TRABAJO FIN DE GRADO

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

"Un nuevo enfoque de modelado y control para la regulación del pH en fotobiorreactores *raceway*"

Curso: 2021/2022

Modalidad TFG: Trabajo Técnico

Alumno/a:

Malena Caparroz

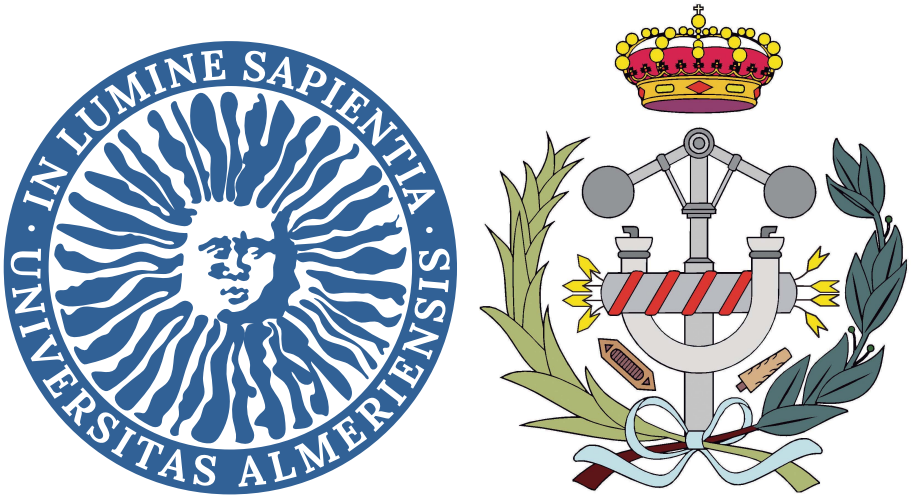
Director/es:

José Luis Guzmán Sánchez

Manuel Berenguel Soria

UNIVERSIDAD DE ALMERÍA

ESCUELA SUPERIOR DE INGENIERÍA



GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL

TRABAJO FIN DE GRADO

Un nuevo enfoque de modelado y control para la regulación del pH en fotobiorreactores raceway

---

Alumno: Malena Caparroz
Director: José Luis Guzmán Sánchez
Codirector: Manuel Berenguel Soria
Fecha: Julio de 2022

Malena Caparroz	*José Luis Guzmán Sánchez*	*Manuel Berenguel Soria*

Índice general

Agradecimientos

Al director **José Luis Guzmán Sánchez** y codirector **Manuel Berenguel Soria** por el apoyo y la atención que han mostrado en todo momento.

Y a los compañeros del proyecto **HYCO2BIO** y del Instituto Andaluz de Investigación y Formación Agraria, Pesquera, Alimentaria y de la Producción Ecológica (**IFAPA**) por la ayuda prestada durante la implementación del trabajo en los fotobiorreactores.

Este trabajo fin de grado ha sido financiado con el Proyecto del Plan Nacional **Control híbrido y optimización de una biorrefinería sostenible para la producción industrial de microalgas (HYCO2BIO), PID2020-112709RB-C21 del Ministerio de Ciencia, Innovación y Universidades**. Los ensayos experimentales se han realizado en las instalaciones del Convenio entre la Universidad de Almería (UAL) e IFAPA.

Dedicatoria

El presente trabajo de fin de grado está dedicado a todas las personas que me han apoyado a lo largo de esta etapa.

Especialmente a mis **padres**, a mis **hermanos** y a mis **abuelos**.

Acrónimos

CART	*Classification and Regression Trees*
E/S	Entrada/Salida
IFAPA	Instituto Andaluz de Investigación y Formación Agraria, Pesquera, Alimentaria y de la Producción Ecológica
MATLAB	*Matrix Laboratory*
N4SID	*Numerical Algorithms for Subspace State Space System Identification*
OD	Oxígeno Disuelto
OLE	*Object Linking and Embedding*
OPC	*OLE for Process Control*
PEM	*Prediction Error Minimization*
PI	Proporcional Integral
PLC	*Programmable Logic Controller*
PT100	*Platinum Resistance Thermometer*
RTD	*Resistance Temperature Detector*
SCADA	*Supervisory Control And Data Acquisition*
TFG	Trabajo Fin de Grado
UAL	Universidad de Almería

Nomenclatura

Variable/Parámetro	Símbolo	Unidades
Modelo de función de transferencia		
Ganancia estática	k	min/L
Constante de tiempo	τ	s
Tiempo de retardo	t_r	s
Controlador PI		
Ganancia proporcional	K_p	L/min
Tiempo integral	T_i	s
Constante de tiempo de bucle cerrado	τ_{BC}	s
Constante de tracking para esquema *antiwindup*	T_t	
Tiempo de muestreo	T_s	s
Caudal de CO_2 inyectado	CO_2	L/min
Predictores del árbol de regresión		
Radiación global		W/m^2
Temperatura del medio		ºC
Nivel del medio		cm
Árboles de regresión		
Suma total de errores cuadráticos	$S(T)$	$\sum_{l=\widetilde{T}} \sum_{i \in l} (y_i - m_l)^2$
Hoja del árbol T	l	
Conjunto de hojas del árbol T	$\widetilde{T}$	
Valores de las variables dependientes	y_i	
Media en la hoja l	m_l	
Error cuadrático medio ponderado	ϵ	$\epsilon = \sum_{i=1}^{n} w_i (y_i - \overline{y})^2$
Peso de la observación i	w_i	
Media ponderada de todas las respuestas	$\overline{y}$	$\overline{y} = \sum_{i=1}^{n} w_i y_i$

Índice de figuras

Índice de tablas

Resumen

Las algas son microorganismos con un gran potencial en la producción de biomasa con alto valor añadido y en el tratamiento de aguas residuales. Su capacidad de crecimiento presenta una fuerte dependencia del pH, el oxígeno disuelto (OD), la radiación global y la temperatura del medio. El pH y el OD pueden controlarse mediante la inyección de CO_2 y aire respectivamente, mientras que la radiación y la temperatura (en general) son factores que en reactores abiertos no pueden ser controlados.

La tasa de crecimiento de las microalgas viene determinada por un factor principal, que es la disponibilidad de luz y depende en gran medida de la concentración del cultivo y el nivel del medio. Además, este término viene ponderado por otros tres que representan la temperatura, el pH y el oxígeno disuelto y cuyos valores se encuentran normalizados entre 0 y 1.

La influencia de la temperatura y el pH son muy similares, ambos presentan un valor óptimo entorno al cual la tasa de crecimiento es máxima, pero fuera de ese valor la tasa decrece drásticamente. En cuanto al oxígeno disuelto, hay un valor a partir del cual la tasa de crecimiento desciende. El pH es, entre todas las variables a controlar, la más crítica y complicada, pues la producción de la fotosíntesis provoca variaciones continuas.

Actualmente, es necesario llegar a un balance entre la complejidad del modelo de pH y la dificultad para desarrollar algoritmos de control. Por un lado, modelos muy complejos reflejan con mucha exactitud el comportamiento del sistema tanto espacial como temporalmente, pero dificultan el diseño de controladores. Por el otro, modelos muy simples facilitan el diseño de los algoritmos de control pero su excesiva sencillez provoca un modelado del sistema poco exacto y la pérdida de eficacia de los controladores cuando se presentan condiciones ambientales distintas a las que había a la hora de obtener el modelo, por lo que estos modelos tendrían que ser constantemente calibrados.

Es por las razones expuestas anteriormente que se han desarrollado modelos de árboles de regresión que sean capaces de calcular los parámetros de un modelo simple. Se buscaba conseguir un modelo sencillo pero que, en cierto modo, sea capaz de autocalibrarse en función de las condiciones a las que se encuentre sometido el sistema. Este modelo ha sido utilizado tanto para fines de modelado como para fines de control.

Finalmente, tras la realización de los ensayos se han obtenido resultados satisfactorios que indican un correcto funcionamiento tanto del estimador de pH como del controlador adaptativo.

Palabras clave: *Raceway, Predicción de pH, Árbol de regresión, Control adaptativo*

Abstract

Algae are microorganisms with great potential in the production of biomass with high added value and in wastewater treatment. Their growth capacity has a strong dependence on pH, dissolved oxygen (DO), global radiation and media temperature. Factors as pH and DO can be controlled by injection of CO_2 and air respectively, while radiation and temperature (in general) cannot be controlled in open reactors.

The growth rate of microalgae is determined by one main factor, which is the availability of light and is highly dependent on the concentration of the culture and the level of the medium. In addition, this term is weighted by three other terms representing temperature, pH and dissolved oxygen whose values are normalized between 0 and 1.

The influence of temperature and pH are very similar, both have an optimum value around which the growth rate is maximum, but outside this value the rate decreases drastically. As for dissolved oxygen, there is a value above which the growth rate decreases. Of all the variables to be controlled, pH is the most critical and complicated, since the production of photosynthesis causes continuous variations.

Currently, it is necessary to reach a balance between the complexity of the pH model and the difficulty in developing control algorithms. On one side, very complex models reflect very accurately the behavior of the system both spatially and temporally, but make it difficult to design controllers. On the other hand, very simple models facilitate the design of control algorithms but their excessive simplicity leads to an inaccurate modeling of the system and the deterioration of the performance of the controllers when environmental conditions are different from those at the time of obtaining the model. Thus, these models would have to be constantly calibrated.

Hence, for the reasons stated above, regression tree models have been developed, that are capable of estimating the parameters of a simple model. The aim was to obtain a simple model but one that, in a way, is capable of self-calibrating depending on the conditions to which the system is subjected.

Finally, after performing the tests, satisfactory results have been obtained, indicating correct operation of both the pH estimator and the adaptive controller.

Keywords: *Raceway, pH prediction, Regression tree, Adaptive control.*

1

Introducción

1.1. Motivación del proyecto

Una línea de investigación que ha cobrado mucha importancia en los últimos años es la referida a la producción de microalgas a escala industrial. Las microalgas son microorganismos fotosintéticos con la capacidad de crecer y reproducirse en entornos sin agua limpia o suelo fértil, convirtiendo la energía solar y fuentes carbónicas como el CO_2 en biomasa.

Una de las variables más importantes en los procesos relacionados con las microalgas es el pH, que determina directamente el rendimiento global del sistema de producción, y suele controlarse mediante algoritmos clásicos de encendido y apagado durante el día, sin tener en cuenta la dinámica del sistema ni las perturbaciones del proceso. Esto se debe principalmente a que, al tratarse de un sistema biológico, es muy difícil obtener un modelo sencillo que refleje adecuadamente el comportamiento del mismo. Si se quiere capturar correctamente la dinámica se debe trabajar con modelos muy complejos que dificultan el desarrollo de los algoritmos de control.

En la literatura se pueden encontrar multitud de modelos, cuya complejidad varía enormemente de acuerdo a los objetivos del mismo [2, 7, 8]. Sin embargo, los modelos sencillos con fines de control que existen hoy en día no se ajustan a la enorme variedad de condiciones ambientales que se presentan, que tienen una gran influencia sobre la dinámica del sistema.

Por ello, con la motivación de realizar un control eficiente del pH del sistema, se busca obtener un modelo sencillo del pH con fines de control que sea capaz de ajustarse a las condiciones del ambiente y capturar la dinámica del sistema en todo momento.

1.2. Contexto

Las microalgas y cianobacterias contribuyen enormemente en la lucha contra el calentamiento global del planeta, pues capturan energía solar y producen una gran cantidad de oxígeno, además de convertir fuentes carbónicas como el CO_2 en biomasa. Para esto, necesitan un aporte adicional de nutrientes tales como carbono, nitrógeno y fósforo. El carbono se suministra mediante inyección de CO_2, puesto que también sirve para regular el pH del sistema. Los nutrientes restantes pueden aportarse de forma directa, introduciéndolos en soluciones al medio acuoso, o ser absorbidos del propio medio, puesto que el nitrógeno y el fósforo son los principales contaminantes de las aguas residuales, por lo que este proceso se convierte también en una solución al tratamiento de estas. Además, las microalgas pueden utilizarse en la elaboración de productos de alto valor en el campo de la cosmética, nutrición humana o alimentación animal.

Es por todos los beneficios expuestos anteriormente que son actualmente objeto de diversos estudios que buscan optimizar su crecimiento, ya sea para maximizar la producción de biomasa o para obtener una mayor absorción de gases de combustión industrial. Con estos fines, se busca someter al sistema a los valores óptimos de pH, oxígeno disuelto, temperatura y radiación, de modo que se maximice el crecimiento de los microorganismos.

El fotobiorreactor en el que se producen las microalgas es un factor clave, dado que la complejidad del control del sistema dependerá de su diseño y forma de operación. Se utilizan principalmente dos tipos de fotobiorreactores:

- **Fotobiorreactores cerrados**

  Presentan una barrera física que separa el cultivo con el ambiente que lo rodea, por lo que las microalgas no estás expuestas a fuentes de contaminación externas. Entre estos se encuentran los reactores tubulares, los de paneles planos o las columnas de burbujeo, y se utilizan comúnmente cuando se busca un producto final de alto valor con cepas muy sensibles a la contaminación [10].

  Entre los reactores cerrados, los **tubulares** son los más extendidos comercialmente (figura 1.1) y en ellos se diferencian dos zonas. Por un lado, una columna de burbujeo donde se produce el proceso de aireación y control de temperatura y, por el otro, el lazo o receptor solar de gran longitud por donde circulan las microalgas impulsadas por una bomba para captar la radiación y producir la fotosíntesis.

En este tipo de reactores se pueden obtener mayores niveles de producción y con biomasa de mayor calidad, pero con elevados costes de producción, un alto consumo energético y de difícil escalado [10].

Figura 1.1: Reactores tubulares de la Universidad de Almería (UAL)

- **Fotobiorreactores abiertos**

  Presentan la ventaja de implicar una inversión considerablemente menor que los reactores cerrados, pues son estructuralmente más sencillos y precisan de materiales menos costosos. Sin embargo, al estar abiertos al ambiente están sometidos a una continua contaminación. Este tipo de reactores se caracterizan por ser grandes estanques con poca profundidad, de modo que se aumenta la productividad gracias a una mejor penetración de la luz. Entre estos se encuentran las balsas, los reactores *Thin-Layers* o de capa fina (figura 1.2a) y los *raceway* (figura 1.2b) y se utilizan para producir biomasa de bajo valor en la que la contaminación de la cepa no sea un problema. Los reactores *raceway* son los más extendidos y serán objeto de estudio en el presente proyecto.

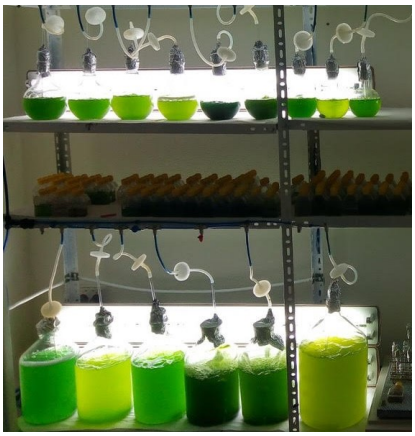
  En los reactores *raceway* se distinguen tres zonas: un foso bajo el suelo de entre 2 y 3 metros de profundidad donde se realiza la inyección de aire y CO_2, el canal por donde circulan las microalgas para que reciban la radiación solar y puedan realizar la fotosíntesis y unas palas que impulsan el cultivo a lo largo del canal. Son los más utilizados a nivel mundial, cubriendo más del 90 % de la producción total con

este tipo de tecnología. Entre sus principales ventajas se encuentran su bajo coste (inferior a 10 €/m^2), su fácil escalado y su bajo consumo energético. Este último aspecto los hace óptimos para aplicaciones que no requieren biomasa de alto valor, como tratamiento de aguas residuales y producción de biocombustibles [10].

