muestreo. Esto puede explicarse mediante la complicación de la obtención de un modelo válido según los criterios definidos en base al *naive* MSE, con lo que cada vez se vuelve más difícil obtener un modelo suficientemente correcto. Por otro lado, es necesario destacar que los cambios de referencia se mantienen constantes cada 400 [s], lo que a menor tiempo de muestreo, con una ventana del mismo número de muestras que en experimentos de tiempos mayores, genera una ventana de tiempo menor, con lo que las componentes no estacionarias del proceso durante las transiciones de referencia tomarán mayor presencia, muchas veces estas componentes no estacionarias vienen acompañadas de valores de  $f$  casi constantes (e.g. bajar de un nivel de 50 [cm] a 20[cm] implica tener la bomba en porcentajes bajos), que generarían variaciones importantes en la EMI, por lo que para obtener resultados menos ruidosos será necesario utilizar ventanas más grandes. Lo anterior trae consigo otro problema, y es que al existir un mayor número de muestras por unidad de tiempo, el costo computacional de realizar la estimación de información mutua crece, lo que implica un mayor tiempo de ejecución y por lo tanto un mayor retardo en la realización del test.

En consecuencia, para problemas de este tipo, en el cual se deba ajustar un modelo en base a datos temporales, existen limitaciones respecto al costo que implica obtener un modelo de calidad suficiente, disminuyendo la efectividad del test por transitividad. Por otro lado, la cantidad de componentes correspondientes a zonas no estacionarias del proceso (o que generen poca varianza de las variables de entrada) aumentarán su presencia, disminuyendo la efectividad del test también.

### 5.4. Influencia del Lazo de Control del sistema

Una gran diferencia del análisis teórico respecto de los experimentos realizados, es la presencia de un control de nivel a lazo cerrado, con cambio de referencias dentro del rango de operación. Esto genera que exista una dependencia directa entre el valor de  $f_t$  con el

valor de  $y_{t-1}$  y, por lo tanto, con  $h_{t-1}$  y  $w_{t-1}$ . Como  $h_{t-1}$  depende directamente de  $f_{t-2}$  y  $h_{t-2}$ , es que se da una propagación hacia atrás de las dependencias; razón por la cual estimar la información mutua del residuo respecto a autorregresores que no son partícipes del modelo como tal, generen una estimación de información mutua similar (ver sección de anexos, la sección de resultados para más autorregresores). De esta forma, el lazo de control genera dependencias no consideradas en un análisis preliminar teórico, que pueden afectar los resultados esperados.

# Capítulo 6

## Conclusiones y Trabajo Futuro

### 6.1. Conclusiones Principales

Una vez concretada la experimentación con un volumen suficiente de resultados notables, se concluye que esta técnica de estimación de información mutua es adecuada para la detección de obsolescencia de modelos, y particularmente útil para la detección y diagnóstico de fallas si se utiliza un modelo suficientemente válido como supervisor en tiempo real del proceso. Para un funcionamiento óptimo del test, es necesario, en primer lugar, disponer de sensores que permitan un muestreo de datos dentro de un error lo más acotado posible; en segundo lugar, disponer de un modelo que sea capaz de captar la dinámica del proceso y, de esta manera, ajustarse lo mejor posible al fenómeno físico detrás del sistema, más que a los datos mismos; y finalmente, definir un parámetro  $\lambda$ , una ventana deslizante y un criterio de filtrado para la realización del test.

Los resultados fueron consistentes con lo esperado tras el análisis teórico, con distintas discrepancias debidas principalmente a los cambios de referencia en el nivel de agua y la presencia de un lazo de control. Esto último genera que exista dependencia entre  $h(t-1)$  y  $f(t)$ , por lo que limita el uso de autorregresores en el análisis y causa que el aislamiento de la falla no resulte tan directo como lo indica la teoría, con lo que existen diversos escenarios en los cuales no será posible aislar la falla de manera segura mediante este método, particularmente los casos de falla leve o mediana con altos niveles de ruido, en las cuales el alza de la EMI es ruidosa y pequeña, y no logra delatar con suficiente veracidad al factor causante de ello.

Las principales ventajas del test, son la seguridad que este tiene para el no rechazo de la hipótesis nula en operación normal del sistema, por lo que tiene una baja tasa de falsos positivos en los casos que el test funciona, y que puede reducirse prácticamente a cero sin perjudicar mayormente la detección de obsolescencia en los casos de fallas medias y graves (en efecto, para todos los experimentos de bajo ruido en las fallas medias y graves, basta con subir el umbral de la EMI mínima de aceptación para filtrar absolutamente todos los falsos positivos sin afectar la detección). Por otro lado, tal como se menciona en la sección anterior, es cualitativamente visible un patrón constante en el comportamiento de las EMIs en cada una de las fallas programadas, lo que hace de este test una alternativa factible para el aislamiento de una falla media o grave y de esta forma sugiere dónde estaría el origen de la falla, una tarea que en general no resulta trivial, y que por lo tanto aporta un gran valor a este test.

El uso del test no solo se limita a la detección y diagnóstico de fallas, sino que también es útil para evaluar la validez del modelo una vez ajustado. Al realizar la estimación de información mutua de la misma forma para un conjunto de prueba, un modelo válido sería aquel que no tenga incrementos sustanciales en la información mutua estimada entre el residuo y las entradas.

## 6.2. Trabajo Futuro

La principal oportunidad de mejora de este método tiene relación con la sensibilidad al ruido que el test posee. Tal como puede verse en los resultados, las fallas incluso graves son prácticamente indetectables cuando el ruido es alto. Para abordar este problema sería necesario incorporar metodologías de filtrado pasa bajo para reducir la presencia del ruido. Otra no conformidad es el aumento del costo computacional para tiempos de muestreo cada vez menores. Sin embargo, esto no es un asunto del del test en sí, sino que tiene relación con la constante de tiempo dominante del proceso (dependiente del sistema físico en sí, en conjunto al lazo de control, incluyendo los parámetros PID), que complejizan el ajuste de modelos cuando el tiempo de muestreo es mucho menor a esta, y a la vez hace que haya mayor densidad de datos respecto a una cierta ventana de tiempo, ralentizando el desarrollo de la estimación de información mutua si se utilizan ventanas de relativamente gran tamaño. En este sentido, el principal desafío consiste en compatibilizar un control funcional del sistema, que a la vez permita el ajuste de modelos causales representativos del proceso mediante la consideración de los tiempos de toma de muestras y actuación. De lo anterior se observa la necesidad de poseer un control a tiempo discreto, ya que en tiempo continuo no se cumplirían las hipótesis del problema, en el sentido que la variable de nivel de agua será dependiente del porcentaje de utilización de la bomba para el mismo instante de tiempo, existiendo cambios de estas variables decididos por el controlador entre un instante finito de tiempo y otro, lo que dificulta el ajuste de un modelo causal en base a datos.

Para este problema en particular, resultaría de interés realizar a futuro distintos experimentos con diferentes configuraciones de los parámetros PID, con el fin de evaluar el cómo afectan en la detección. Se puede prever que dado que las componentes I y D tienen una memoria en el tiempo, exista una afección de estos en un modelo que solo considera pocos autorregresores, por lo que se podrían obtener modelos más precisos con redes que consideren estas componentes (por ejemplo, una red LSTM que considera el pasado de corto plazo y largo plazo). Por otro lado, sería relevante llevar la misma metodología utilizada en este trabajo a una planta física real, de la cual se dispongan datos para el ajuste, y se pueda implementar el test en tiempo real con fallas controladas, lo que finalizaría la demostración de la utilidad real de el test implementado para este sistema.

Posterior a los resultados obtenidos en este sistema, es necesario implementar el test en otro sistema con aún mayor nivel de complejidad, con sistemas de varias variables y un múltiple control de éstas. En esta línea, actualmente (12/2022) T. Rojas, integrante del IDS, está trabajando en la implementación del test en el *Tennessee Eastman Process*, un proceso químico industrial complejo de varias variables, obteniendo resultados prometedores en la detección y diagnóstico de fallas del proceso.