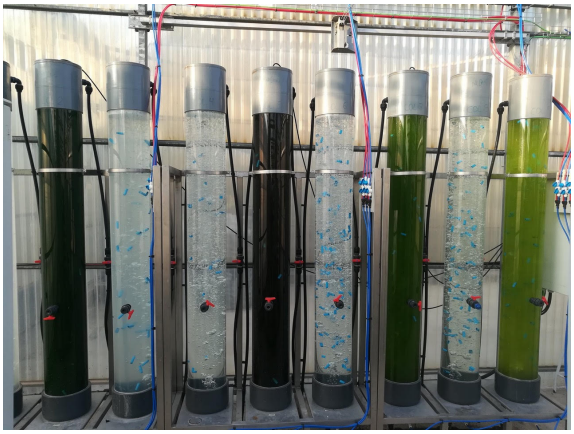
(a) *Thin-Layer* **(b)** *Raceway*

Figura 1.2: Fotobiorreactores abiertos del Convenio UAL-IFAPA

Además, las microalgas siguen un proceso prolongado de adaptación a las condiciones del medio previamente a ser inoculadas en reactores de escala industrial. En dicho proceso, primero se inocula la cepa de interés en pequeños matraces redondos de aproximadamente 250 mL (figura 1.3a, zona superior) y luego se traspasan a matraces de 5 litros (figura 1.3a, zona inferior). En estas dos etapas, el sistema se encuentra aislado en condiciones de laboratorio: sin contaminación externa y con condiciones de luz y temperatura controlados. Posteriormente, se traspasan a columnas de burbujeo (figura 1.3b). Suelen tener un volumen de aproximadamente 500 L y las condiciones de temperatura y radiación ya no son controladas, pero sigue siendo aislado. La próxima etapa, es el traspaso al fotobiorreactor deseado, ya sea abierto o cerrado.



(a) *Inóculo en laboratorio*



(b) *Columnas de burbujeo*

Figura 1.3: Proceso de adaptación del cultivo a condiciones reales en Convenio UAL-IFAPA

La obtención de modelos del proceso en estudio es una tarea muy compleja, pues presenta una dinámica no lineal, con un régimen permanente no estacionario, en presencia de perturbaciones cambiantes y con influencia de una gran cantidad de variables, la mayoría no manipulables, como la radiación solar. Las variables que más influyen en la velocidad de crecimiento de las microalgas son la radiación solar, el aporte de nutrientes, la temperatura del medio, el pH y el oxígeno disuelto. El pH es una de las variables controladas del proceso más influyentes en la productividad, por lo que se convierte en una de las variables de mayor interés y será objeto de estudio en el presente proyecto. Para conseguir un control óptimo del pH se debe llegar a un balance entre el consumo de CO_2 y el aumento de la productividad de biomasa.

A día de hoy, en la literatura existen principalmente dos tipos de modelos para capturar la dinámica del proceso tanto para fotobiorreactores tubulares como para *raceway*. Por un lado, se encuentran los **modelos basados en primeros principios**, donde se representan los principales fenómenos físico-químicos y biológicos del sistema, considerando las relaciones entre la disponibilidad de luz, condiciones de cultivo, velocidad de fotosíntesis, transferencias de materia y mezclas entre estado líquido y gaseoso del sistema. Por lo tanto, estos modelos describen el comportamiento temporal y espacial de las principales variables del proceso. El modelo completo de un fotobiorreactor industrial se divide en el modelo de fotosíntesis, que describe la velocidad de crecimiento de los microorganismos, y el modelo ingenieril del reactor, que describe

los fenómenos biológicos de transferencia de materia y de calor. Este tipo de modelos permiten una comprensión exhaustiva de los distintos fenómenos que tienen lugar en el proceso de producción, por lo que son muy adecuados para el desarrollo de simuladores [5, 6, 11]. Sin embargo, tanta precisión dificulta el desarrollo de algoritmos de control, por lo que se utilizan además **modelos simplificados con fines de control**. Se pueden encontrar tanto simplificaciones de los modelos basados en primeros principios como modelos de primer orden con retardo (respecto a la inyección de CO_2 y aire) que modelan la dinámica del pH y el oxígeno disuelto alrededor de los puntos de operación [10].

Este Trabajo Fin de Grado ha sido realizado en el contexto del proyecto **HYCO2BIO** y en las instalaciones del Convenio UAL-IFAPA. Los resultados principales obtenidos se han enviado a las XLIII Jornadas de Automática [3].

1.3. Objetivos

El proyecto se centrará en el desarrollo de modelos simplificados de la relación del pH del cultivo con la inyección de CO_2 y con la fotosíntesis de los microorganismos, teniendo en cuenta la dependencia de los parámetros del modelo con condiciones de contorno tales como radiación, concentración y temperatura del medio. Por lo tanto, para modelar la relación pH-CO_2 se obtendrá una función de transferencia de primer orden, mientras que para la relación pH-fotosíntesis se utilizará un modelo en espacio de estados.

Con el fin de obtener dichos modelos, se realizará una campaña de ensayos en el reactor de aguas limpias disponible en las instalaciones del Convenio UAL-IFAPA. Una vez obtenida una cantidad suficiente de datos, se obtendrán modelos que posteriormente se utilizarán para entrenar árboles de regresión. Estos se conseguirán utilizando el *Regression Learner Toolbox* de MATLAB, y permitirán estimar los parámetros del modelo en función de los valores de radiación global, temperatura, y otros parámetros que se estimen importantes.

Una vez obtenidos los árboles de regresión, se realizará la validación de estos implementando un estimador de pH en línea en el reactor. Tras comprobar su correcto funcionamiento, estos modelos se utilizarán para diseñar algoritmos de control adaptativos para regulación del pH de los fotobiorreactores.

1.4. Planificación temporal

La planificación del trabajo, organizada por meses y actividades, se muestra en la tabla 1.1.

Mes	Actividad	A	B	C	D	E	F	G	
	Semana								
Octubre 2021	4								
Noviembre 2021	1								
	2								
	3								
	4								
Enero 2022	4								
Febrero 2022	1								
	2								
	3								
	4								
Marzo 2022	1								
	2								
	3								
	4								
Abril 2022	1								
Mayo 2022	1								
	2								
	3								
	4								
Junio 2022	1								
	2								
	3								Total
Horas		40	10	-	220	60	20	100	450

Tabla 1.1: Planificación temporal

Las actividades que aparecen en la tabla 1.1 corresponden a:

A: *Estudio bibliográfico.* Se refiere a la búsqueda de información sobre el funcionamiento de los reactores *raceway*, así como los modelos existentes de pH.

B: *Diseño de ensayos para obtención de datos.* Creación de ensayos con Control Todo/Nada con banda de histéresis para la obtención de datos.

C: *Campaña de ensayos.* Implementación del control diseñado en la fase **B** en el reactor real.

D: *Tratamiento de datos.* Obtención de modelos, árboles de regresión y estimador de pH.

E: *Desarrollo del estimador de pH y de los controladores y prueba en simuladores.*

F: *Implementación del estimador de pH y controladores en reactores de las instalaciones del Convenio UAL-IFAPA.*

G: *Elaboración de la documentación del proyecto.*

1.5. Competencias utilizadas en el Trabajo Fin de Grado

Durante el desarrollo de este Trabajo Fin de Grado (TFG) han sido aplicadas diferentes competencias adquiridas a lo largo de los estudios del Grado en Ingeniería Electrónica Industrial.

Las **competencias básicas** que se han utilizado a lo largo del presente trabajo son:

- **CB1 -** Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
- **CB2 -** Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio .
- **CB3 -** Que los estudiantes tengan la capacidad de reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.
- **CB4 -** Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
- **CB5 -** Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

Durante el desarrollo de este trabajo ha sido necesario trabajar una serie de competencias transversales sin las cuales no hubiera sido posible alcanzar los resultados obtenidos y que completan un perfil profesional en el área de la ingeniería. El uso de diferentes herramientas para el trabajo como para la búsqueda de información se encuentran relacionados con las siguientes **competencias transversales**:

- **UAL 1 -** Conocimientos básicos de la profesión.
- **UAL 2 -** Habilidad en el uso de las TIC.
- **UAL 3 -** Capacidad para resolver problemas.
- **UAL 4 -** Comunicación oral y escrita en la propia lengua.
- **UAL 5 -** Capacidad de crítica y autocrítica.
- **UAL 9 -** Capacidad para aprender a trabajar de forma autónoma.

Por último, ha sido necesario realizar tareas auxiliares y adquirir conocimientos nuevos, como lo es el funcionamiento de reactores *raceway*. Pueden señalarse como las principales **competencias específicas** trabajadas las siguientes:

- **E-CT3 -** Conocimiento en materias básicas y tecnológicas, que les capacite para el aprendizaje de nuevos métodos y teorías, y les dote de versatilidad para adaptarse a nuevas situaciones.
- **E-CRI6 -** Conocimientos sobre los fundamentos de automatismos y métodos de control.
- **E-CTEM6 -** Conocimiento aplicado de los fundamentos de los sistemas y máquinas fluidomecánicas.
- **E-CTEE7 -** Conocimiento y capacidad para el modelado y simulación de sistemas.
- **E-CTEE8 -** Conocimientos de regulación automática y técnicas de control y su aplicación a la automatización industrial.

1.6. Estructura del trabajo fin de grado

El trabajo técnico se divide en un total de 5 capítulos que se detallan a continuación:

1. En el **primer capítulo**, se ha expuesto de manera introductoria la motivación del proyecto, el contexto al que se encuentra sujeto el proyecto, y el objetivo a cumplir durante la realización del mismo. Además, se ha incluido una elaboración de la planificación temporal seguida en la realización de dicho trabajo y un resumen de los resultados.

2. En el **segundo capítulo** se incluye una descripción de los reactores utilizados, incluyendo la cepa de microalgas, los sensores y actuadores y los programas utilizados para llevar a cabo el control. Además, se explican las representaciones del sistema que se utilizarán a lo largo del proyecto y el concepto y la utilidad de los árboles de regresión. También se describen las herramientas informáticas utilizadas para la obtención de modelos. Por último, se explican los algoritmos de control utilizados y el método de sintonía utilizado.

3. El **tercer capítulo** abarca el proceso seguido para conseguir una cantidad suficiente de datos, realizar su tratamiento y obtener los modelos que representen la dinámica del pH.

4. El **cuarto capítulo** muestra los resultados obtenidos tanto en simulación como en los reactores de las instalaciones, en lo que respecta a un estimador de pH y a algoritmos de control clásicos.

5. El **quinto capítulo** expone las conclusiones que se han obtenido tras la realización de este trabajo fin de grado.

1.7. Resumen de resultados

Tras los ensayos en simulación y en los reactores del Convenio UAL-IFAPA, se ha detectado que la técnica utilizada para diseñar un control adaptativo es viable, aunque podría mejorarse entrenándose los árboles de regresión con una mayor cantidad de datos. Para eso, sería necesario realizar una campaña de ensayos en invierno y otra en verano, de modo que se obtengan modelos para condiciones más extremas de radiación y temperatura del medio. Así, la estimación de los parámetros podría mejorar y, por tanto también el control.

Además, una vez que se ha visto que el control funciona adecuadamente, se podría hacer este más agresivo. Puesto que se trata de un problema de rechazo a perturbaciones, se podría imponer una constante de tiempo de bucle cerrado igual a 0.2 veces la constante del sistema en bucle abierto. Así, se conseguiría llevar el pH a la referencia en un período de tiempo menor y se conseguirían mejores índices de comportamiento.

En cuando el modelado del sistema, se debe realizar un esfuerzo por obtener un mejor modelo para la respuesta libre, puesto que se ha visto que no es posible estimarla con los dos modelos obtenidos a lo largo de este trabajo. Por otro lado, se han obtenido resultados muy satisfactorios en modelado de la respuesta forzada y la estimación de pH para controladores PI adaptativos.

En las figuras 1.4 y 1.5 se muestran los resultados de implementar el estimador y un controlador en simulación. En la primera de ellas el controlador es de parámetros fijos, mientras que en la segunda sus parámetros se adaptan a los parámetros del modelo estimados en cada instante. Se puede observar que los resultados tanto para el comportamiento del pH real como del estimado son adecuados, pues se consigue mantener el pH cercano al valor de su consigna y el pH estimado se acerca en todo momento al valor del pH real.

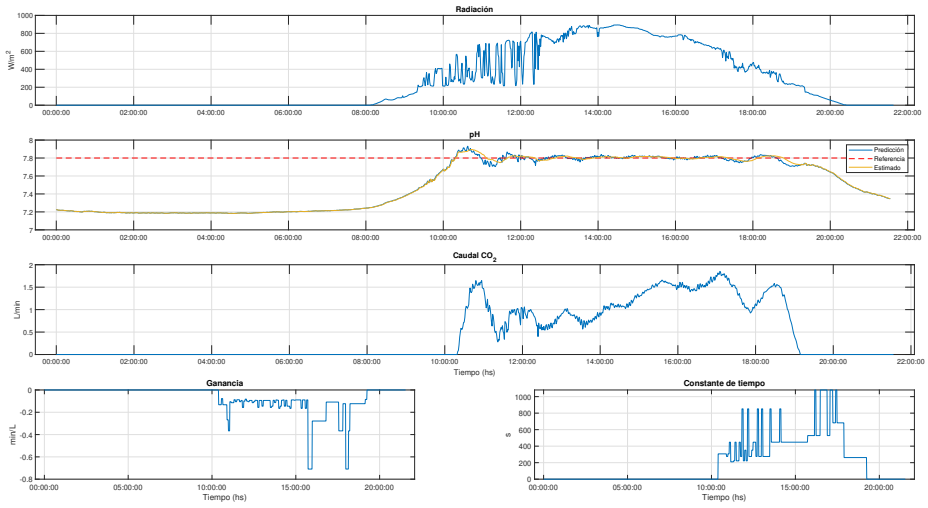


Figura 1.4: Implementacion del estimador de pH y controlador de parámetros fijos en simulación

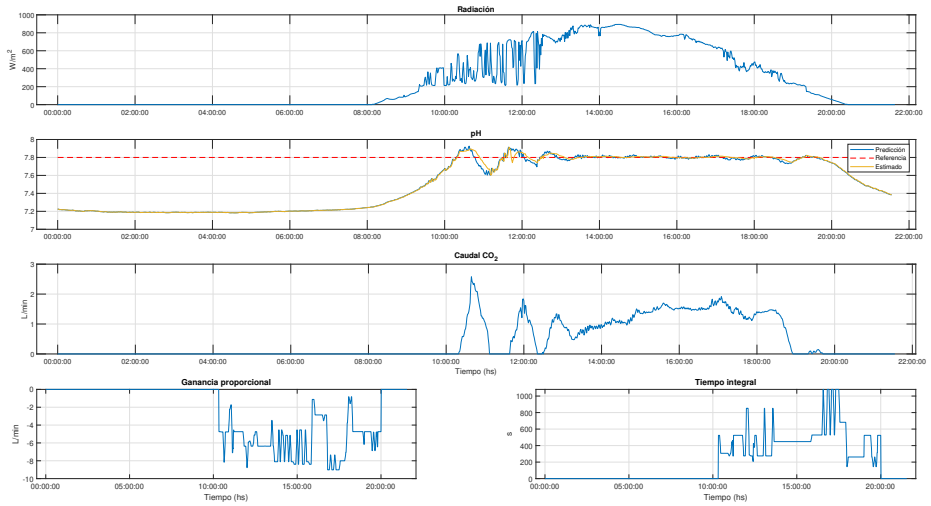


Figura 1.5: Implementacion del estimador de pH y controlador de parámetros adaptativos en simulación

	Control					Modelado
Control	**IAE**	**ISE**	**ITAE**	**ITSE**	**EC**	**RMSE**
Fijo	18.4033	1.177	5.3643 e+03	343.0883	9.738	0.0208
Adaptativo	17.3405	1.3214	5.0548 e+03	185.1821	11.1577	0.0274

Tabla 1.2: Índices de comportamiento en simulación

Se puede observar, tanto gráfica como numéricamente, que el comportamiento de ambos controladores es similar y muy adecuado. Además, el funcionamiento del estimador es correcto tanto para el controlador de parámetros fijos como para el de parámetros adaptativos.

En las figuras 1.6 y 1.7 se muestra el comportamiento del estimador de pH cuando el sistema se encuentra controlado mediante un controlador PI de parámetros fijos. Se puede observar que es necesario realizar un esfuerzo por obtener una mejor estimación en casos como estos, pues el pH estimado nunca alcanza los valores máximos y mínimos del pH medido en el reactor. Esto se debe, probablemente, a la falta de datos obtenidos en condiciones de radiación y temperatura tan altas como las que se presentaron los días de implementación del estimador en el reactor.

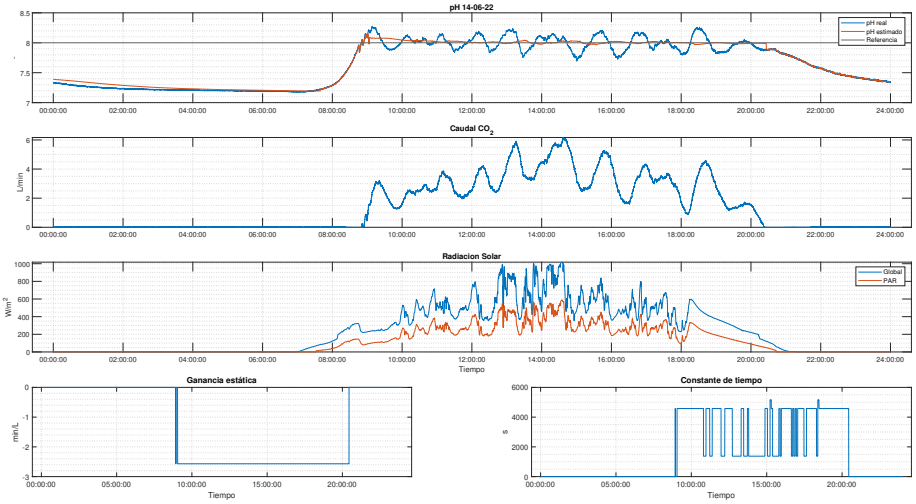


Figura 1.6: Control fijo y estimación de pH en reactor del Convenio UAL-IFAPA - 14 de junio

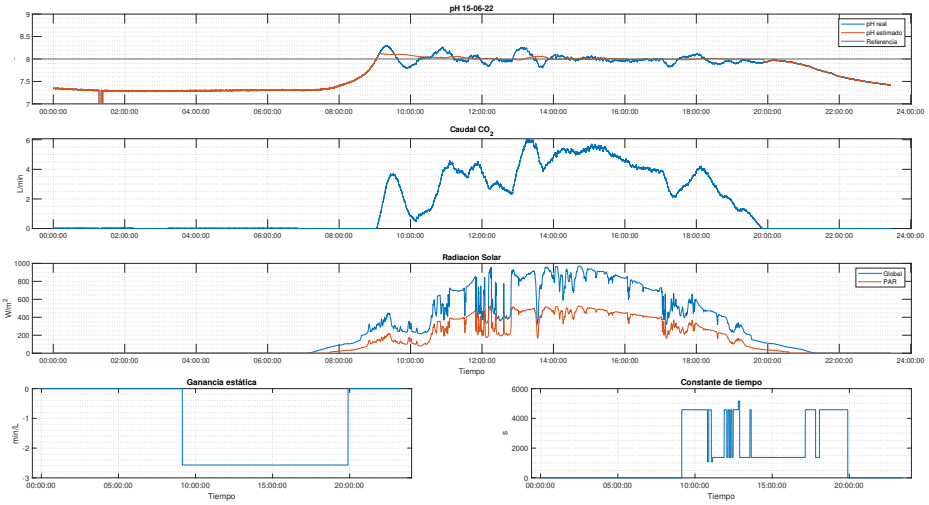


Figura 1.7: Control fijo y estimación de pH en reactor del Convenio UAL-IFAPA - 15 de junio

Por otro lado, los resultados del estimador de pH en combinación con el control adaptativo han sido muy satisfactorios. En las figuras 1.8 y 1.9 se muestran resultados de la implementación de dichos algoritmos en los reactores disponibles en las instalaciones del Convenio UAL-IFAPA.

A simple vista, si se comparan ambos tipos de controles se puede observar una cantidad de oscilaciones considerablemente mayor en el caso del controlador PI fijo. Además, el comportamiento del estimador de pH mejora considerablemente al implementar un controlador cuyos parámetros se ajusten a los parámetros del modelo estimados en cada instante.

En la tabla 1.3 se muestran los índices de comportamiento clásicos para evaluar cuantitativamente el comportamiento del controlador tanto de parámetros fijos como adaptativos. Además, en la tabla 1.4 se observa el valor del $RMSE$ obtenido en el estimador de pH para los días mostrados anteriormente, tanto para el control PI fijo como adaptativo.

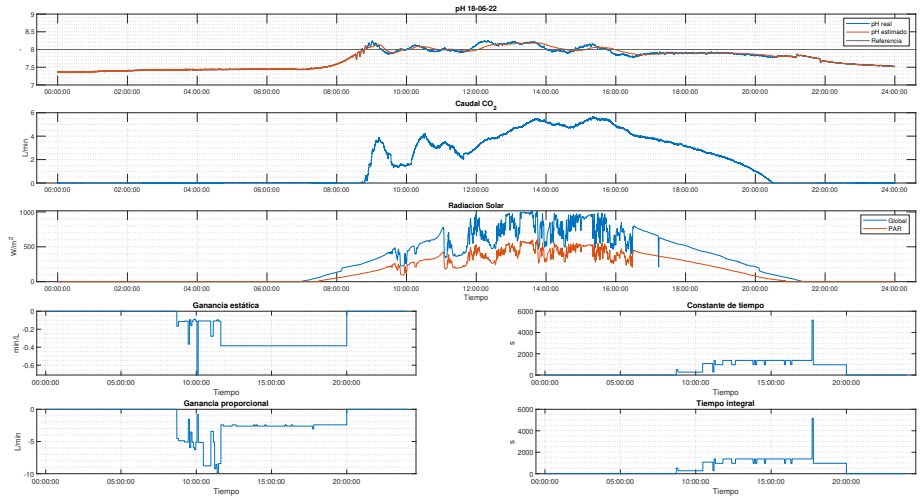


Figura 1.8: Control adaptativo y estimación de pH en reactor del Convenio UAL-IFAPA - 18 de junio

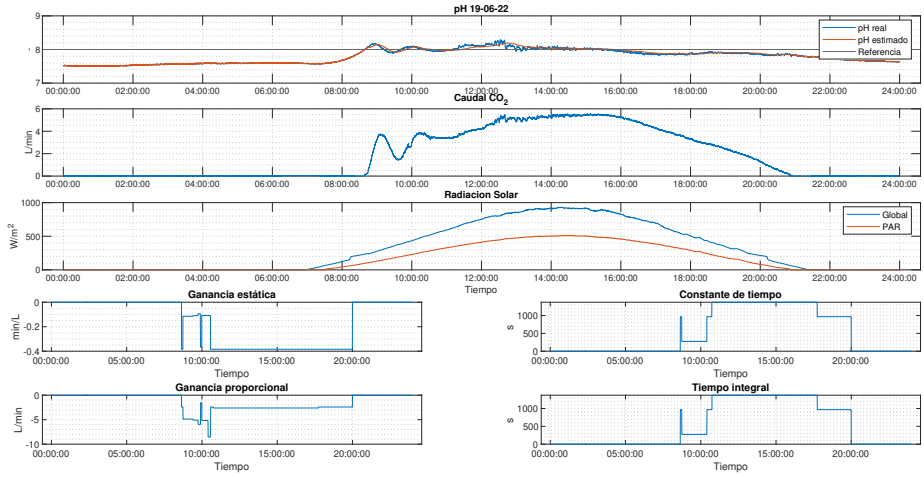


Figura 1.9: Control adaptativo y estimación de pH en reactor del Convenio UAL-IFAPA - 19 de junio

Se observa que los índices de comportamiento de control son similares para ambos casos, aunque gráficamente se puede ver que el control adaptativo presenta menos oscilaciones. Por otro lado, el error cuadrático medio del pH modelado es considerablemente menor al aplicar el control de parámetros adaptativos. Sin embargo, para que la comparación de los índices fuera preciso, deberían implementarse los dos controladores el mismo día en dos reactores sometidos a las mismas condiciones.

	Control				
Día	**IAE**	**ISE**	**ITAE**	**ITSE**	**EC**
Control de parámetros fijos					
14 de junio	4.0102 e+03	589.4044	8.2974 e+07	1.2525 e+07	525.8072
15 de junio	2.9245 e+03	382.2602	4.8215 e+07	4.8641 e+06	457.0744
Control de parámetros adaptativos					
18 de junio	4.1221 e+03	558.9979	9.0100 e+07	1.1964 e+07	510.5508
19 de junio	3.3471 e+03	389.7199	7.5293 e+07	8.8794 e+06	534.3708

Tabla 1.3: Índices de comportamiento - Controlador PI en reactor del Convenio UAL-IFAPA

	Modelado
Día	**RMSE**
Control de parámetros fijos	
14 de junio	0.1275
15 de junio	0.1
Control de parámetros adaptativos	
18 de junio	0.0618
19 de junio	0.0451

Tabla 1.4: Índices de comportamiento - Estimador de pH en reactor del Convenio UAL-IFAPA

Los resultados principales de este Trabajo Fin de Grado han sido enviados a las XLIII Jornadas de Automática [3].

2

Materiales y métodos

2.1. Reactor Raceway

2.1.1. Descripción física y dimensiones

Durante el desarrollo del presente trabajo, se ha utilizado el fotobiorreactor *Raceway* localizado en el centro IFAPA de la Junta de Andalucía que se encuentra anexo a la UAL (figura 2.1). Se dispone de dos de ellos, uno utiliza agua limpia y un aporte externo de nutrientes, mientras que el segundo utiliza aguas residuales provenientes de la UAL. En el presente trabajo se utilizará el primero de ellos, aunque la descripción realizada a continuación aplica a ambos.

Figura 2.1: Reactores *Raceway* disponibles en el centro IFAPA

El reactor ha sido utilizado tanto para la obtención de modelos de pH como para la implementación del estimador de dicha variable y de algoritmos de control para mantener el pH en su valor óptimo. Por lo tanto, el proyecto incluye no solo resultados obtenidos en simulación sino que, gracias a la disponibilidad de las instalaciones, ha sido posible realizar ensayos en una planta real de dimensiones muy similares a las de una planta industrial.

Los dos reactores presentan las mismas características físicas (figura 2.2). Están compuestos por dos canales de 40 m de longitud y 1 m de ancho, con una profundidad de 30 cm. Ambos canales se unen por un acople en forma de U en sus extremos. Por lo tanto, cada reactor cuenta con una superficie total de 80m^2, lo cual proporciona una relación entre el volumen y el nivel del medio de 800 L/cm. A pesar de contar con una profundidad de 30 cm, la altura de cultivo óptima se encuentra alrededor de los 15 cm.

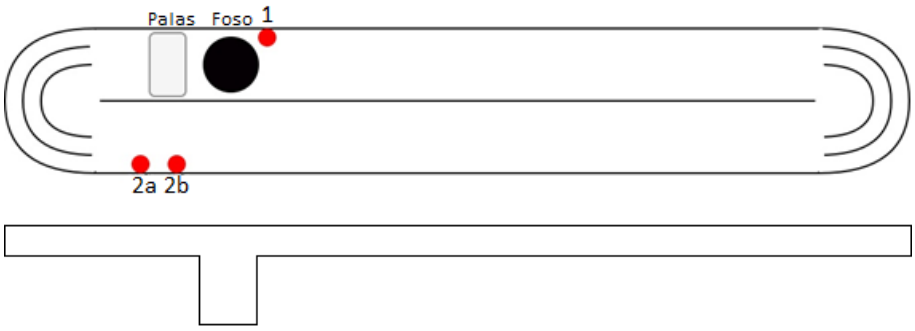


Figura 2.2: Esquema de los fotobiorreactores *Raceway*

Los reactores cuentan con tres puntos de medidas ((1, 2a y 2b, figura 2.2), siendo dos de ellos redundantes. Los sensores que se utilizan en dichos puntos se describen en la sección 2.1.3.

El mezclado e impulsión del medio se realiza mediante una rueda de paletas de aluminio, con un diámetro de 1.2 metros y formada por 8 palas. La misma es accionada por un grupo motorreductor formado por el motor (W12 35kW, 1550 rpm, Ebarba, Barcelona, España) y el reductor (WEB Iberica S.A., Barcelona, España). El grupo motorreductor es controlado eléctricamente por medio de un variador de frecuencia, de modo que se pueda proporcionar al fluido una velocidad constante de 0.2 m/s.

En el foso que se encuentra justo detrás de las palas se realiza la inyección de aire y de CO_2, a través de 3 discos difusores (AFD 270,EcoTec,España). El mismo se encuentra a 1.8 m de distancia de las palas impulsoras y presenta forma cilíndrica con un diámetro 65 cm y 1 m de profundidad.

2.1.2. Cepa de microalgas

Los reactores se encuentran inoculados con microalgas de la cepa *Scenedesmus*, la cual se caracteriza por su adaptabilidad y su buen crecimiento en reactores de exterior. Su resistencia a los contaminantes la convierte en una excelente cepa para el tratamiento de aguas residuales. Su pH óptimo se encuentra en el rango de 7.0 a 9.0 y la máxima actividad fotosintética a una temperatura en el rango comprendido entre 26 y 34 ºC [1].

2.1.3. Sensores y actuadores

El fotobiorreactor que se ha utilizado cuenta con diversos sensores y actuadores distribuidos a lo largo del mismo, los cuales proporcionan información sobre el estado del cultivo y permiten actuar sobre el mismo. A continuación, se realiza una descripción detallada de cada elemento.

- **Sensores**
  Los sensores de pH, temperatura y OD que se describen a continuación se encuentran distribuidos a lo largo del reactor como se muestra en la figura 2.2. Hay tres puntos de lectura de cada parámetro: el primero se encuentra justo después del foso donde se realiza la inyección, mientras que los otros dos se encuentran al final del canal, antes de las palas. Estos últimos se encuentran situados uno al lado del otro con el fin de contar con una medida redundante y poder comprobar el correcto funcionamiento de las sondas.

  1. **Sensores de pH**
     Obtienen una lectura del pH del medio y la cuantifican en potencial eléctrico. Están formadas por dos electrodos, uno de los cuales actúa como referencia. El segundo electrodo es aquel que mide y se compone de una membrana de vidrio sensible a las concentraciones de iones de hidrógeno en el medio. El pH del medio se puede conocer midiendo la diferencia de potencial entre ambos electrodos. La señal que se obtiene de los electrodos es acondicionada, amplificada y calibrada, para poder obtener una medida fiable. El sensor utilizado en el reactor ofrece un rango a la salida de 4 a 20 mA.

Los sensores instalados son de la marca Crison 50 10 T (figura 2.3) e incluyen un sensor de temperatura RTD (PT1000). Son capaces de medir un pH entre 0 y 14, soportan una temperatura de trabajo comprendida en el rango de 0 a 80 ºC, son de bajo mantenimiento y debe controlarse y rellenarse periódicamente con electrolíticos. Son muy adecuados para aguas residuales o con partículas en suspensión, por lo cual son óptimos para los reactores estudiados.

Figura 2.3: Sensor Crison 50 10 T - Fuente: `https://www.crisoninstruments.com/es`

2. **Sensores de temperatura**
   Se recogen medidas de dos temperaturas distintas, ambos con termoresistencias (RTD) (figura 2.4). Por un lado, la temperatura del suelo donde se encuentra el reactor es medida con termo-resistencias tipo PT100. Por el otro, la temperatura del medio es medida con el sensor PT1000 incluida en las sondas de pH descritas anteriormente.