# Bibliografía

- [1] Gonzalez, M., Silva, J., Videla, M., y Orchard, M., “Data-driven representations for testing independence: Modeling, analysis and connection with mutual information estimation,” *IEEE Transactions on Signal Processing*, vol. 70, pp. 158–173, 2022, doi:[10.1109/tsp.2021.3135689](https://doi.org/10.1109/tsp.2021.3135689).
- [2] Isermann, R., “Model-based fault-detection and diagnosis – status and applications,” *Annual Reviews in Control*, vol. 29, no. 1, pp. 71–85, 2005, doi:<https://doi.org/10.1016/j.arcontrol.2004.12.002>.
- [3] Khorasgani, H., Farahat, A., y Gupta, C., “Data-driven residual generation for early fault detection with limited data,” 2021, doi:[10.48550/ARXIV.2110.15385](https://doi.org/10.48550/ARXIV.2110.15385).
- [4] Ramírez, C., Rojas, T., Silva, J., y Orchard, M., “Development of a model obsolescence (mismodeling) detector using a data-driven mutual information estimator,” 2022.
- [5] Jáuregui, C., “Evaluación de estrategias de sintonización de controladores fraccionarios para planta no lineal: sistema de estanques.,” pp. 35–67, 2016, <https://repositorio.uchile.cl/handle/2250/140963>.
- [6] “¿qué es el control pid? - matlab & simulink.” <https://la.mathworks.com/discovery/pid-control.html>. (Consultado el 06/28/2022).
- [7] Adams, M., Sáez, D., Zúñiga, R., y Sippa, S., “Pid control - apuntes de clases el4004 - fundamentos de control de sistemas,” 2020. (Consultado el 06/28/2022).
- [8] Haykin, S., *Neural Networks: A Comprehensive Foundation*, vol. 2. 2004.
- [9] Moore, K. L., *Clinically oriented anatomy*. Lippincott Williams & Wilkins, 2006.
- [10] “Esquema de la neurona.” <https://esquema.net/neurona/>. (Consultado el 06/28/2022).
- [11] Brownlee, J., “Loss and loss functions for training deep learning neural networks - machinelearningmastery.com.” <https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/>, 2019. (Consultado el 12/04/2022).
- [12] Venkateswarlu, C. y Karri, R. R., “Chapter 5 - data-driven modeling techniques for state estimation,” en *Optimal State Estimation for Process Monitoring, Fault Diagnosis and Control* (Venkateswarlu, C. y Karri, R. R., eds.), pp. 91–111, Elsevier, 2022, doi:<https://doi.org/10.1016/B978-0-323-85878-6.00010-5>.
- [13] Beers, B., Potters, C., y Schmitt, K. R., “What is regression? definition, calculation, and example.” <https://www.investopedia.com/terms/r/regression.asp>, 2022. (Consultado el 12/04/2022).
- [14] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt,

- B., y Varoquaux, G., “Underfitting vs. overfitting — scikit-learn 1.1.3 documentation.” [https://scikit-learn.org/stable/auto\\_examples/model\\_selection/plot\\_underfitting\\_or\\_overfitting.html](https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_or_overfitting.html). (Consultado el 12/04/2022).
- [15] Kathuria, C., “Regression — why mean square error? | by chayan kathuria | towards data science.” <https://towardsdatascience.com/https-medium-com-chayankathuria-regression-why-mean-square-error-a8cad2a1c96f>, 2019. (Consultado el 12/04/2022).
- [16] Brownlee, J., “How to grid search naive methods for univariate time series forecasting - machinelearningmastery.com.” <https://machinelearningmastery.com/how-to-grid-search-naive-methods-for-univariate-time-series-forecasting/#:~:text=A%20naive%20forecast%20involves%20using,adjusted%20slightly%20for%20seasonal%20data.>, 2018. (Consultado el 12/04/2022).
- [17] Privitera, G., Statistics for the Behavioral Sciences. Part III: Probability and the foundations of inferential statistics. Chapter 8. St. Bonaventure University, 2016, [https://www.sagepub.com/sites/default/files/upm-binaries/40007\\_Chapter8.pdf](https://www.sagepub.com/sites/default/files/upm-binaries/40007_Chapter8.pdf).
- [18] Vu, M., “Lecture 1: Entropy and mutual information, tufts university, departament of electrical and computer engineering.” <http://www.ece.tufts.edu/ee/194NIT/lect01.pdf>. (Consultado el 28/06/2022).
- [19] Cerrada, M., Cardillo, J., y Prada, A., “Diagnóstico de fallas basado en modelos: Una solución factible para el desarrollo de aplicaciones scada en tiempo real,” Revista Ciencia e Ingeniería, vol. 32, pp. 163–172, 2011.
- [20] Suárez, M., Detección de Fallas en Sistemas de Control Automático. Aplicación a una Planta de Tratamiento de Efluentes. PhD thesis, Departamento de electrociencias y electrónica industrial. Facultad de ingeniería Universidad Nacional de La Plata, 2002.
- [21] Grenaille, S., Henry, D., y Zolghadri, A., “A method for designing fdi filters for polytopic lpv models,” en Fault Detection, Supervision and Safety of Technical Processes 2006 (Zhang, H.-Y., ed.), pp. 717–722, Oxford: Elsevier Science Ltd, 2007, doi:<https://doi.org/10.1016/B978-008044485-7/50121-4>.
- [22] Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., y Hjelm, D., “Mutual information neural estimation,” en Proceedings of the 35th International Conference on Machine Learning (Dy, J. y Krause, A., eds.), vol. 80 de Proceedings of Machine Learning Research, pp. 531–540, PMLR, 2018, <https://proceedings.mlr.press/v80/belghazi18a.html>.
- [23] Noshad, M., Zeng, Y., y Hero, A. O., “Scalable mutual information estimation using dependence graphs,” en ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2962–2966, 2019, doi:[10.1109/ICASSP.2019.8683351](https://doi.org/10.1109/ICASSP.2019.8683351).
- [24] Videla, M., “Tsp-mismodeling. mismodeling detection experiments through independence tests based on tree-structured partitions (tsp).” 2022. (Consultado el 28/06/2022).
- [25] Ramírez, C., “Detección de *mismodeling* en base a test de independencia definido según una estimación de información mutua *data-driven*. experimentos preliminares de ruido aditivo.” 2022.
- [26] Videla, M., “mvidela31/tsp-it: An independence test based on data-driven tree-structured representations.” <https://github.com/mvidela31/TSP-IT>, 2022. (Accessed

on 12/11/2022).

# Anexos

## Anexo A. Resultados para modelos de desempeño regular

En esta sección se muestran resultados individuales en las cuales no hubo éxito de detección de obsolescencia debido a un mal ajuste del modelo .