   El funcionamiento de las termoresistencias RTD (del inglés *Resistance Temperature Detector*) se basa en la variación de la resistencia provocada por la variación de temperatura. Es decir, al aumentar o disminuir la temperatura, aumenta o disminuye la resistencia, respectivamente. Esto es debido a que, a mayor temperatura hay una mayor agitación térmica, por lo que se dispersan más los electrones y se reduce su velocidad media, aumentando así la resistencia.

La única diferencia entre la PT100 y la PT1000 es la resistencia que presentan a una temperatura de 0ºC, siendo de 100Ω para la PT100 y 1000Ω para la PT1000. Ambas están fabricadas con platino, son capaces de medir altos rangos de temperatura y se caracterizan por su exactitud, su estabilidad y su linealidad.

Figura 2.4: Termoresistencia (RTD) - Fuente: `https://es.rs-online.com/web/`

3. **Sensores de oxígeno disuelto**

   Los sensores de oxígeno disuelto recogen medidas de la cantidad de oxígeno gaseoso disuelto en el agua y la traducen a una señal eléctrica.

   Los sensores utilizados son de la marca Mettler Toledo, modelo inPro 6050/120 (figura 2.5). Se trata de sensores polarográficos que utilizan una membrana permeable al oxígeno que cubre dos electrodos (ánodo y cátodo). La concentración de oxígeno tiene una influencia directa sobre el flujo de electrones, por lo que la diferencia de potencial entre ambos electrodos permite conocer la concentración del oxígeno disuelto en el medio.

   El modelo utilizado es muy recomendado en aplicaciones de agua que incluyen el tratamiento de aguas residuales y requiere muy poco mantenimiento. La membrana del sensor es muy duradera y fácil de sustituir y mantener.

Figura 2.5: Sensor Mettler Toledo inPro 6050/120 - Fuente: `https://www.mt.com/es`

4. **Sensor de ultrasonido**

   Este es utilizado para medir la altura de medio acuoso que hay en el reactor y se coloca en el punto 2b (figura 2.2) junto con los dos sensores de pH y OD (oxígeno disuelto). Se encuentra ubicado en dicho sitio ya que, al encontrarse justo antes de las palas y al final del recorrido, el nivel del medio no se ve afectado por las turbulencias que genera la pala. El sensor con el que se trabaja es de la marca Wenglor, modelo UMD402U035 (figura 2.6).

   El funcionamiento de los sensores ultrasónicos se basa en principios acústicos: emiten pulsaciones acústicas a través del aire y temporiza el tiempo que tarda en recibir el eco. Conociendo la velocidad de la onda y la velocidad del sonido se consigue conocer la distancia al objeto. El sensor obtiene la distancia entre él mismo y el medio, por lo que luego se realiza una conversión mediante una recta de calibración para conocer la altura del medio, medida en centímetros.

Figura 2.6: Sensor Wenglor UMD402U035 - Fuente: `https://www.wenglor.com/es/`

5. **Caudalímetros**

   Los caudalímetros permiten conocer el caudal (en L/min) tanto de CO_2 como de aire que se está inyectando en el foso. Ambos caudalímetros instalados son de la marca SMC, pero sus modelos varían debido a que los volúmenes de caudal a medir en cada caso son muy distintos.

   El principio de funcionamiento se basa en una pieza giratoria colocada en su interior cuya velocidad de giro es proporcional al caudal que circula. Dicha pieza está hecha de resina y una aleación ferromagnética, de la cual se hace la lectura aprovechando el efecto hall. La señal obtenida es acondicionada, de modo que el caudal medido se traduce a una corriente comprendida entre los 4 y los 20 mA.

   El sensor utilizado para la medición de caudal de CO_2 es el modelo PF2M7 (figura 2.7a), que permite un caudal de hasta 25 L/min, mientras que para el caudal de aire se utiliza el modelo PFMB (figura 2.7b), ya que permite un caudal máximo de 500 L/min.

(a) Modelo PF2M7 **(b)** Modelo PFMB

Figura 2.7: Caudalímetros SMC - Fuente: `https://www.smc.eu/es`

- **Actuadores**

1. **Electroválvula proporcional**

   Las electroválvulas proporcionales brindan un caudal dependiente de una señal de entrada, que puede ser tanto corriente como tensión. Esto permite realizar controles muy precisos del caudal, puesto que brindan la posibilidad de diseñar algoritmos de control que no sean todo/nada.

El reactor dispone de dos electroválvulas que permiten implementar controladores tipo PID (controlador Proporcional, Integral y Derivativo) para controlar el pH (mediante inyección de CO_2) y el oxígeno disuelto (mediante inyección de aire).

La válvula de CO_2 es una válvula piezoeléctrica de la marca FESTO, modelo VEMD (figura 2.8a). En estas válvulas se aprovecha el comportamiento piezoeléctrico de determinados materiales cerámicos que modifican su forma mecánicamente al recibir tensión. El modelo utilizado puede brindar un caudal de hasta 20 L/min y, gracias a la incorporación de un sensor y una regulación electrónica, presenta un comportamiento lineal entre la tensión de control y el caudal de salida.

La válvula de aire, por otro lado, es de la marca Camozzi, modelo MX2-1/2 (figura 2.8b), que se caracteriza por combinar reguladores modulares MX2 con micro reguladores proporcionales electrónicos K8P para asegurar alta precisión en la regulación de presión, altos rangos de caudal y bajo consumo eléctrico. Brinda un caudal de hasta 6000 L/min.

(a) FESTO, VEMD

(b) Camozzi, MX2-1/2

Figura 2.8: Electroválvulas proporcionales - Fuente: `https://www.festo.com/es/es/`, `https://www.rowse-pneumatics.co.uk/`

2. **Bomba sumergible**

   La bomba sumergible se encuentra instalada en el foso con los fines de extraer medio del reactor y enviarlo a los tanques de cosechado y de bajar el nivel del medio. Es decir, con esta se realiza el cosechado diario y se controla el nivel. Se trata de una bomba de la marca ESPA, modelo VXV 1100AS (figura 2.9).

La bomba puede suministrar un caudal de hasta 12000 L/min con una diferencia de altura de 4 metros entre la bomba y la salida del medio en los tanques de cosechado (a mayor altura menor caudal es capaz de suministrar, con una diferencia de altura máxima de funcionamiento de 10 m). Además, es apta para aguas sucias y cargadas, característica que la convierte en una elección adecuada para el proceso en estudio.

Figura 2.9: Bomba ESPA, VXV 1100AS - Fuente: `https://www.espa.com/es/`

3. **Electroválvula de dilución**

   Esta válvula se utiliza para llenar el fotobiorreactor, ya sea con agua limpia (para el reactor al que se le agregan nutrientes diluidos) o aguas residuales provenientes de la UAL. Se utilizan para controlar el nivel del medio y llenar el reactor luego del cosechado. El modelo utilizado es el L10-J2 de la marca Cepex (figura 2.10), del tipo todo/nada.

Figura 2.10: Electroválvula Cepex, L10-J2 - Fuente: `https://www.cepex.com/`

2.1.4. Autómata programable (PLC)

Los sensores y actuadores descritos en la sección 2.1.3 van conectados a dos autómatas programables de la marca Schneider. El primero de ellos es el maestro, se trata del modelo TM241CE24T/U (figura 2.11a) y lleva el código de control de los reactores. Cuenta con cinco módulos de expansión (2 de salidas y 3 de entradas analógicas), necesarios para la toma datos. Cuenta con una batería interna, por lo que, en caso de una interrupción del suministro eléctrico, el código guardado en él prevalece y es capaz de continuar el control del reactor. El segundo es el modelo TM251MESE (figura 2.11b) y su función es enviar datos de lectura al maestro.

(a) TM241CE24T/U **(b)** TM251MESE

Figura 2.11: Autómatas programables *Schneider Electric* - Fuente: `https://www.se.com/es/es/`

Los módulos de salidas analógicas permiten conectar 4 sensores analógicos con entrada configurable en corriente (4 a 20 mA o 0 a 20 mA) o en tensión (-10 a 10 V o 0 a 10V), con 12 bits de resolución. Se trata del modelo TM3AQ4/G (figura 2.12a). Por otro lado, los módulos de entradas analógicas permiten conectar hasta 8 sensores analógicos con entrada configurable en corriente o en tensión (con las mismas opciones que las salidas) y 12 bits de resolución. El modelo utilizado es el TM3AI8/G (figura 2.12b).

(a) TM3AQ4/G **(b)** TM3AI8/G

Figura 2.12: Módulos de entradas y salidas analógicas - Fuente: `https://www.se.com/es/es/`

Los autómatas descritos anteriormente se encuentran conectados mediante Ethernet Industrial. El maestro, además, se conecta a un ordenador industrial IBOX-601 (figura 2.13) con procesador intel i5-6200U y 16 Gb de memoria RAM. En él se dispone de todos los parámetros de lectura y escritura presentes en los autómatas, gracias al uso del protocolo OPC (*OLE for Process Control*). Esto permite utilizarlos en un sistema SCADA (*Supervisory Control and Data Acquisition*) y en demás algoritmos de control desarrollados en MATLAB.

Figura 2.13: Ordenador industrial IBOX-601 - Fuente: `https://www.amazon.es/Fanless-Industrial-Rugged-Computer-Support/dp/B07QGQTPZ8`

La estructura de control y comunicación consta de 4 grandes bloques (MATLAB, SCADA, Python y PLC) comunicados mediante protocolo OPC (figura 2.14). Los valores de cada parámetro se leen del autómata programable en cada instante a través del protocolo OPC. Una vez leídas dichas variables, se registran en archivos de extensión .csv mediante un archivo en Python, y se leen dichas variables en la pantalla del SCADA, el cual se utiliza únicamente para visualización de datos y para cambiar ciertos parámetros específicos, tales como consignas de pH, OD, nivel, entre otros. Los controladores se encuentran implementados en un MATLAB *script* que se ejecuta instante a instante y, con los valores de las variables, calcula y envía la consigna de caudales (o la señal de control que aplique en cada caso). Las consignas de las señales de control son enviadas nuevamente al PLC mediante protocolo OPC.

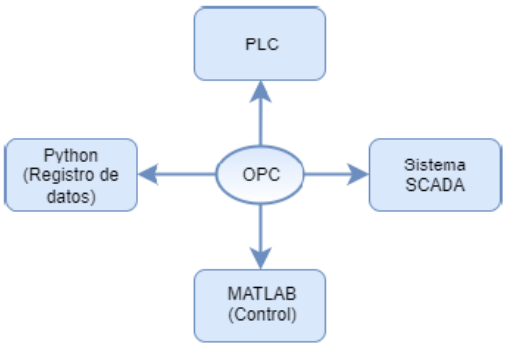


Figura 2.14: Esquema de comunicación entre dispositivos

Para realizar la programación de los controladores en el PLC se ha utilizado *EcoStruxure Machine Expert Logic Builder* de *Schneider Electric*. Se trata de un sistema de programación de controladores que, de acuerdo con la norma IEC 61131-3, admite todos los lenguajes de programación estándar [14].

La organización del proyecto se basa en la orientación a objetos. Un proyecto *EcoStruxure Machine Expert* contiene un programa de controlador compuesto por diversos objetos de programación, e incluye definiciones de los recursos necesarios para ejecutar instancias del programa (aplicaciones) en sistemas de destino (dispositivos, controladores) específicos [14]. Así hay dos tipos principales de objetos en un proyecto:

- **Objetos de programación (POU):** son programas, funcionas, bloques de funciones, métodos, interfaces, acciones, tipos de datos, definiciones, entre otros.
- **Objetos de recurso:** son los dispositivos.

Los lenguajes de programación admitidos son FBD (*Function Block Diagram*), LD (*Ladder Diagram*), IL (*Instruction List*), SFC (*Sequential Function Chart)*, ST (*Structured text*) y CFC (*Continuous Function Chart*) [14].

En este caso se hará uso de **SFC**, un lenguaje orientado gráficamente que describe el orden cronológico de acciones concreta en un programa. Estas acciones están disponibles como objetos de programación independientes, y están escritas en cualquier lenguaje de programación disponible. Las acciones se asignan a elementos de paso y los elementos de transición controlan la secuencia de procesamiento [14]. En la figura 2.15 se muestra un ejemplo de una secuencia de pasos en un módulo SFC.

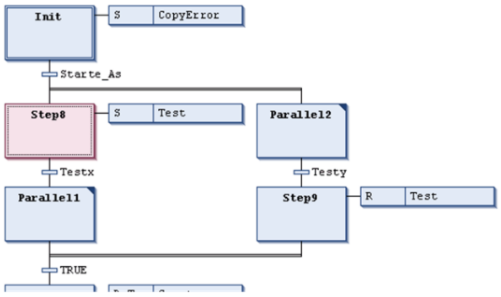


Figura 2.15: Ejemplo de una secuencia de pasos en un módulo SFC - Fuente: [14]

2.1.5. Estación meteorológica

Los datos sobre variables meteorológicas (temperatura, radiación, velocidad del viento, entre otros) se obtienen en tiempo real gracias a la estación meteorológica disponible en las instalaciones del Convenio UAL-IFAPA, que cuenta con los siguientes sensores, todos de la marca ONSET.

- **Sensor de temperatura y humedad relativa**

  Se utiliza el modelo S-THB-M008 (figura 2.16a), el cual puede leer temperaturas comprendidas en el rango de -40 a 75 ºC con una precisión de $\pm 0.21\ ^{\circ}C$ y humedades relativas entre el 10 y el 90 %, con una precisión de ± 2.5 %. Presenta una resolución de 0.02 ºC a 25 ºC y 0.1 % para la humedad relativa. A este sensor se le ha incorporado un protector de lluvia y radiación solar modelo RS3-B (figura 2.16b).

(a) S-THB-M008 **(b)** RS3-B

Figura 2.16: Sensor de temperatura y protector ONSET - Fuente: `https://www.onsetcomp.com/`

- **Sensor de radiación solar y luz PAR**

  El sensor de radiación solar es del modelo S-LIB-M003 (figura 2.17a) y presenta un rango de medición de 0 a 1280 W/m^2 en un rango espectral de 300 a 1100 nm, con una precisión de ± 10 W/m^2 o ± 5 %, aquel que sea mayor con la radiación solar. Además, a partir de los 25 ºC presenta un error adicional inducido por la temperatura de $\pm 0.38\ W/m^2{}^{\circ}C$.

  El sensor de luz PAR (luz fotosintética) es del modelo S-LIA-M003 (figura 2.17b) y puede medir en un rango de 0 a 2500 $umol/m^2s$ en longitudes de onda de 400 a 700 nm, con una precisión de ± 5 $umol/m^2s$ o ± 5 %, aquel que sea mayor con la luz solar. Además, el error adicional inducido por la temperatura en este caso es de $\pm 0.75\ umol/m^2s{}^{\circ}C$, a partir de lo 25 ºC.

(a) S-LIB-M003 **(b)** S-LIA-M003

Figura 2.17: Sensor de radiación solar y luz PAR ONSET - Fuente: `https://www.onsetcomp.com/`

- **Sensor de velocidad y dirección del viento**

  El sensor utilizado es del modelo S-WCF-M003, el cual puede medir velocidades de hasta 76 m/s, en una dirección entre los 0 y los 355º, con una precisión de ± 1.1 m/s o ± 5 % para la velocidad (el mayor de los dos) y de $\pm 7^{\circ}$ para la dirección. Presenta una resolución de 0.5 m/s para la velocidad y de 1º para la dirección (figura 2.18).

Figura 2.18: Sensor ONSET, S-WCF-M003 - Fuente: `https://www.onsetcomp.com/`

- **Registro de datos**

  La información proporcionada por los sensores es recopilada por un dispositivo comunmente conocido como ***Datalogger***, el modelo utilizado es el HOBO RW2102 (figura 2.19). Este recopila la información, la procesa, la almacena y la envía mediante HOBOlink (servicio en la nube propio de OMSET). El modelo utilizado presenta un tamaño compacto y es de fácil implementación. Hay dos posibilidades de alimentación: mediante panel solar o mediante baterías de litio recargables por el usuario. Se puede elegir un tiempo de muestreo mínimo de 1 minuto y máximo de 18 horas.

Figura 2.19: Datalogger ONSET, HOBO RW 2102 - Fuente: `https://www.onsetcomp.com/`

2.2. Representaciones del sistema

El pH es, además del OD, una de las variables que pueden ser controladas y que poseen una fuerte dependencia con la velocidad de fotosíntesis y, por tanto, con la producción de biomasa [10]. Es muy importante que se consigan modelos lo suficientemente representativos de su dinámica, pero a su vez sencillos, para poder desarrollar algoritmos de control que consigan mantener esta variable cerca de su valor óptimo.

El control del pH es un problema no lineal que puede ser linealizado en ciertas circunstancias y su regulación se consigue con la inyección de CO_2 [2]. Un modelo simple que captura la dinámica del pH es el mostrado en las siguientes ecuaciones:

$$pH(s) = \frac{K_1}{(1+\tau s)} \frac{K_2 w_n^2}{(s^2 + 2\delta w_n s + w_n^2)} e^{-t_r s} CO_2 + \frac{K_r}{(1+\tau_r s)} I_{sat} \tag{2.1}$$

$$pH(s) = TF_1(s) TF_2(s) CO_2(s) + TF_3(s) I_{sat}(s) \tag{2.2}$$

donde pH es el pH del medio, CO_2 es el porcentaje de apertura de la válvula e I_{sat} un valor saturado de radiación global [2].

Como se puede observar en la ecuación 2.1, la dinámica del pH con respecto al CO_2 viene dada por un término de primer orden con retardo ($TF_1(s)e^{-t_r s}$) que marca la dinámica dominante del proceso, junto a una función de transferencia de segundo orden ($TF_2(s)$) que representa las oscilaciones existentes en el sistema debido a las inyecciones de CO_2 que se van atenuando por la recirculación del medio a lo largo del receptor solar. La función $TF_3(s)$ representa el efecto sobreamortiguado de la radiación solar sobre el pH como un efecto la fotosíntesis. Los parámetros de las funciones toman valores particulares dependiendo del tipo de reactor, de la cepa y de las condiciones meteorológicas [10].

Es por esto último que este tipo de modelos debe ser constantemente recalibrados, pues un cambio en las condiciones ambientales (radiación, temperatura, entre otros) produce una variación en los parámetros del modelo, lo cual produce que un controlador diseñado bajo ciertas condiciones no consiga buenos resultados cuando estas varíen.

Como se ha mencionado en la sección 1.3, el objetivo del presente trabajo fin de grado es hallar modelos simplificados que relacionen el pH del cultivo con la inyección del CO_2 y con la fotosíntesis de los microorganismos, teniendo en cuenta las condiciones a las que se encuentra sometido el medio.

Con lo expuesto anteriormente y como mejora del modelo 2.1, se ha decidido separar la dinámica del pH en dos modelos simples: por un lado, una función de transferencia que relacione el pH con la inyección de CO_2 y, por otro, una descripción en espacio de estados que represente el aumento del pH por la fotosíntesis al cesar la inyección de gas.

2.2.1. Modelo de función de transferencia

Una función de transferencia es la relación que existe entre la transformada de Laplace de la salida con respecto a la transformada de Laplace de la entrada con condiciones iniciales nulas (las variables que relaciona son variables de desviación respecto a un determinado punto de operación), se trata de una **descripción externa** del sistema, pues es una relación explícita directa entre las señales de entrada y de salida.

Para reflejar la influencia de la inyección de CO_2 sobre el pH se utilizará una función de transferencia de primer orden con retardo, cuya forma genérica es la mostrada en la siguiente ecuación:

$$Y(s) = \frac{K}{\tau s + 1} e^{-t_r s} U(s) \tag{2.3}$$

donde Y(s) es la salida (pH), U(s) es la entrada (CO_2), K es la ganancia estática, τ es la constante de tiempo y t_r es el tiempo de retardo. El significado de los parámetros del modelo (K, τ y t_r) es el siguiente:

- **Ganancia estática:**

  Es la relación entre la amplitud de la variación de CO_2 y la amplitud de la variación de pH, es decir, es la relación entre la razón de cambio de la salida (pH) y la razón de cambio de la entrada (CO_2), una vez que la salida se ha estabilizado tras un cambio en forma de escalón en la entrada ($K = \Delta Y / \Delta U$).

- **Constante de tiempo:**

  Es un parámetro indicador de la velocidad de respuesta. Representa el tiempo que tarda la salida en alcanzar el 63 % de su cambio, medido a partir del instante en que el sistema comienza a reaccionar.

- **Tiempo de retardo:**

  Se define como el tiempo transcurrido entre que se produce un cambio en la entrada y la salida comienza a reaccionar.

Para una caso genérico, en la figura 2.20 se muestran gráficamente los conceptos explicados anteriormente.

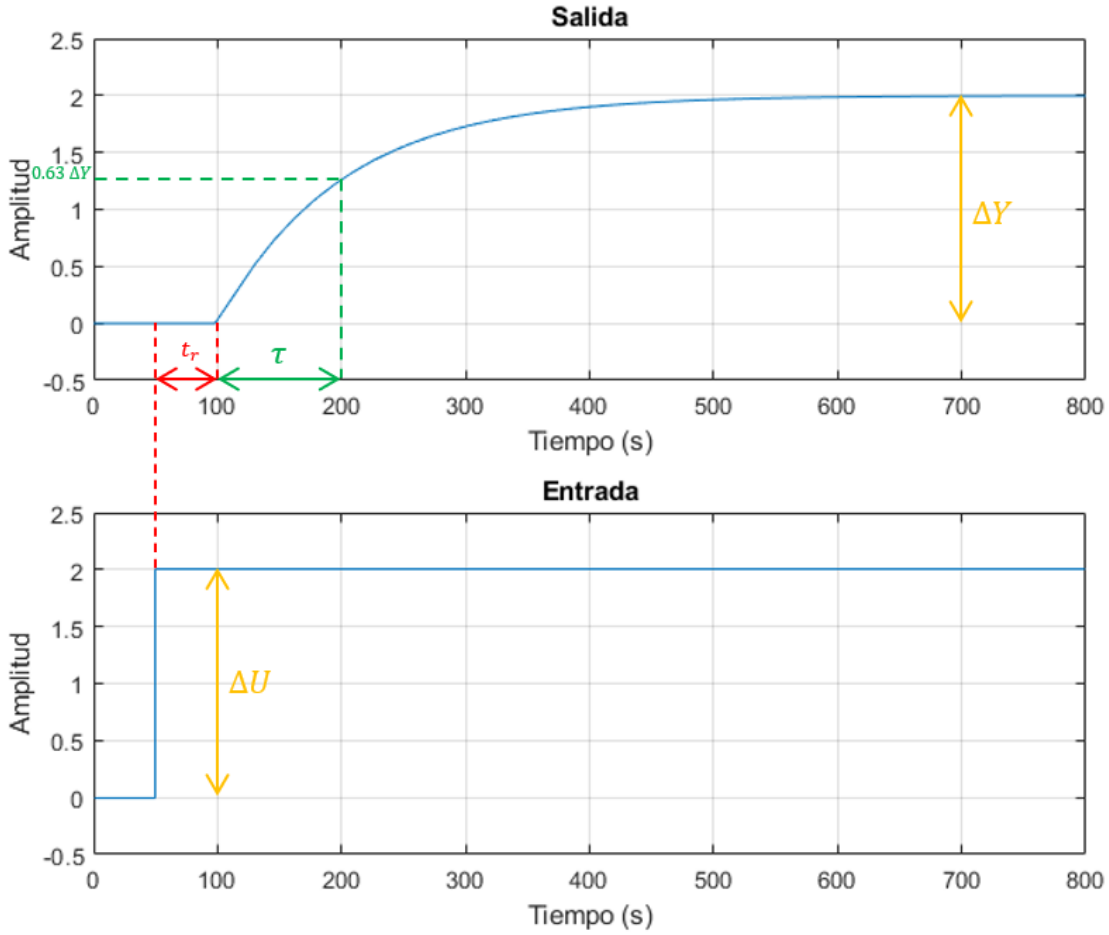


Figura 2.20: Parámetros de una función de transferencia de primer orden con retardo

En el caso específico de la relación entre el pH y el CO_2, un aumento en la inyección de CO_2 produce una disminución del pH, por lo que la ganancia estática será negativa. El tiempo de retardo es fijo, pues es un parámetro puramente físico y depende de las dimensiones del reactor y de la velocidad a la que se desplaza el fluido, cuyo valor es constante en este caso (si se variara la velocidad del fluido cambiaría el tiempo de retardo). En el caso específico del sistema estudiado durante el desarrollo del presente trabajo, es el tiempo que tarda una masa de cultivo en desplazarse desde el foso (donde se realiza la inyección del gas) hasta el punto de medida, en este caso, el sensor 2b (figura 2.2). Por último, el valor de la ganancia estática y de la constante de tiempo dependerá de parámetros como la radiación solar, la temperatura del medio, el nivel del medio, la concentración de biomasa, entre otros.

2.2.2. Descripción en espacio de estados

Una descripción en espacio de estados es una descripción interna, ya que es una relación explícita indirecta entre las señales de entrada y de salida. Estas dos variables se relacionan a través de una variable $x(t)$ llamada **estado del sistema dinámico**. Un estado es el conjunto mínimo de variables de modo que, si se conocen tanto su valor en un instante t_0 como las entradas para tiempos posteriores a t_0, se puede determinar completamente el comportamiento del sistema para cualquier tiempo futuro.

Una descripción en espacio de estados genérica es de la siguiente forma:

$$\dot{x}(t) = \frac{dx(t)}{dt} = Ax(t) + Bu(t) \tag{2.4}$$

$$y(t) = Cx(t) + Du(t) \tag{2.5}$$

donde A es la matriz dinámica, B es la matriz de control, C es la matriz de sensor y D es el término directo. Además, la ecuación (2.4) es la ecuación de estado y la ecuación (2.5) es la ecuación de salida.

La matriz dinámica (A) será cuadrada de orden n, mientras que la matriz sensor (C) será de orden $n \times 1$, siendo n el orden del sistema. Además, como esta representación se utilizará para modelar la dinámica del pH al cesar la inyección de CO_2, el sistema no contará con ninguna entrada, por lo que $u(t) = 0$ y el modelo obtenido no tendrá las matrices B y D. Por lo tanto, el modelo en espacio de estados obtenido (en tiempo continuo) será de la forma:

$$\dot{x}(t) = Ax(t) \tag{2.6}$$

$$y(t) = Cx(t) \tag{2.7}$$

Donde la dimensión de A dependerá del orden escogido y los valores de sus elementos variarán según la propia forma del modelo escogido. Es decir, para un modelo de orden 2, los elementos de A variarán según se elija un modelo de segundo orden puro (dos constantes de tiempo) o con integrador (una constante de tiempo y un integrador).

En el caso de obtener modelos de segundo orden, los polos serán reales, de modo que se obtenga una dinámica sobreamortiguada. La respuesta libre presenta oscilaciones debidas al tiempo de residencia del fluido en el lazo que no se pretenden modelar, pues complicarían el diseño de los algoritmos de control y no proporcionarían mejoras en términos de crecimiento de microalgas. Es por esto que se evitarán modelos con polos complejos conjugados que brinden una respuesta subamortiguada.

2.3. Árboles de regresión

2.3.1. Conceptos generales

Como se ha mencionado en la sección 1.2, el principal objetivo del presente trabajo es conseguir adaptar los parámetros del modelo a las condiciones ambientales y del sistema (radiación solar, temperatura del medio, nivel, entre otros). Con dicho fin, se utilizarán árboles de regresión que, entrenándolos con una gran cantidad de datos, sea capaz de estimar los parámetros instante a instante en función de los valores de los parámetros de estimación.

Los **árboles de clasificación y regresión** (CART) son una técnica de aprendizaje supervisado que predice valores de respuestas mediante el aprendizaje de reglas de decisión derivadas de características. Se pueden utilizar tanto en una regresión como en un contexto de clasificación [9]. También se puede definir como un *grafo conexo acíclico dirigido formado por un conjunto finito de nodos conectados* [4]. Su representación gráfica toma la forma de la figura 2.21.

La unidad principal de los árboles son los nodos (cuadrados de la figura 2.21). Dicha unidad puede tener varios **nodos hijos** (nodos a los que apuntan, se encuentran por debajo) y se llama **nodo padre** si tiene al menos un nodo hijo. Si varios nodos comparten un mismo nodo padre, entonces se trata de **hermanos**. Al nodo sin antecesores (el que se encuentra en la parte superior) se le conoce como **nodo raíz** y aquellos sin sucesores (los que se encuentran en la zona inferior del diagrama) como **nodos terminales** u hoja.

Además, el camino desde la raíz a la hoja se llama **rama**. El **grado** de un nodo es el número de descendientes directos que posee y su **nivel** el número de arcos que deben recorrerse desde la raíz, a la cual se le presupone nivel 1. El **grado de un árbol** es el máximo de los grados de sus nodos y su **altura** el máximo de los niveles [4].

Aplicados los conceptos explicados anteriormente al árbol genérico de la figura 2.21 , se obtiene que:

- E es el nodo raíz y es el nodo padre de F, G y H. Como tiene tres descendientes directos, presenta un grado 3 y, al tratarse del nodo raíz, su nivel es el 1 .
- F es un nodo hijo de E, es hermano de G y H y es nodo padre de I y J. Como tiene dos descendientes directos es de grado 2 y es de nivel 2, pues hay que recorrer una rama desde el nodo raíz (que es nivel 1).
- G es nodo hijo de E, hermano de F y H y padre de K. Es de grado 1 y de nivel 2.
- H es nodo hijo de E, hermano de F y G y nodo padre de L y M. Es de grado 2 y de nivel 2.
- I y J son nodos hijos de F y son hermanos entre sí. Se trata de nodos terminales y presentan un nivel 3.
- K es nodo hijo de G, sin hermanos. Es un nodo terminal de nivel 3.
- L y M son nodos hermanos, ambos hijos del nodo H. Se trata de nodos terminales y son de nivel 3.
- El grado del árbol es 3 y su altura es de 3.

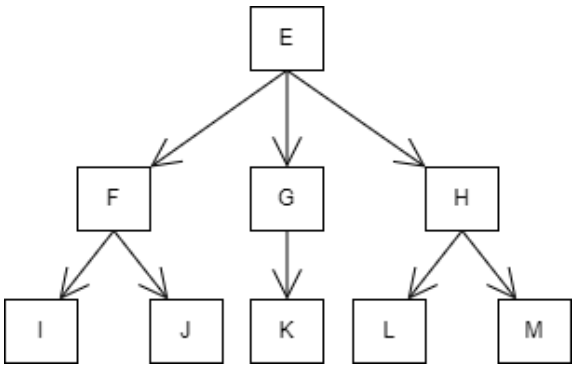


Figura 2.21: Esquema de un árbol de regresión

Cada nodo no terminal representa una proposición lógica acerca de una variable predictora y cada camino a una hoja es una región diferente del espacio que resulta de la intersección de la secuencia de las proposiciones lógicas. Es decir, el espacio predictor se divide en regiones simples que incluyen observaciones con valores similares para la variable respuesta [4].

Para hacer la predicción dentro de cada hoja se utiliza la media de los datos de entrenamiento que se encuentran en la misma región de predictores [4].

Aplicado al caso específico que atañe al presente trabajo, los nodos no hojas serán proposiciones lógicas acerca de variables como radiación solar, temperatura del medio u otras, mientras que los nodos hojas serán los valores estimados de los parámetros de los modelos.