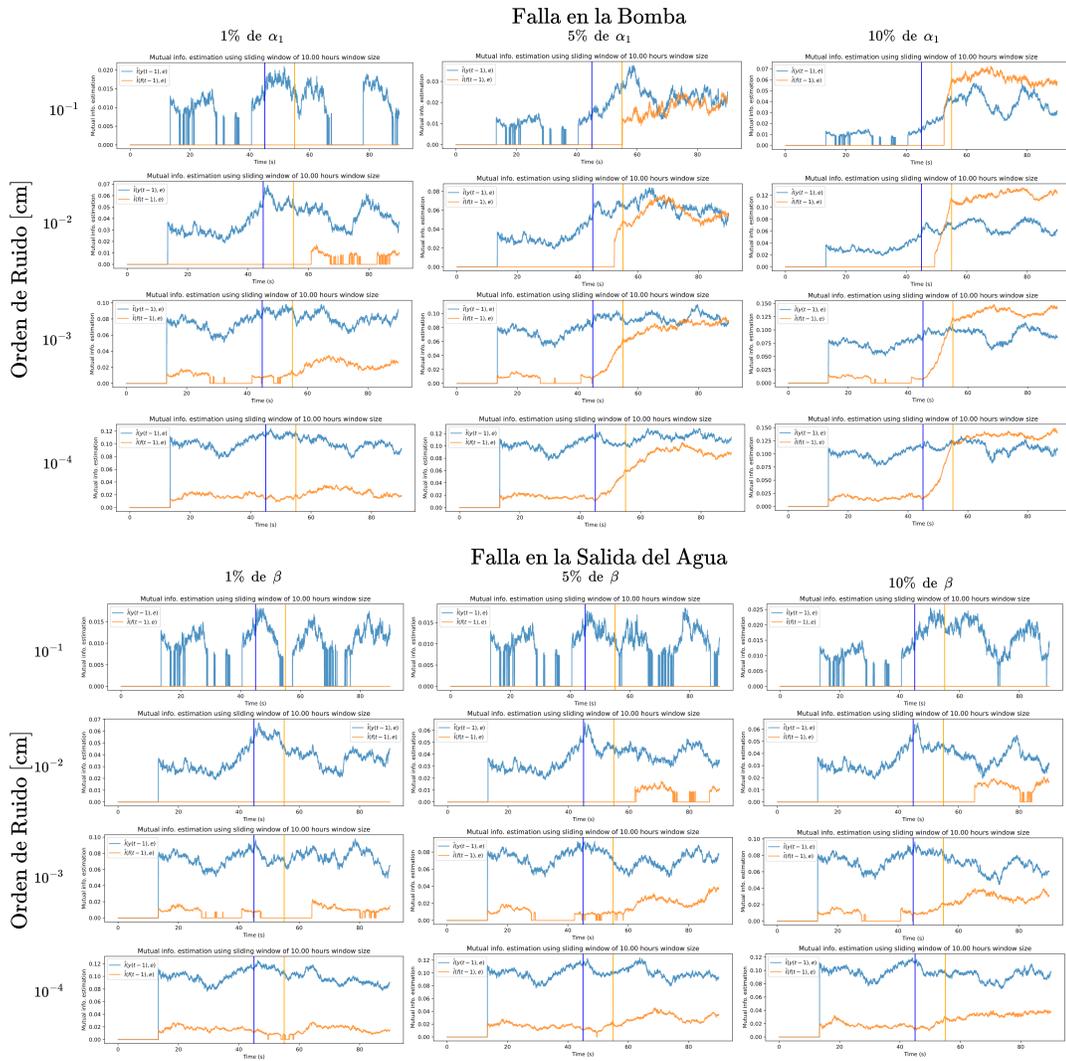


Figura A.1: Modelo ajustado por 300 épocas

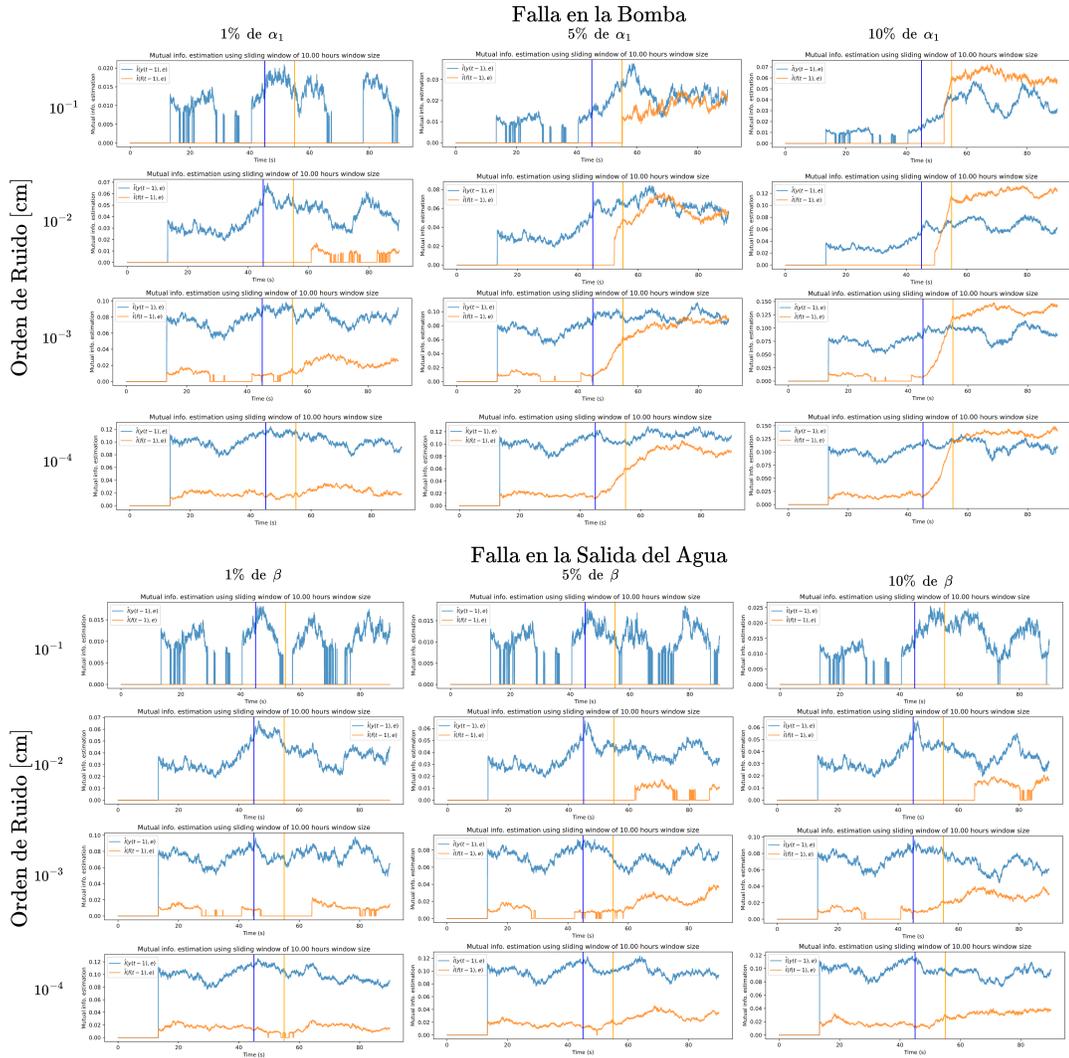


Figura A.2: Modelo ajustado por 1000 épocas

Se muestran resultados para modelos que no tuvieron un entrenamiento adecuado. Particularmente, en las figuras A.1 y A.2 se muestran modelos ajustados a un tiempo de muestreo de 25 segundos, que fueron entrenados por 300 y 1000 épocas, alcanzando un MSE del conjunto de test de 3.3 y 0.38 respectivamente (*naïve* MSE = 23.2). Notar que si bien la EMI del residuo respecto a  $f$  se comporta de manera similar que en los experimentos válidos, la EMI c/r al nivel de agua medido ( $y$ ) tiene un comportamiento totalmente fuera de lo esperado, y el valor de esta es sostenidamente  $> 0$  incluso en zonas de operación normal.

# Anexo B. Resultados con distintas semillas de generación de ruido

En las imágenes de la figura B.1 se muestran resultados para distintas semillas de generación de ruido en el conjunto de prueba, para una falla grave en la bomba, y un nivel de ruido del orden de  $10^{-3}$ .