2.3.2. Algoritmo de construcción

Para construir los árboles de regresión se aplica el siguiente algoritmo [4]:

- **Datos:** para entrenar el árbol se deben brindar como datos de entrada el vector de la variable dependiente "Y" (la que se desea estimar), la matriz de variables predictoras "X", el criterio de parada y el criterio de nodo terminal u hoja.
- **Inicialización:** el nodo raíz representa todo el espacio de predictores X. A él se le asigna la media de los valores de Y y se debe calcular la suma total de errores cuadráticos, que se define como:

$$S(T) = \sum_{l=\widetilde{T}} \sum_{i \in l} (y_i - m_l)^2 \tag{2.8}$$

  siendo l una hoja del árbol, $\widetilde{T}$ el conjunto de hojas del árbol T, y_i los valores de la variable dependiente y m_l la media en la hoja l.

- **Paso 1:** se calcula $S(T)$ para todos los posibles cortes c de los nodos no terminales, si todos los nodos son terminales se ha llegado al final del algoritmo. Un corte en c es una proposición lógica del tipo $X_j \leq c$, siendo c una constante en el dominio de los reales.
- **Paso 2:** se escoge el corte que arroje menor S.
- **Paso 3:** si se cumple el criterio de parada, el último nodo será el nodo terminal y se vuelve al paso 1. En caso contrario, se añaden dos nodos no terminales y se vuelve al paso 1.

Se observa que se trata de un algoritmo que toma decisiones localmente óptimas (busca el mejor corte) a cada paso, y avanza en sentido descendente, pues comienza en el nodo raíz y divide sucesivamente el espacio de predictores X. El **criterio de parada** suele establecerse como un umbral δ que es la mejora mínima que se exige cada vez que se realiza un corte. Además, el **criterio de nodo terminal** suele exigir que cada nodo contenga un número mínimo de datos, pues si las muestras son muy pequeñas la precisión predictiva del modelo empeora (el $S(T)$ de una hoja con un solo dato siempre será cero). Si existen nodos con muy pocos datos asociados, se dice que hay *overfitting* o que el árbol está **sobreentrenado** [4].

2.4. Toolbox MATLAB

Para la obtención de los modelos explicados en la sección 2.2 y, posteriormente, de los árboles de regresión explicados en la sección 2.3, se han utilizado diversas aplicaciones disponibles en el entorno de programación de MATLAB, conocidas como *MATLAB Toolboxes*.

2.4.1. System Identification Toolbox

System Identification Toolbox es una aplicación para construir modelos matemáticos de sistemas dinámicos a partir de datos de entrada y salida medidos. Los datos para realizar la estimación pueden estar en el dominio de la frecuencia o en el dominio del tiempo, y con ellos se pueden estimar funciones de transferencia en tiempo continuo o discreto, modelos de proceso, modelos de espacio de estados, entre otros [12].

Previo a la estimación de modelos debe llevarse a cabo la preparación de los datos que, como se ha comentado, pueden estar en el dominio del tiempo y de la frecuencia. En el presente trabajo, se desarrollarán modelos para datos en el **dominio del tiempo**, por lo que en la presente sección se hará hincapié en ellos. Los datos pueden tener entradas y salidas únicas o múltiples, y pueden ser reales o complejos. Además, deben ser muestreados en instantes de tiempo discreto y uniformemente espaciados, de modo que se obtenga una secuencia de entradas *u(t)* y una secuencia de salidas *y(t)* asociada. Los datos permitidos de los cuales se hará uso son [12]:

- **Datos de entrada/salida (E/S) en el dominio del tiempo:** una o más variables de entrada y una o mas variables de salida, muestreadas a lo largo del tiempo. Se utilizarán para la relación entre el pH y el caudal de CO_2.

- **Datos de series temporales:** se trata de conjuntos de datos con una o más salidas pero ninguna entrada. Se utilizarán para modelar la respuesta libre del sistema, que relaciona el comportamiento del pH con la fotosíntesis producida.

Antes de entrar en materia de identificación de sistemas debe tenerse en cuenta que hay que seguir las siguientes etapas:

1. Diseño y ejecución de experimentos.
2. Procesamiento de datos.
3. Selección de la estructura del modelo.
4. Estimación de parámetros.
5. Validación.

Para llevar a cabo los experimentos para la obtención de datos, debe tenerse un muy buen conocimiento previo del sistema que se desea estudiar, pues los ensayos varían enormemente de un sistema a otro. Los experimentos desarrollados en este trabajo se explican en la sección 3.1. Una vez obtenidos los datos, se cargan en el *System Identification Toolbox* indicando la entrada (en caso de haberla) y la salida del sistema, el tiempo de inicio y el tiempo de muestreo, para casos de datos en el dominio del tiempo (figura 2.22).

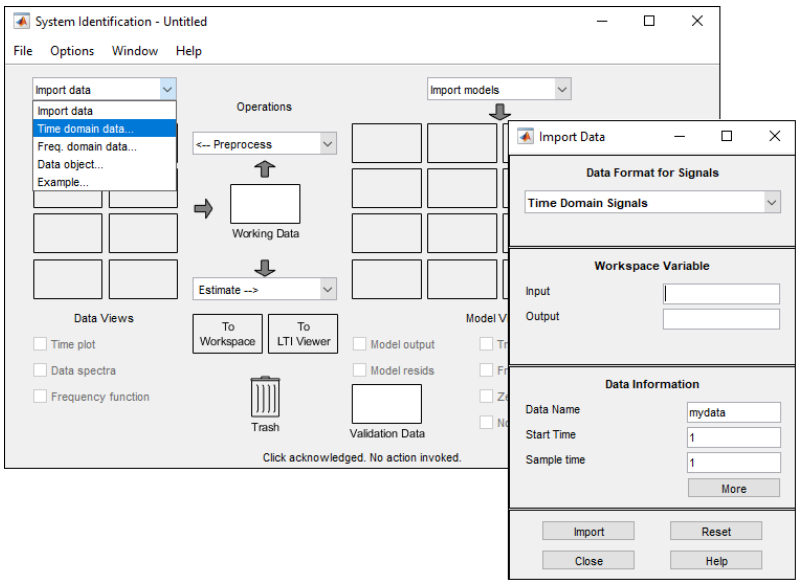


Figura 2.22: *System Identification Toolbox* - Cargar datos

El siguiente paso es uno de los más importantes en el proceso, pues se deben adecuar los datos mediante su **procesamiento**. Esto es imprescindible, especialmente cuando ocurre alguna de las siguientes situaciones:

- **Existen valores faltantes o defectuosos:**

  Los fallos pueden presentarse tanto en los datos de entrada como de salida y puede deberse a errores en la adquisición de datos (interrupción en la comunicación con los sensores, fallo de medida de los sensores, entre otros).

  En el caso de **datos faltantes** será fácil su identificación, puesto que en el set de datos habrá celdas con el valor "NaN" (*Not a Number*). Este problema se soluciona realizando una interpolación utilizando, por ejemplo, el comando *misdata*. Hay dos opciones de emplearlo:

  1. Se puede estimar un modelo con el comando *tfest* a la fracción del set de datos que tiene todos sus valores y, posteriormente, estimar la parte faltante con dicho modelo. Es decir, si hay datos faltantes entre los índices 100 y 200 de un conjunto de datos, pero a partir del instante 201 los datos presentan un comportamiento adecuado, se estima un modelo para los datos a partir de dicho momento del siguiente modo:

     sys = tfest (data, np, nz)

     donde *data* será el conjunto de datos del instante 201 en adelante, *np* el número de polos que debe contener el modelo y *nz* el número de ceros. Una vez obtenido el modelo, se realiza la estimación de los datos faltantes con el mismo. Para utilizar esta opción es necesario conocer qué tipo de modelos se ajusta bien a los datos.
  2. Si no se conoce el modelo que se ajusta al sistema, se puede especificar el número máximo de iteraciones y la tolerancia. Las iteraciones terminan cuando la diferencia entre dos estimaciones de datos consecutivas difiere en menos de la tolerancia especificada.

  La existencia de **valores atípicos** se detecta fácilmente representando los datos gráficamente en función del tiempo e identificando los valores que se alejan del rango. Para minimizar sus efectos se puede proceder de varias maneras:

  1. Reemplazar los valores atípicos por valores "NaN" y tratarlos como si fueran datos faltantes.
  2. Eliminarlos filtrando previamente los datos para el contenido de alta frecuencia, pues a menudo resultan de cambios abruptos. Para este fin existen múltiples comandos disponibles en la librería de MATLAB.

- **Existen *offset* y *drifts* en los niveles de la señal:**

  Para estimar modelos lineales más precisos hay que eliminar tendencias (medias u *offset*) de las señales de datos de entrada-salida en el dominio del tiempo. Así, los modelos lineales obtenidos a partir de los datos sin tendencia describen la relación entre la **variación** de la señal de salida y la **variación** de la señal de entrada [12].

- **Existen perturbaciones de alta frecuencia por encima del intervalo de frecuencia de interés para la dinámica del sistema:**

  Si los datos se muestrean más rápido de lo necesario durante los experimentos, se puede cambiar el muestreo si perder información. El remuestreo tiene en cuenta cómo se comportan los datos entre muestras y aplica un filtro FIR *antialiasing* (paso bajo) a los datos para cambiar la tasa de muestreo de la señal [12].

Todas estas y más operaciones de preprocesamiento se pueden realizar desde líneas de comandos o desde la misma aplicación, bajo la pestaña de *Preprocess* (figura 2.23). Los cambios que se realizan se aplican al set de datos seleccionado como *"Working Data"* y, dependiendo de las características del mismo, habrá unas u otras operaciones de preprocesamiento disponibles [12].

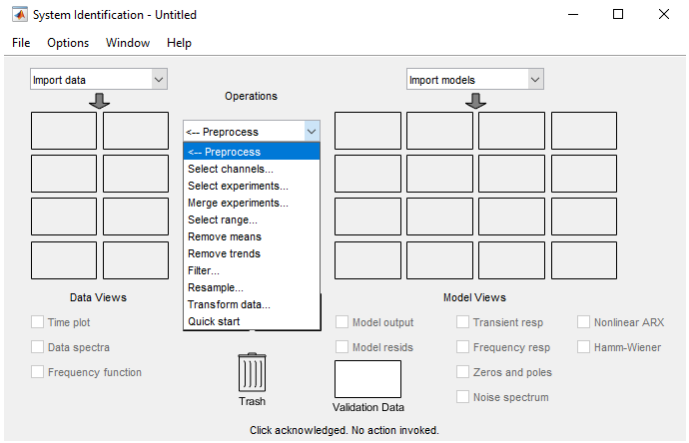


Figura 2.23: *System Identification Toolbox* - Preprocesamiento

Una vez adecuados los datos, se procede a la elección del modelo y la estimación. Como se ha explicado en la sección 2.2, se trabajará fundamentalmente con dos tipos de modelos: función de transferencias y modelo en espacio de estados, utilizados para la respuesta forzada (durante la inyección de CO_2) y la respuesta libre (sin inyección de CO_2) respectivamente.

Estos dos tipos de modelos se pueden estimar desde la propia aplicación o desde línea de comandos, mediante un *script*. En este trabajo se utilizará la segunda opción, puesto que para manejar grandes cantidades de datos y obtener muchos modelos es más eficiente, aunque desde la aplicación es muy sencillo. Además, el hecho de realizar un programa que estime los modelos permite automatizar la obtención de modelos.

Para elegir el modelo que se desea estimar, se deben colocar los datos correctamente procesados en el área de "*Working Data*" (figura 2.24).

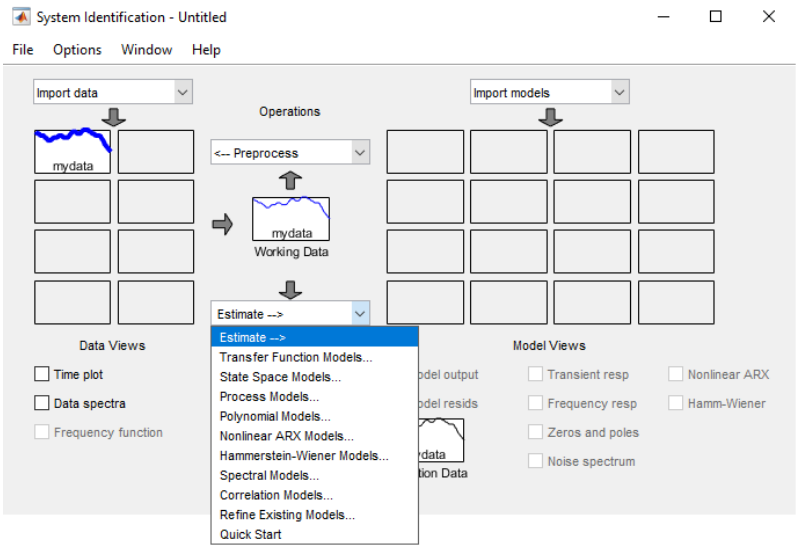


Figura 2.24: *System Identification Toolbox* - Estimación de modelos

A continuación, se explica el procedimiento a seguir tanto desde la aplicación como desde línea de comandos para cada uno de los modelos a estimar.

- **Función de transferencia:**

  Para estimar la función de transferencia hay dos opciones. Se puede estimar seleccionando la opción ***Estimate Transfer Functions*** o bien eligiendo ***Process Models***.

  En caso de utilizar la opción de ***"Transfer Functions"***, se debe seleccionar el número de polos y ceros, además de establecer si la estimación debe hacerse en tiempo continuo o tiempo discreto. Pasar del dominio en tiempo continuo a discreto y vicerversa es muy sencillo, basta con conocer el tiempo de muestreo

del sistema. Sin embargo, en tiempo continuo la interpretación de los parámetros es más sencilla e intuitiva, por lo que puede resultar conveniente elegir dicha opción. Además, se puede indicar si el sistema presenta retardo, estableciéndolo como un parámetro fijo, caso en el cuál tendría que ser calculado manualmente, o estableciendo límites máximo y mínimo, caso en el cual sería calculado por la aplicación. Todas estas opciones se observan en la figura 2.25.

Además, se pueden establecer otras opciones en la pestaña "*Estimation Options*", como el rango de frecuencias al que debe ajustarse el modelo, el método utilizado en la estimación, el máximo de iteraciones, la tolerancia, entre otros.

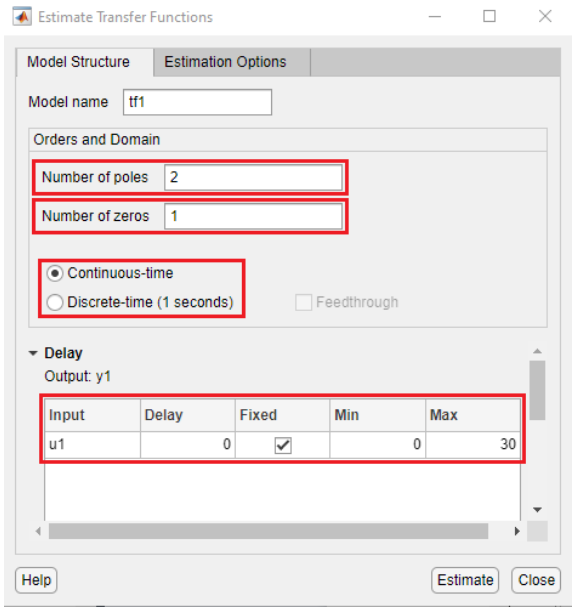


Figura 2.25: *System Identification Toolbox - Estimate Transfer Functions*

A pesar de ser una herramienta muy útil, en ciertas ocasiones es necesario definir los parámetros con más exactitud de lo que la misma permite. Es por esto que, aunque sea una opción muy válida, puede resultar conveniente utilizar la opción de estimar el modelo con la opción ***"Process Models"*** (figura 2.26).

En dicha herramienta se puede establecer el número de polos y su forma (reales o complejos). Además, se puede elegir un modelo con o sin ceros, con o sin retardo y se puede establecer como condición que contenga un integrador (polo en $s = 0$). Otra característica muy útil de esta opción es que, una vez determinados los parámetros que tendrá el sistema, se pueden establecer bien su valor y determinarlo como conocido, o bien un valor inicial para la estimación y los límites entre los que debe estar.

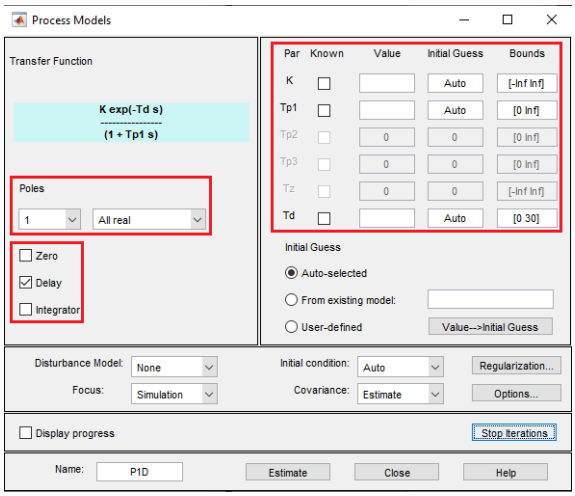


Figura 2.26: *System Identification Toolbox - Process Models*

Como la segunda opción presenta ventajas visibles sobre la primera, será esta la que se utilice en el presente trabajo. Para realizar la estimación desde líneas de comandos, se debe seguir el siguiente procedimiento:

- Una vez obtenidos y procesados los datos, se debe construir un objeto "*iddata*", utilizando la siguiente instrucción:

  data = iddata (y, u, Ts)

  donde "y" es la salida del sistema, "u" la entrada y "Ts" el tiempo de muestreo.

- A continuación, se estima el modelo con la instrucción:

  sys = procest (data, type)

  donde "data" son los datos del sistema preparados anteriormente y "type" es el tipo de sistema que se quiere estimar.

El tipo de sistema empieza por la letra P seguida del número de polos que se espera que tenga el modelo. Además, se puede agregar la letra I para exigir un integrador, la D para imponer un tiempo de retardo y la Z para agregar un cero. Además, si se quieren polos complejos se debe agregar una U. Por ejemplo, un tipo de modelo "P1D" tendrá un polo y retardo, mientras que un modelo tipo "P2DZ" tendrá dos polos, retardo y un cero.

- En la estimación se pueden utilizar otras opciones como el objetivo de la estimación, las condiciones iniciales, el método numérico utilizado, entre otras.
- Una vez estimado el modelo, se pueden obtener sus parámetros con el comando ***getpvec*** y se puede comparar con los datos originales (gráficamente) utilizando el comando ***compare***.

- **Espacio de estados:**

  Para estimar modelos en espacio de estados se utiliza la interfaz de la figura 2.27. Se debe definir el orden del modelo o bien un rango, caso en el cual se escogerá aquel que se ajuste mejor. Se puede escoger una estimación en tiempo continuo o discreto y con o sin retardo.

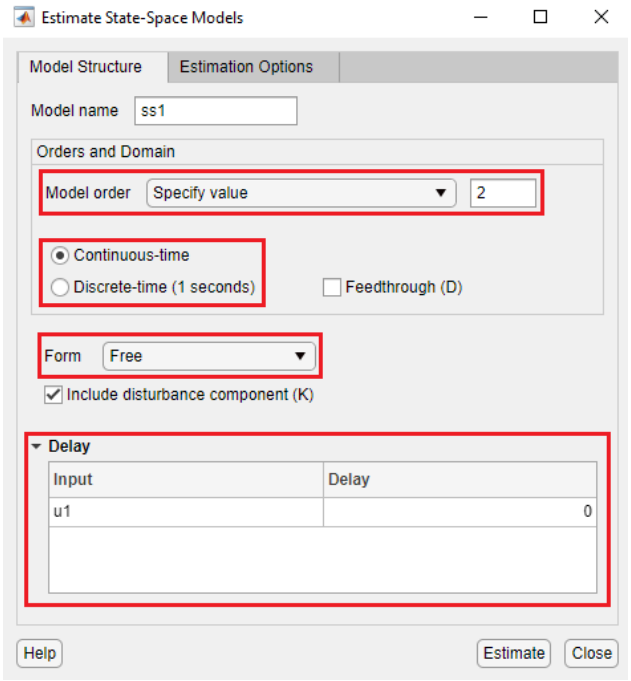


Figura 2.27: *System Identification Toolbox - State Space Models*

Por último, en la pestaña referente a la estructura del modelo, se debe escoger su forma. Esta puede ser de libre elección para el algoritmo, o se puede fijar una estructura fija, siendo la más común la canónica de observabilidad.

En la pestaña correspondiente a las opciones de estimación hay que escoger entre dos métodos de estimación. El primero es el llamado **PEM**, cuyas siglas son la abreviación de *Prediction Error Minimization*. Se basa en iteraciones que buscan minimizar cierto criterio. Las iteraciones se empiezan con valores de los parámetros calculados mediante el método *n4sid* y la parametrización de las matrices A, B, C, D y k es libre. El segundo método es el **N4SID** y se trata de un método basado en el subespacio no iterativo. La calidad de los resultados obtenidos depende enormemente de dos opciones llamadas *N4Weight* y *N4Horizon* [12]. La primera determina algunas matrices de ponderación que se utilizan en la descomposición de valores singulares, un paso central del algoritmo. Se puede escoger "MOESP", correspondiente al algoritmo MOESP de Verhaegen o "CVA", correspondiente al algoritmo de variable canónicas de Larimore. La segunda opción, *N4horizon*, determina los horizontes de predicción hacia adelante y hacia atrás utilizados por el algoritmo. Se trata de un vector fila con tres elementos:

N4Horizon = [r sy su]

donde "r" es el máximo horizonte de predicción hacia adelante, "sy" el número de salidas pasadas y "su" el número de entradas pasadas utilizadas [12].

2.4.2. Regression Learner Toolbox

La aplicación *Regression Learner* de MATLAB será utilizada en el presente trabajo para poder estimar los parámetros de los modelos a partir de las condiciones a las que se encuentra sometido el sistema: radiación global, temperatura del medio y nivel del medio principalmente.

Para entrenar los árboles de regresión a partir de líneas de código se debe utilizar el comando **fitrtree**, al cual se le deben brindar como parámetros los predictores y las salidas del sistema (los parámetros que se quieren estimar) [15]:

tree = fitrtree (Tb1, Y)

donde "Tb1" es una tabla que contiene las variables de entrada (predictores), "Y" es un vector que contiene la salida y "*tree*" será el árbol de regresión obtenido.

Además, como en todos los comandos disponibles en el *Software* utilizado, se pueden especificar numerosas características que se quieran exigir a los resultados.

Uno de los parámetros más importantes que es posible especificar es la **profundidad** del árbol entrenado, puesto que, por defecto, el comando hace crecer árboles de decisión profundos. Disminuir esta característica ayuda a reducir la complejidad y el tiempo de cálculo. Con este fin, se utilizan los argumentos de '*MaxNumSplits*', '*MinLeafSize*'y '*MinParentSize*':

- ***MaxNumSplits:*** es el número máximo de divisiones de decisión (o nodos de rama), especificado como ***'MaxNumSplits',n***, siendo "n" un número entero positivo.
- ***MinLeafSize:*** es el número mínimo de observaciones del nodo hoja, especificado como ***'MinLeafSize', n***, siendo "n" un valor entero positivo.
- ***MinParentSize:*** es el número mínimo de observaciones del nodo de la rama, especificado como ***'MinParentSize', n***, siendo "n" un valor entero positivo.

Si se proporciona tanto *MinParentSize* como *MinLeafSize*, *fitrtree* utiliza la configuración que proporciona hojas más grandes: *MinParentSize = max(MinParentSize,2*MinLeafSize)*.

Esto es importante también para no tener tanta variabilidad en los parámetros estimados para los modelos. Es decir, si se deja la profundidad de los árboles por defecto, ante mínimas variaciones de las condiciones, los parámetros de los modelos variarán y, por tanto, también lo harán los parámetros del controlador. De esta forma, la señal de control presentará muchas oscilaciones que no brindan un mejor comportamiento de la salida, sino que requieren un mayor esfuerzo del actuador.

También se puede especificar que el árbol finalice cuando el error cuadrático medio ponderado por nodo caiga por debajo de $QuadraticErrorTolerance \cdot \epsilon$, donde ϵ es el error cuadrático medio ponderado de todas las n respuestas calculadas antes de hacer crecer el árbol de decisión (ecuación (2.9)).

$$\epsilon = \sum_{i=1}^{n} w_i (y_i - \overline{y})^2 \tag{2.9}$$

donde w_i es el peso de la observación i y la suma de todos los pesos suman 1 ($\sum_{i=1}^{n} w_i = 1$) y $\overline{y} = \sum_{i=1}^{n} w_i y_i$ es la media ponderada de todas las respuestas [15].

El árbol obtenido se observa gráficamente en la figura 2.28, el cuál es análogo al de la figura 2.21. Para obtener la vista gráfica se utiliza el comando **view (tree, 'Mode', 'graph')**.

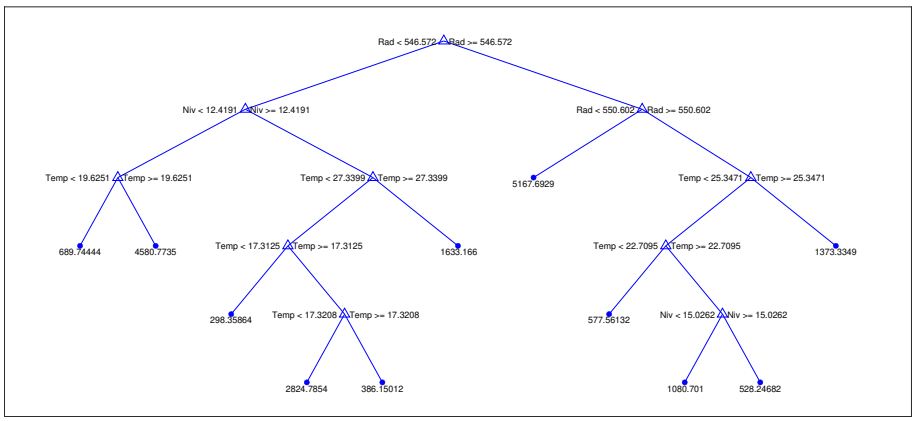


Figura 2.28: Árbol de regresión obtenido mediante MATLAB - vista gráfica

Si no se especifica el modo en el comando *'view'*, por defecto se muestra en forma de lista (figura 2.29).

```
>> view(TreeModeltau_sinCO2)

Decision tree for regression
 1  if Rad<546.572 then node 2 elseif Rad>=546.572 then node 3 else 523.274
 2  if Niv<12.4191 then node 4 elseif Niv>=12.4191 then node 5 else 410.079
 3  if Rad<550.602 then node 6 elseif Rad>=550.602 then node 7 else 826.474
 4  if Temp<19.6251 then node 8 elseif Temp>=19.6251 then node 9 else 1078.85
 5  if Temp<27.3399 then node 10 elseif Temp>=27.3399 then node 11 else 362.31
 6  fit = 5167.69
 7  if Temp<25.3471 then node 12 elseif Temp>=25.3471 then node 13 else 747.542
 8  fit = 689.744
 9  fit = 4580.77
10  if Temp<17.3125 then node 14 elseif Temp>=17.3125 then node 15 else 353.167
11  fit = 1633.17
12  if Temp<22.7095 then node 16 elseif Temp>=22.7095 then node 17 else 656.281
13  fit = 1373.33
14  fit = 298.359
15  if Temp<17.3208 then node 18 elseif Temp>=17.3208 then node 19 else 427.483
16  fit = 577.561
17  if Niv<15.0262 then node 20 elseif Niv>=15.0262 then node 21 else 868.219
18  fit = 2824.79
19  fit = 386.15
20  fit = 1080.7
21  fit = 528.247
```

Figura 2.29: Árbol de regresión obtenido mediante MATLAB

Por último, se puede estimar la importancia de cada predictor con las siguientes líneas de código, el resultado arrojado es un gráfico de barras (figura 2.30).

```
imp = predictorImportance(Tree);

figure;
bar(imp);
title('Predictor Importance Estimates');
ylabel('Estimates');
xlabel('Predictors');
h = gca;
h.XTickLabel = Tree.PredictorNames;
h.XTickLabelRotation = 45;
h.TickLabelInterpreter = 'none';
```

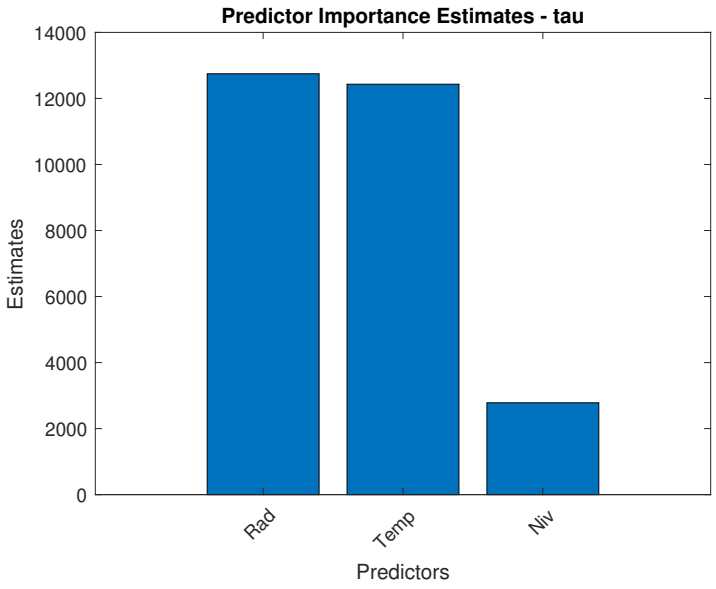


Figura 2.30: Importancia de cada predictor en un árbol de regresión

Una vez entrenado el árbol, se utiliza el comando **predict** para predecir el valor de la respuesta a partir de unos valores de entrada dados.

Yfit = predict (Mdl, X)

donde Yfit es el valor predicho de respuesta para los valores de los predictores X, basados en el árbol de regresión Mdl [16].

2.5. Algoritmos de control

Dado que el objetivo final del trabajo es realizar un control adaptativo del pH, se utilizará el lazo de control mostrado en la figura 2.31. En la misma se puede observar un bucle de control primario, que será el encargado de controlar el pH del sistema utilizando como señal de control el caudal de CO_2, y un bucle de control secundario, que es el encargado de controlar el comportamiento de la válvula de CO_2 utilizando como señal de control tensión.

Figura 2.31: Lazo de control

El bucle secundario ya se encuentra implementado con un controlador tipo PID (Proporcional, Integral y Derivativo), por lo que en este trabajo se hará hincapié en el lazo primario y el diseño del controlador PI. La señal de control en el dominio del tiempo brindada por un PI es de la forma:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau\right) \tag{2.10}$$

donde u es la señal de control y e es el error de control ($e = y_{sp} - y$, donde y_{sp} es la referencia o *setpoint* e y es la salida medida del sistema). La señal de control es la suma del término proporcional e integral [17]. Los parámetros del controlador que se deben diseñar en este caso son la ganancia proporcional K y el tiempo integral T_i. La acción de cada término es la siguiente:

- **Acción proporcional:** genera una acción proporcional al error y queda definida por la ganancia proporcional K.
- **Acción integral:** queda definida por el tiempo integral T_i y permite eliminar el error en régimen permanente, es decir, que la salida del sistema alcance la referencia. Produce una respuesta más oscilatoria.

En el dominio de Laplace, la ecuación de un controlador PI es la que se muestra a continuación:

$$C(s) = \frac{U(S)}{E(s)} = K\left(1 + \frac{1}{T_i s}\right) \tag{2.11}$$

Además, debido a no linealidades de los actuadores como la saturación, debe incluirse al esquema *anti-windup*. Esto se debe a que, cuando la señal de control alcanza el límite del actuador, el lazo de realimentación se rompe y el sistema opera como un sistema en bucle abierto independientemente de la salida del proceso. Al utilizar acción integral, el error sigue siendo integrado y el término integral se hace cada vez más grande, conocido como *windup* [17].