Falla al 10% de  $\alpha_1$ . Distintas ejecuciones con distinta semilla de generación de ruido

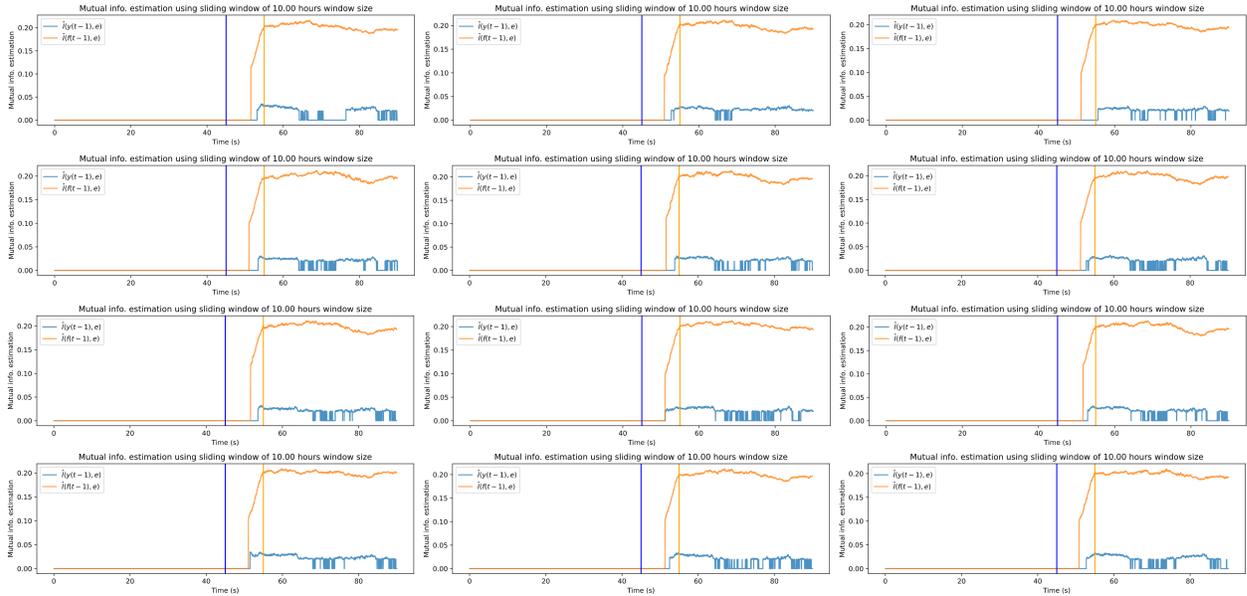


Figura B.1: Resultados para falla grave en la bomba, con distintas semillas de generación de ruido.

## Anexo C. Resultados de aplicar el test con mayor cantidad de autorregresores de $f$ y $h$

En las figuras C.1 - C.10 se muestra el resultado de evaluar la EMI con más autorregresores. Notar que todas hay correcta detección de obsolescencia del modelo, lo que se debe a la dependencia que tienen las variables sensibles a los valores pasados ante el control a lazo cerrado (tal como se explica en la sección 5.4).

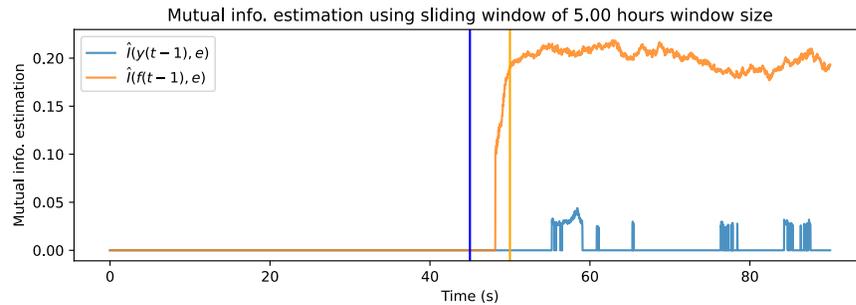


Figura C.1: EMI del primer autorregresor de  $f$  e  $y$

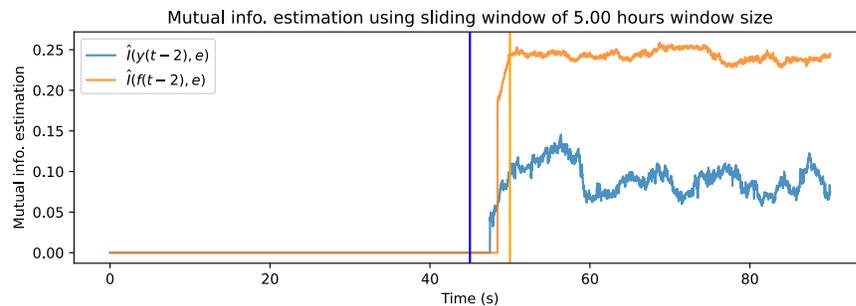


Figura C.2: EMI del segundo autorregresor de  $f$  e  $y$

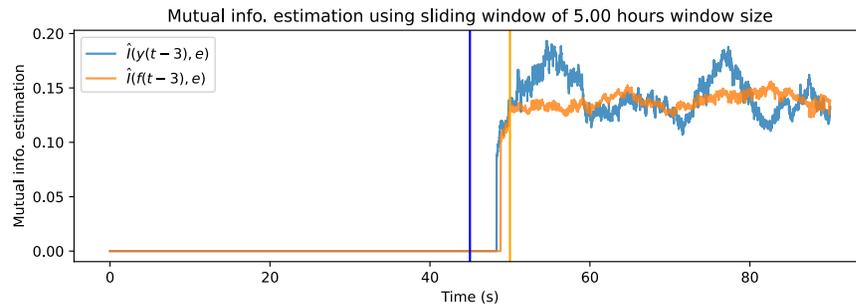


Figura C.3: EMI del tercer autorregresor de  $f$  e  $y$

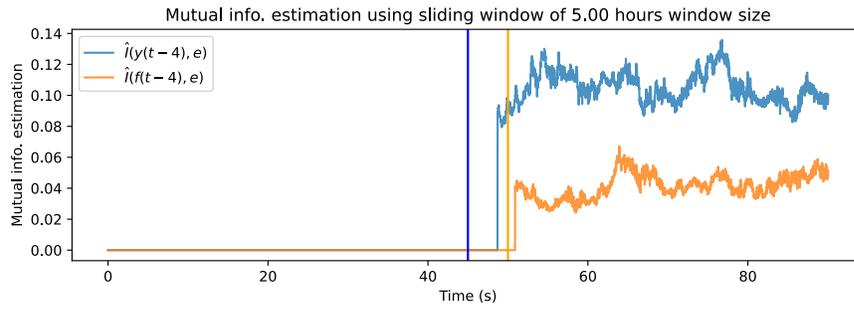


Figura C.4: EMI del cuarto autorregresor de  $f$  e  $y$

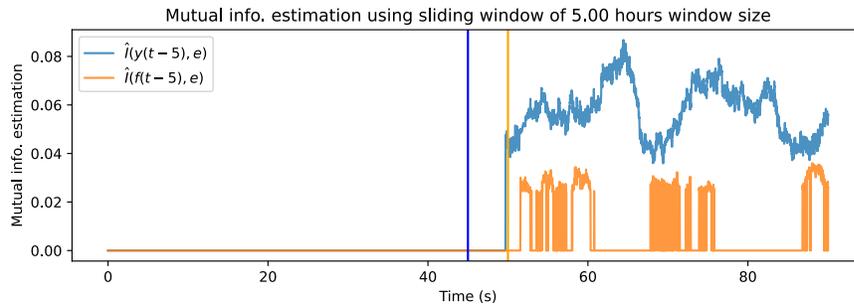


Figura C.5: EMI del quinto autorregresor de  $f$  e  $y$

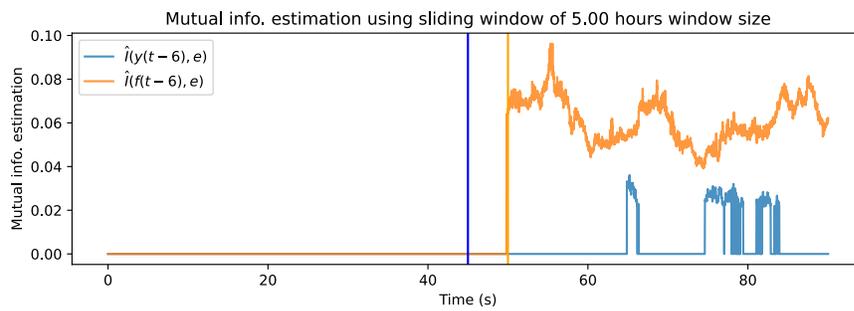


Figura C.6: EMI del sexto autorregresor de  $f$  e  $y$

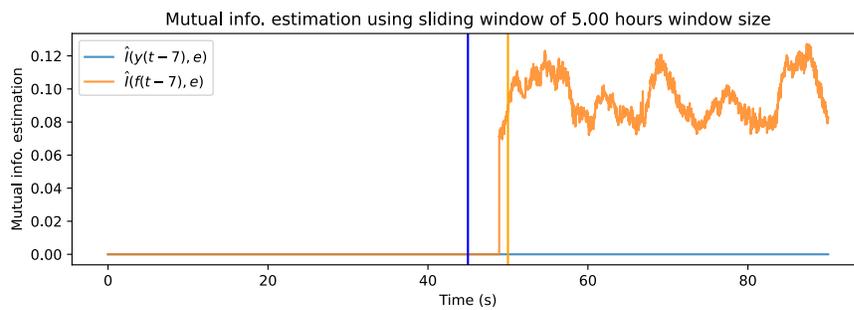


Figura C.7: EMI del séptimo autorregresor de  $f$  e  $y$

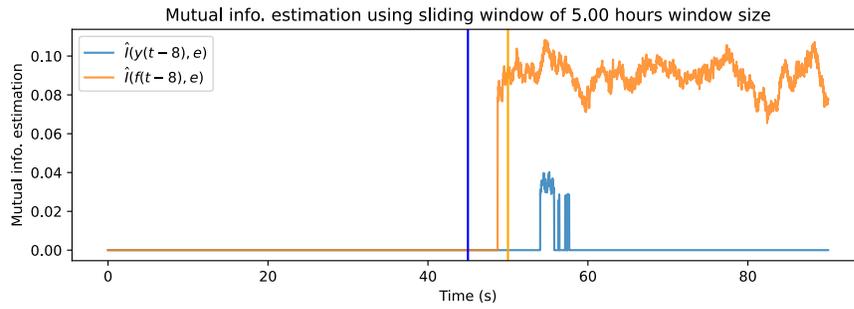


Figura C.8: EMI del octavo autorregresor de  $f$  e  $y$

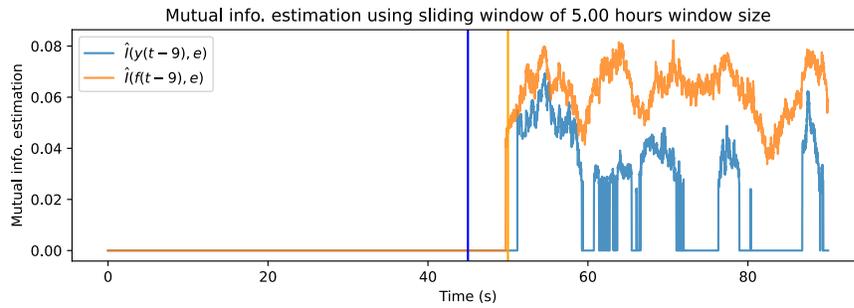


Figura C.9: EMI del noveno autorregresor de  $f$  e  $y$

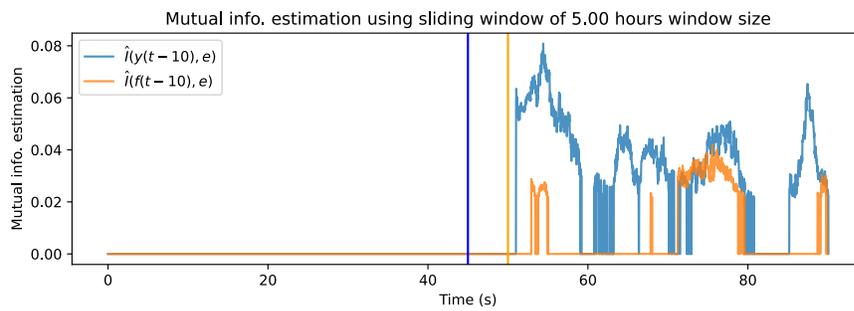


Figura C.10: EMI del décimo autorregresor de  $f$  e  $y$

## Anexo D. Herramientas Computacionales

Esta sección del anexo tiene como objetivo exponer paso a paso el flujo de trabajo descrito en la memoria, desde una perspectiva técnica orientada a las herramientas computacionales utilizadas, mostrando etapas y código relevante. Para más información y la obtención de este código, visitar el repositorio de [GitHub](#) dedicado. Algunos fragmentos de código han sido comentados excesivamente si se miran desde una perspectiva de buenas prácticas de programación, pero ha sido expuesto así en este informe con el objetivo de explicar qué se está haciendo en cada bloque de código. Estos comentarios no están en los archivos oficiales.

### D.1. Generación de Datos: Matlab y Simulink

Para generar los datos del estanque desde MATLAB y Simulink, se utilizó la plantilla desarrollada en [5], sumando el lazo de control de nivel y la adición de ruido, tal como se muestra en la figura D.1. Notar que los bloques generadores de señales pueden contener cualquier señal definida en ellos, con lo que la señal de referencia es una concatenación de funciones `step` que simulan cambios de referencia (ver sección D.2).

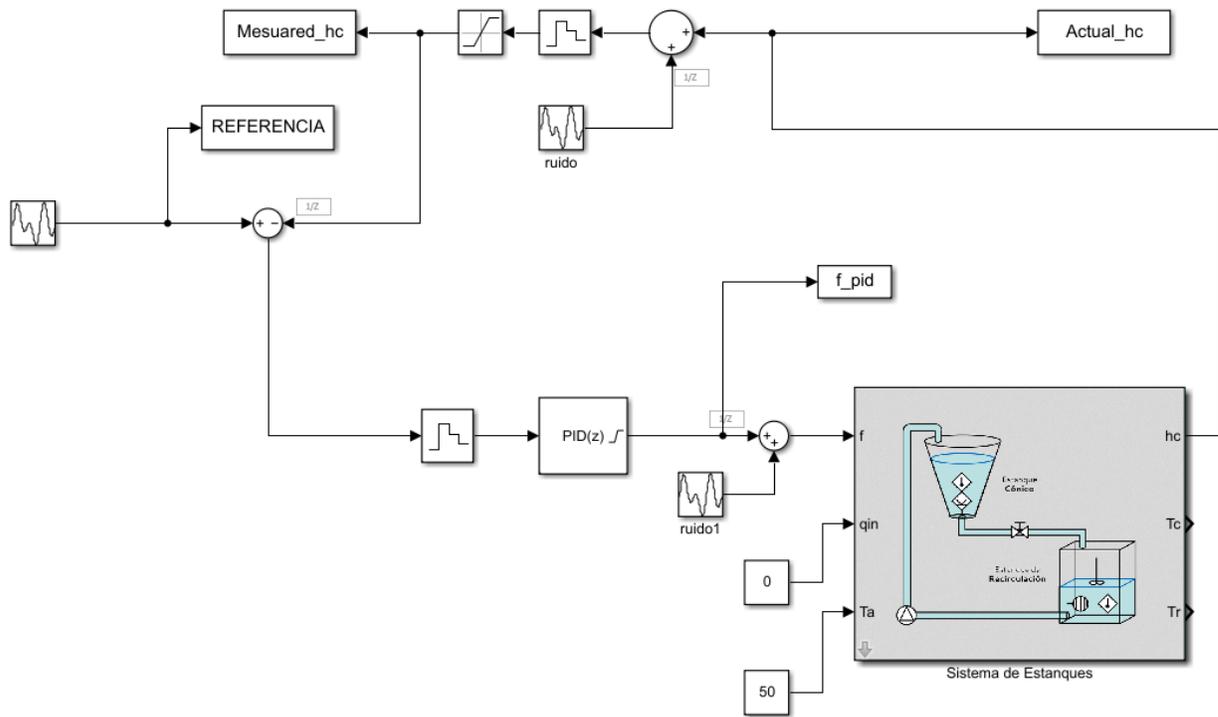


Figura D.1: Diagrama de simulink utilizado, notar que abajo a la derecha está el simulador del estanque desarrollado en [5], regido por las ecuaciones de este.

Para la simulación de fallas, se modificó la estructura original de la plantilla, para poder modificar a voluntad las constantes de perturbación ( $\delta_1, \delta_2$ ), tal como se muestra en la figura D.2. En ella se especifica además en dónde se programaron las diversas fallas.

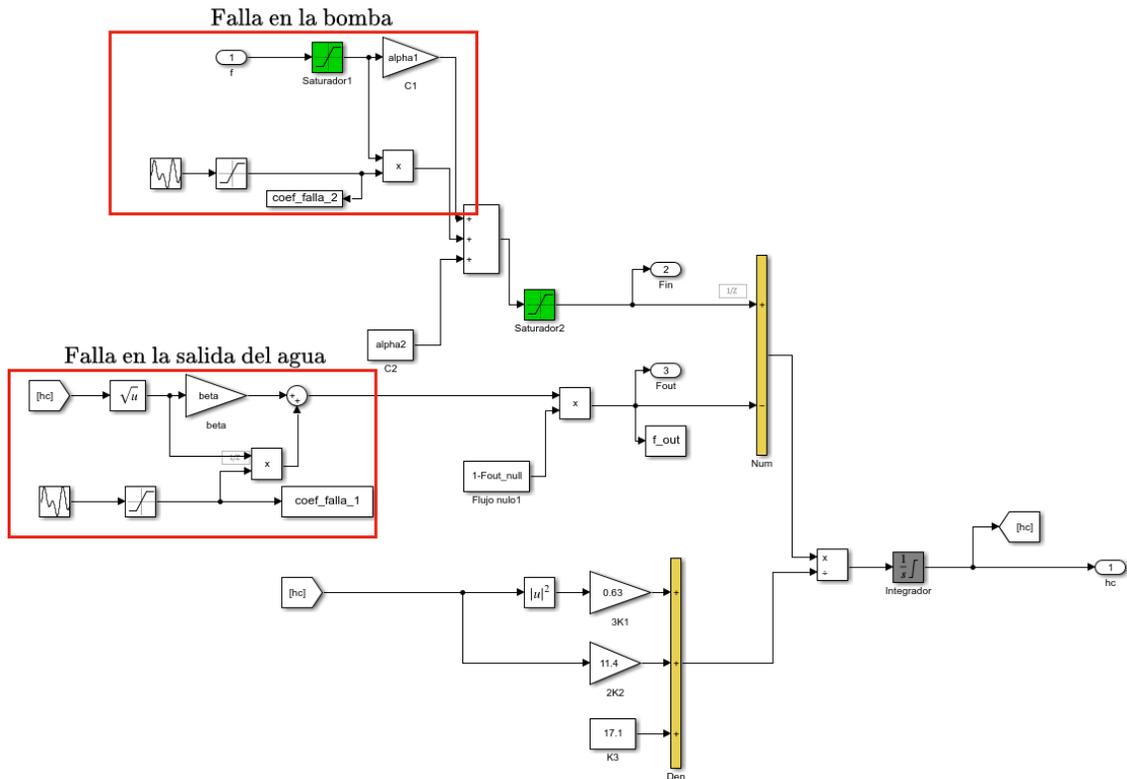


Figura D.2: Sección modificada correspondiente a la simulación de las ecuaciones del sistema. Notar que las constantes llamadas en el diagrama como “coef\_falla\_1” y “coef\_falla\_2” corresponden a los coeficientes de perturbación  $\delta_2$  y  $\delta_1$  respectivamente.

Para exportar los datos a formato *.csv*, se utilizó el siguiente *script*, el cual debe ser ejecutado posterior a la ejecución de la simulación.

```

1 %cargar datos desde el Workspace
2 TIME = f_pid.Time;
3 F_PID = f_pid.Data;
4 HC = Mesured_hc.Data;
5 Referencia = REFERENCIA.Data;
6
7 %Generar tabla de datos
8 data = [TIME, Referencia, HC, F_PID];
9 DATOS_MODELO = data;
10
11 #Exportar como .csv
12 writematrix(DATOS_MODELO, 'path/nombreachivo.csv');

```

## D.2. Generación de Referencias: Microsoft Excel

Tal como se menciona en la sección anterior, para generar referencias aleatorias, se requiere de una concatenación de funciones **step**, las cuales se ingresan a MATLAB en formato de cadena (texto). Debido a lo anterior, para generar referencias aleatorias dentro del rango de operación del sistema, se desarrolló una plantilla de Excel (ver figura D.3), que mediante la

definición del rango de operación en una función `ALEATORIO()`, retorna un texto continuo correspondiente a la entrada que MATLAB entiende.

	A	B	C	D	E	F	G	H	I	J
4				DEFINIR RANGO DE OPERACION				Texto a concatenar:		texto concatenado:
5	Inicial:	1	40	40	0	0		step(0,0,40)+	TO MATLAB:	step(0,0,40)+step(400,0,2)+step(800,0,-20)+step(1200,0,23)+step(1600,0,0)+step(2000,0,3)+
6	Repetir:	2	2	42	400	200		step(400,0,2)+		
7		3	-20	22	800	400		step(800,0,-20)+		
8		4	23	45	1200	600		step(1200,0,23)+		
9		5	0	45	1600	800		step(1600,0,0)+		
10		6	3	48	2000	1000		step(2000,0,3)+		
11		7	0	48	2400	1200		step(2400,0,0)+		
12		8	2	50	2800	1400		step(2800,0,2)+		
13		9	-30	20	3200	1600		step(3200,0,-30)+		
14		10	1	21	3600	1800		step(3600,0,1)+		
15		11	7	28	4000	2000		step(4000,0,7)+		
16		12	14	42	4400	2200		step(4400,0,14)+		
17		13	-12	30	4800	2400		step(4800,0,-12)+		
18		14	11	41	5200	2600		step(5200,0,11)+		
19		15	7	48	5600	2800		step(5600,0,7)+		
20		16	-28	20	6000	3000		step(6000,0,-28)+		
21		17	1	21	6400	3200		step(6400,0,1)+		
22		18	18	39	6800	3400		step(6800,0,18)+		
23		19	2	41	7200	3600		step(7200,0,2)+		
24		20	7	48	7600	3800		step(7600,0,7)+		

Figura D.3: Plantilla generadora de referencias, notar que se logra un texto concatenado que permite ingresar un gran número de cambios de referencia a MATLAB.

### D.3. Ordenamiento y Limpieza de Datos: Pandas y NumPy

Una vez disponibles los datos, se deben impotar en Python para el ajuste de modelos. Para ello, se utilizan las librerías Pandas y Numpy.

En el siguiente fragmento de código, se muestra cómo importar los datos:

```

1 #importar librerías
2 import pandas as pd
3 import numpy as np
4
5 #Cargar datos e incluir headers
6 data_tanque = pd.read_csv('./path/nombreadarchivo.csv', names = ['tiempo', 'referencia', 'hc'
    ↪ , 'f_pid'], low_memory= False)[1:].reset_index(drop=True)
7
8 #muestrear segun el tiempo de muestreo
9 df_sampled = data_tanque.copy()
10 sampling_time = 25 #por ejemplo, 25 segundos
11 df_sampled = df[::sampling_time] #se retorna el dataframe sampleado

```

### D.4. Generación y Análisis de Modelos: PyTorch y SciKitLearn

Para generar los datos de entrenamiento y testing, se utiliza la librería Torch:

```

1 import torch
2 import torch.nn as nn
3 import torch.nn.functional as F
4 torch.cuda.is_available()
5
6 #verificar si hay GPU disponible
7 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
8 print(device)

```

La red neuronal es definida en el siguiente código:

```
1 class MLP(nn.Module):
2     def __init__(self):
3         super(MLP, self).__init__()
4         self.layers = nn.Sequential(
5             nn.Linear(4, 200), #4 features in
6             nn.ReLU(),
7             nn.Linear(200, 50),
8             nn.ReLU(),
9             nn.Linear(50, 10),
10            nn.ReLU(),
11            nn.Linear(10, 1), #1 feature out
12            nn.ReLU(),
13
14        )
15
16    def forward(self, x):
17        x = self.layers(x)
18        return x
```

Para cargar los datos que se van a ingresar a la red neuronal, se utiliza el siguiente fragmento de código

```
1 #Set de train (se eliminan los 3 primeros valores para evitar tratar con NaNs en el
   ↪ entrenamiento):
2 X_train = torch.from_numpy(np.vstack(((df['hc'].shift(1).to_numpy()[3:],
3                                     df['f_pid'].shift(1).to_numpy()[3:],
4                                     df['hc'].shift(2).to_numpy()[3:],
5                                     df['f_pid'].shift(2).to_numpy()[3:],
6                                     )).T).float()
7 y_train = torch.from_numpy(((df['hc'].to_numpy()[3:step]))).float()
8
9 #data de test
10
11 data_tanque = pd.read_csv('./path/test_data.csv', names = ['tiempo', 'referencia', 'hc', '
   ↪ f_pid'], low_memory= False)[1:].reset_index(drop=True)
12 data_tanque.head()
13 df_test = data_tanque.copy()
14 sampling_time = 25
15 df_test = df_test[:, :sampling_time][3:]
16
17 #Define X_test and y_test
18 X_test = torch.from_numpy(np.vstack(((df_test['hc'].shift(1).to_numpy()[3:],
19                                     df_test['f_pid'].shift(1).to_numpy()[3:],
20                                     df_test['hc'].shift(2).to_numpy()[3:],
21                                     df_test['f_pid'].shift(2).to_numpy()[3:],
22                                     )).T).float()
23 X_test = X_test.to(device)
24 y_test = torch.from_numpy(((df_test['hc'].to_numpy()[3:]))).float()
25 y_test = y_test.to(device)
```

Para calcular el MSE de la predicción *naive*, se utiliza la librería SciKit Learn:

```
1 #naive prediction: tomar el valor inmediatamente anterior en la serie de tiempo
2 from sklearn.metrics import mean_squared_error
3 y_pred_naive = (df_test['hc'].shift(1)[1:].values)
4 #replace nans by 0
5 y_pred_naive[np.isnan(y_pred_naive)] = 0
6 naive_loss = mean_squared_error((df_test['hc'])[1:].values), y_pred_naive)
7 print(naive_loss)
```

Finalmente, se entrena la red. En este ejemplo se utiliza validación cruzada para evitar el sobreajuste:

```
1 #import kfold to do crossvalidation in timeseries
2 from sklearn.model_selection import KFold
3
4 model = MLP()
5 model.to(device)
6 optimizer = torch.optim.Adam(model.parameters(), lr=0.00005)
7 loss_fn = nn.MSELoss()
8
9 loss_curve_train =[naive_loss*0.05]
10 loss_curve_valid =[naive_loss*0.05]
11 saved_i = 0
12 epochs = 10000
13 nsplits = 5
14 kfold = KFold(n_splits=nsplits, shuffle=True)
15
16 for epoch in range(epochs):
17     mean_loss_val = 0
18     mean_loss_train = 0
19     for train_index, valid_index in kfold.split(X_train):
20         X_train_cv, X_valid_cv = X_train[train_index], X_train[valid_index]
21         y_train_cv, y_valid_cv = y_train[train_index], y_train[valid_index]
22
23         model.train()
24         y_pred = model(X_train_cv)
25         loss = loss_fn(y_pred.squeeze(), y_train_cv)
26         optimizer.zero_grad()
27         loss.backward()
28         optimizer.step()
29         #validation
30         model.eval()
31         with torch.no_grad():
32             y_pred_val = model(X_valid_cv)
33             loss_val = loss_fn(y_pred_val.squeeze(), y_valid_cv)
34             mean_loss_val += loss_val.item()
35             mean_loss_train += loss.item()
36
37     mean_loss_val = mean_loss_val/nsplits
38     mean_loss_train = mean_loss_train/nsplits
39
```

```

40 loss_curve_train.append(mean_loss_train)
41 loss_curve_valid.append(mean_loss_val)
42
43 # Guardar el mejor modelo, asegurarse que sea representativo para el set de train y
  ↪ validacion
44 if mean_loss_train <= min(loss_curve_train) and (mean_loss_val <= min(
  ↪ loss_curve_train) or mean_loss_val <= min(loss_curve_valid)):
45     print('saving model...')
46     saved_i = epoch
47     torch.save(model.state_dict(), '../modelos torch/st25s_m.pth')
48     print('Model saved')
49
50     print('Epoch: {}, Loss validation: {:.4f}, Loss train: {:.4f}'.format(epoch, mean_loss_val,
  ↪ mean_loss_train))
51 #imprimir en pantalla la ultima epoca guardada
52 print(f'last epoch saved {saved_i}')
53
54 #testear el modelo
55 model.eval()
56 y_pred_gpu = model(X_test)
57 #to cpu
58 y_pred = y_pred_gpu.cpu().detach().numpy()
59 y_test = y_test.cpu().detach().numpy()
60
61 loss = mean_squared_error(y_pred[2:], y_test[2:])
62 print('Loss test: {:.4f}'.format(loss.item()))

```

De esta forma, basta con reemplazar el dataset de test por el dataset de interés a evaluar, y repetir el testeo del modelo de las últimas líneas de código mostradas, para generar los datos de estimación a analizar con el test de información mutua.

## D.5. Almacenamiento de Datos: Google Drive

Como plataforma de almacenamiento de datos, se utilizó Google Drive, en la cual se subieron las predicciones ordenadas correctamente según los tiempos de muestreo y niveles de ruido, tal como se muestra en la figura

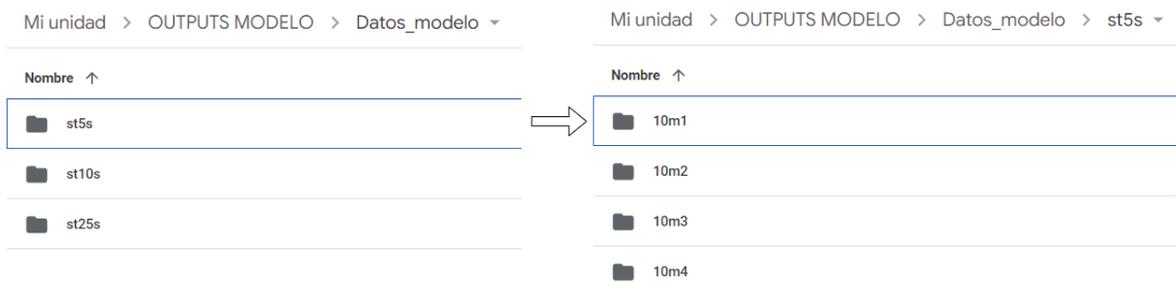


Figura D.4: Almacenamiento de los datos en Google Drive

## D.6. Realización del Test y Automatización del proceso: Python en Google Colaboratory

Para evitar problemas de compatibilidad, el test fue desarrollado en un entorno virtual de Google Colaboratory (también llamado Colab). Para cargar los datos se dió acceso a Google Drive, mientras que para importar la librería, fue necesario adaptar los requerimientos de esta, generando una versión no oficial compatible con el entorno de Colab. Esta versión está disponible en el [GitHub](#) dedicado. La función desarrollada que realiza el test y retorna las figuras se muestra en el siguiente código

```
1 import sys
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 sys.path.insert(1, './src/build')
6 from TSP import TSP
7
8 def experimentos(df,ws,sf,step,name_emi=None,name_figure=None,n_reg=3, lambda =
   ↪ 0.000075):
9     """
10    PARAMETERS:
11    df: pandas DataFrame with the data
12    ws: window size in hours
13    sf: hour when the failure starts
14    step: sampling time (in seconds)
15    name_figure: the name of the new figure.
16    n_reg: number of regressors
17    lambda = regularization parameter from the TSP library
18    """
19
20    tsp = TSP(0.001, 0.1, lambda)
21
22    window_size = int(3600/step*ws)
23
24    start =int(3600/step*0)+n_reg
25    stop = int(3600/step*0-1) -1
26
27    #DEFINICION DE ESPACIO PARA GUARDAR LOS ARRAYS
28    d_ys = {}
29    d_fs = {}
30
31
32    d_y_tsp = {}
33    d_f_tsp = {}
34    d_conj_tsp = {}
35
36    d_emi_y = {}
37    d_emi_f = {}
38    d_emi_conj = {}
39
40    d_emis_y = {}
```

```

41 d_emis_f = {}
42
43 d_emis_y_filtered = {}
44 d_emis_f_filtered = {}
45
46 y = df['hc'][start:stop]
47
48 y_hat = df['y_hat'][start:stop]
49
50 #residuos
51 e = {0: (y - y_hat )}
52
53
54 for i in range(n_reg):
55     d_ys[i] = df['hc'].shift(i+1).rolling(1).mean()[start:stop]
56     d_fs[i] = df['f_pid'].shift(i+1).rolling(1).mean()[start:stop] + np.random.normal(0,
57         ↪ 0.00001, len(y)) #ADICION DE RUIDO LEVE A f PARA EVITAR EXCEPCIONES
58         ↪ CAUSADAS POR UN CASO DEGENERADO
59     d_emis_y[i] = [0] * (window_size + start)
60     d_emis_f[i] = [0] * (window_size + start)
61     d_emis_y_filtered[i] = [0] * (window_size + start)
62     d_emis_f_filtered[i] = [0] * (window_size + start)
63     all_emis = []
64
65 for k in range(len(e)):
66     for j in range(n_reg):
67         for i in range(len(y)-(window_size)):
68
69             e_tsp = np.copy(np.expand_dims(e[k][i:i+window_size], axis=1), order='F')
70             d_y_tsp[j] = np.copy(np.expand_dims(d_ys[j][i:i+window_size], axis=1), order='F')
71
72             d_f_tsp[j] = np.copy(np.expand_dims(d_fs[j][i:i+window_size], axis=1), order='F')
73
74             d_conj_tsp[j] = np.copy(np.stack((d_ys[j][i:i+window_size],d_fs[j][i:i+window_size]),
75                 ↪ axis=-1), order='F')
76
77             #REALIZACION DEL TEST
78             tsp.grow(d_f_tsp[j], e_tsp)
79             tsp.regularize()
80             d_emi_f[j] = tsp.emi()
81
82             tsp.grow(d_y_tsp[j], e_tsp)
83             tsp.regularize()
84             d_emi_y[j] = tsp.emi()
85
86             d_emis_y[j].append(d_emi_y[j])
87             d_emis_f[j].append(d_emi_f[j])
88
89 # FILTRO DE PEAKS INDIVIDUALES

```

```

90 for p in range(window_size,len(d_emis_y[j])-1):
91
92     if np.count_nonzero(d_emis_y[j][p-window_size:p]) >= 1/3*window_size:
93         try:
94             d_emis_y_filtered[j].append(d_emis_y[j][p])
95         except:
96             d_emis_y_filtered[j].append(0)
97     else:
98         d_emis_y_filtered[j].append(0)
99
100
101 for p in range(window_size,len(d_emis_f[j])-1):
102
103     if np.count_nonzero(d_emis_f[j][p-window_size:p]) >= 1/3*window_size:
104         try:
105             d_emis_f_filtered[j].append(d_emis_f[j][p])
106         except:
107             d_emis_f_filtered[j].append(0)
108     else:
109         d_emis_f_filtered[j].append(0)
110
111 # GRAFICOS
112 plt.figure(figsize=(10,3))
113
114 tpo = np.linspace(0, len(d_emis_y_filtered[j])*step/3600, num = len(
↪ d_emis_y_filtered[j]))
115
116 plt.plot(tpo, np.array(d_emis_y_filtered[j]), label=r'$\hat{\{I\}}(y(t-{:d}),e)$'.format(j
↪ +1), alpha=0.8)
117 plt.plot(tpo, np.array(d_emis_f_filtered[j]), label=r'$\hat{\{I\}}(f(t-{:d}),e)$'.format(j
↪ +1), alpha=0.8)
118
119 plt.title("Mutual info. estimation using sliding window of %.2f hours window size" % (
↪ window_size/(3600/step)))
120 plt.ylabel('Mutual info. estimation')
121 plt.xlabel('Time (s)')
122 plt.axvline(x=int(sf), color = 'b')
123 plt.axvline(x=int(sf+ws), color = 'orange')
124 plt.legend()
125
126 #GUARDAR FIGURAS
127 if name_figure:
128     name_figure_2 = name_figure + f'_{j}.svg'
129     print(name_figure_2)
130     plt.savefig(name_figure_2, bbox_inches='tight', dpi=150)
131
132 plt.show()
133
134 #GUARDAR ARRAYS
135 if name_emi:
136
137     np.save( f'{name_emi}_hc.npy',np.array(d_emis_y))

```

```

138     np.save( f'{name_emi}_f.npy',np.array(d_emis_y))
139
140     print('\n\n')
141
142     #VOLVER A CONFIGURAR LOS ESPACIOS, PARA LA SIGUIENTE EJECUCION
143     for i in range(n_reg):
144         d_ys[i] = df['hc'].shift(i+1).rolling(1).mean()[start:stop]
145         d_fs[i] = df['f_pid'].shift(i+1).rolling(1).mean()[start:stop]+ np.random.normal(0,
↪ 0.000001, len(y))
146
147         d_emis_y[i] = [0] * (window_size + start)
148         d_emis_f[i] = [0] * (window_size + start)
149
150         d_emis_y_filtered[i] = [0] * (window_size + start)
151         d_emis_f_filtered[i] = [0] * (window_size + start)
152
153     print('\n')

```

Finalmente, para automatizar el proceso, basta con buscar todos los `.csv` de una carpeta de Drive, e iterar con la función `experimentos()` en cada dataset, tal como se muestra en el siguiente código:

```

1 #fijar el path a scrapear
2 %cd /content/drive/MyDrive/OUTPUTS MODELO/path_de_los_csvs
3
4 import glob
5 import os
6
7 #hallar todos los csv
8 path = os.getcwd()+'/'
9 csv_files = glob.glob(os.path.join(path, "*.csv"))
10 csv_files.sort()
11
12 #iterar en ellos
13 for csv_file in csv_files:
14     print(csv_file[43:-3])
15     df = pd.read_csv(csv_file)
16     name_figure = f'./emi_' + csv_file[43:-3] + 'svg'
17     name_emi = None
18     experimentos(df, 10, 45, 25, name_emi = name_emi, name_figure = name_figure, n_reg
↪ = 1, lmbd = 0)

```