En la figura 2.32 se observa la señal de control, la señal medida y la referencia en un caso donde la señal se satura. Después del primer cambio en la referencia la señal de control aumenta hasta superar su máximo u_{max}, pero la señal máxima no es capaz de corregir el error, por lo que la parte integral de la señal de control u continúa aumentando. Sin embargo, la señal de control que le llega al sistema es el límite máximo u_{out}. Si después de cierto tiempo vuelve a bajarse la referencia, el signo del error se hace negativo y la señal de control comienza a disminuir. Como la señal de control u se encuentra muy por encima del máximo u_{max}, la señal de control que le llega al sistema u_{out} permanece en el límite durante un tiempo. En la figura 2.32 se observa que, debido a esto, la respuesta se retarda [17].

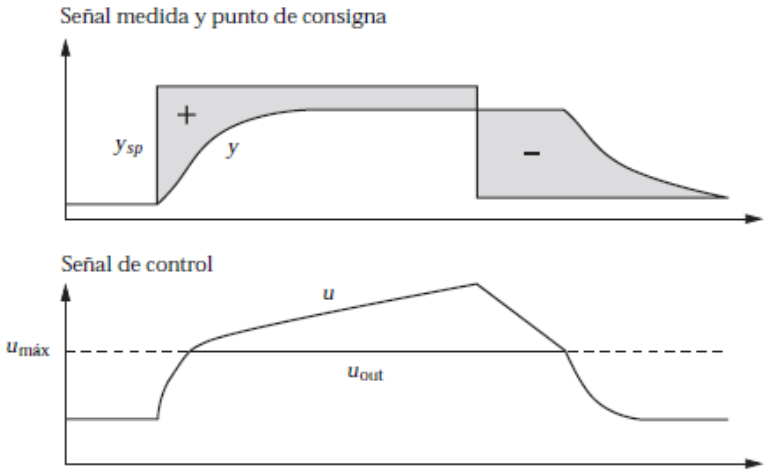


Figura 2.32: Ilustración de la saturación del integrador - Fuente: [17]

Para corregir este comportamiento, se utiliza un esquema *Antiwindup*. Cuando la salida se satura, se recalcula el término integral en el controlador de forma que su nuevo valor da una salida en el límite de la saturación. Esto se realiza dinámicamente con una constante de tiempo T_t. En la figura 2.33 se muestra un diagrama de bloques de un controlador PID (para el caso del PI es análogo pero eliminando la acción derivativa) con

antiwindup. Su funcionamiento se basa en realimentar el error e_s entre la señal máxima del actuador u y la salida generada por el controlador v. Esta señal e_s es alimentada a la entrada del integrador con una ganancia $1/T_t$. Cuando no haya saturación, el error e_s es nulo y la realimentación no tiene ningún efecto [17].

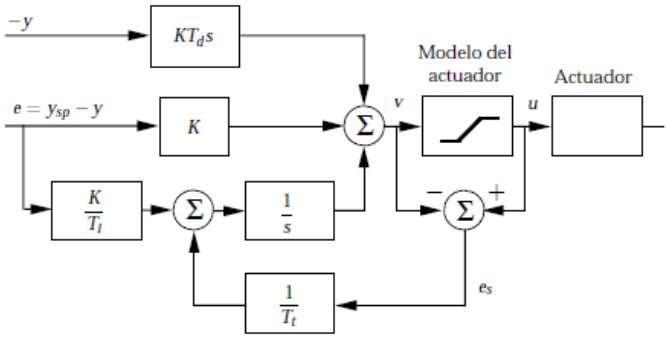


Figura 2.33: Controlador PID con mecanismo *anti-windup* - Fuente: [17]

La constante T_t se calcula según la ecuación (2.12) para controladores PID y según la ecuación (2.13) para controladores PI.

$$T_t = \sqrt{T_i T_d} \tag{2.12}$$

$$T_t = \sqrt{T_i} \tag{2.13}$$

Para sintonizar el controlador se debe analizar el modelo de función de transferencia del proceso. En este trabajo, se utilizarán modelos de la forma mostrada en la siguiente ecuación:

$$G(s) = \frac{k}{\tau s + 1} e^{-t_r s} \tag{2.14}$$

donde, como se verá en la sección 4.1.3 el retardo será no dominante. En ese caso, se podrá utilizar el método de sintonización analítico λ, cuyo desarrollo se muestra a continuación.

$$G(s) = \frac{k}{\tau s + 1} e^{-t_r s} \approx \frac{k(-t_r s + 1)}{\tau s + 1} \tag{2.15}$$

$$C(s) = K_p \left(\frac{T_i s + 1}{T_i s} \right) \tag{2.16}$$

$$G_{bc}(s) = \frac{C(s)G(s)}{1 + C(s)G(s)} \tag{2.17}$$

$$G_{bc}(s) = \frac{\frac{k(-t_r s+1)}{\cancel{\tau s+1}} K_p \left(\frac{\cancel{T_i s+1}}{T_i s} \right)}{1 + \frac{k(-t_r s+1)}{\cancel{\tau s+1}} K_p \left(\frac{\cancel{T_i s+1}}{T_i s} \right)} \Longrightarrow \boxed{T_i = \tau}$$

$$G_{bc}(s) = \frac{kK_p(-t_r s + 1)}{T_i s + kK_p(-t_r s + 1)} = \frac{kK_p(-t_r s + 1)}{s(T_i - kK_p t_r) + kK_p}$$
$$G_{bc}(s) = \frac{-t_r s + 1}{s(\frac{T_i}{kK_p} - t_r) + 1}$$

$$G_{bc} = \frac{1}{s(\frac{T_i}{kK_p} - t_r) + 1} e^{t_r s} \Longrightarrow \boxed{\tau_{bc} = \frac{T_i}{kK_p} - t_r} \tag{2.18}$$

La función de transferencia de bucle cerrado es la mostrada en la ecuación (2.18) y los parámetros del controlador K y T_i se calculan a partir de las expresiones recuadradas.

Además, el controlador diseñado presentará auto-ajuste, de modo que en cada momento se recalcularán los parámetros del modelo (k y τ) y según las expresiones desarrolladas anteriormente se recalcularán los parámetros del controlador (K y T_i). En la figura 2.34 se muestra el esquema de funcionamiento del control adaptativo que se debe implementar, donde la implementación de parámetros se realizará utilizando los árboles de regresión entrenados. Hay que tener en cuenta que, para evitar cambios bruscos en la señal de control se deberá implementar una transferencia sin saltos, de modo que no se permitan variaciones abruptas en los parámetros calculados.

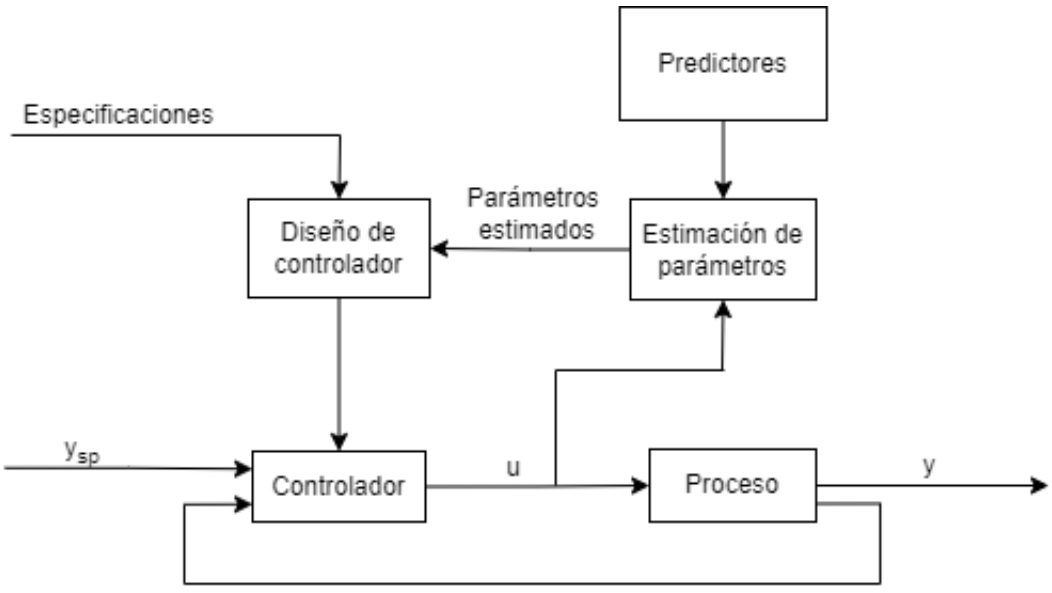


Figura 2.34: Esquema de control adaptativo

3

Desarrollo de modelos

Una vez explicado el funcionamiento del sistema, las herramientas de que se disponen y los métodos utilizados, se procede a describir detalladamente cómo ha sido aplicado lo anterior al presente trabajo.

3.1. Ensayos para la obtención de datos

Los experimentos para la obtención de datos se han realizado en el reactor *raceway* de aguas limpias disponible en las instalaciones del Convenio UAL-IFAPA, descrito en la sección 2.1. Se han realizado en un sistema real con el objetivo de contar con un sistema que se encuentre sometido a las condiciones reales, pues en un laboratorio las condiciones se encontrarían demasiado controladas. Así, se han obtenido datos de días con condiciones ambientales muy diversas: días muy soleados, días con nubes pasajeras, días con lluvia o incluso días en presencia de calima, donde la radiación solar era considerablemente baja.

Los ensayos realizados consistían de un bucle de control todo/nada con una banda de histéresis de $\pm 0{,}5$, siendo la referencia un pH de 8. Así, cuando el pH llegaba a 8.5 comenzaba la inyección de CO_2, que provocaba su descenso. Una vez que el pH llegaba a 7.5 cesaba la inyección del gas y, pasado el tiempo de retardo característico del sistema, la fotosíntesis realizada por las microalgas provocaba un aumento del pH. El caudal de CO_2 podía ser manipulado, de modo que cada semana se variaba el mismo para variar las condiciones.

Los ensayos se realizaron durante aproximadamente 7 semanas, distribuidas en los meses de noviembre (4 semanas), marzo (2 semanas) y abril (1 semana), obteniendo así datos de otoño y primavera.

El control descrito para realizar los ensayos se ha implementado en el software *Machine Expert 2.0* (Figuras 3.1 y 3.2).

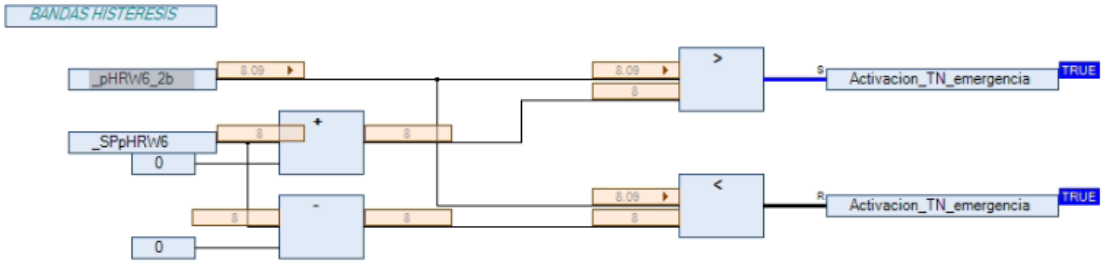


Figura 3.1: Bandas de histéresis para implementación de control todo/nada

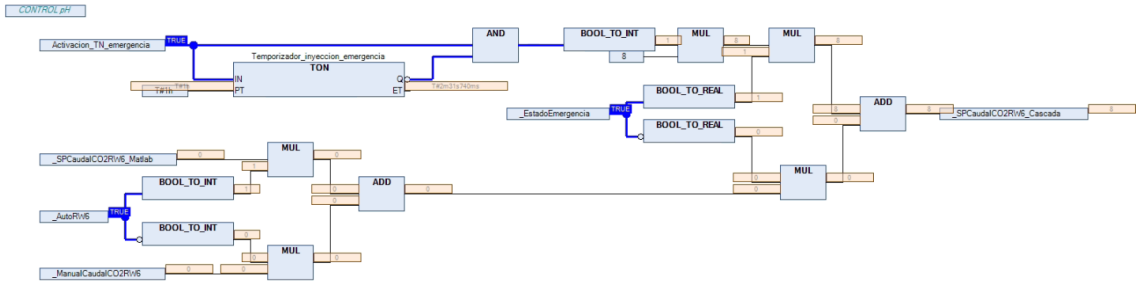


Figura 3.2: Implementación de control todo/nada con histéresis

El esquema se encuentra divido en dos partes:

1. **Banda de histéresis:** se compara el pH medido por el sensor en el punto más alejado del foso con el valor de la referencia más la banda de histéresis. En el momento en que se cumple la condición de $pHRW6\_2b > \_SPpHRW6 + 0.5$ se activa una variable llamada "Activación_TN_emergencia", siendo pHRW6_2b el valor de pH medido y _SPpHRW6 la referencia o *setpoint* de pH en el reactor. La variable "Activación_TN_emergencia" se mantiene activa hasta que se cumpla la condición $pHRW6\_2b < \_SPpHRW6 - 0.5$, es decir, hasta que el pH se encuentre por debajo de 7.5 (figura 3.1).

2. **Control pH:** esta parte del esquema recibe como entrada la variable que se activa y desactiva según el pH medido se encuentre por encima o por debajo de la banda de histéresis (Activacion_TN_emergencia). Dicha variable se convierte de tipo *bool* a *int*, es decir, se convierte de "TRUE" a un 1 y se multiplica por el valor del caudal que se desea inyectar, en este caso 8 L/min y se envía a la tercer parte del esquema (figura 3.3) como referencia de caudal ("_SPCaudalCO2RW6_Cascada"). El temporizador "Temporizador_inyeccion_emergencia" es un temporizador cuya salida será 1 si lleva menos de una hora inyectándose CO_2 y, en el momento que se cumpla dicho tiempo, se pondrá a cero. Así, si hay algún fallo en la medida de los sensores se restringe la inyección de CO_2 a una hora, intervalo de tiempo suficiente para conseguir la disminución de pH exigida. La zona inferior de este esquema permite enviar una referencia de caudal manualmente o desde un controlador programado en MATLAB (figura 3.2).

3. **Control en cascada:** el esquema recibe la consigna de caudal y, mediante un control PID, lo traduce a mV, señal de control de la válvula (figura 3.3).

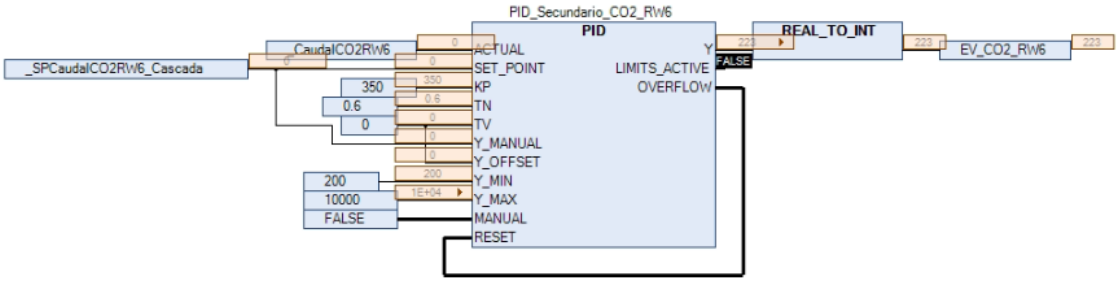


Figura 3.3: Control en cascada de la electroválvula de CO_2

Los datos obtenidos presentan la forma mostrada en las figuras 3.4 a 3.6. En ellas se puede observar el funcionamiento del control implementado y explicado anteriormente, además de poder observar perfiles de radiación muy distintos entre los tres ejemplos mostrados. El perfil de la temperatura del medio no presenta grandes variaciones, pero sí se nota una pequeña diferencia en los valores mínimo y máximo que se alcanzan.

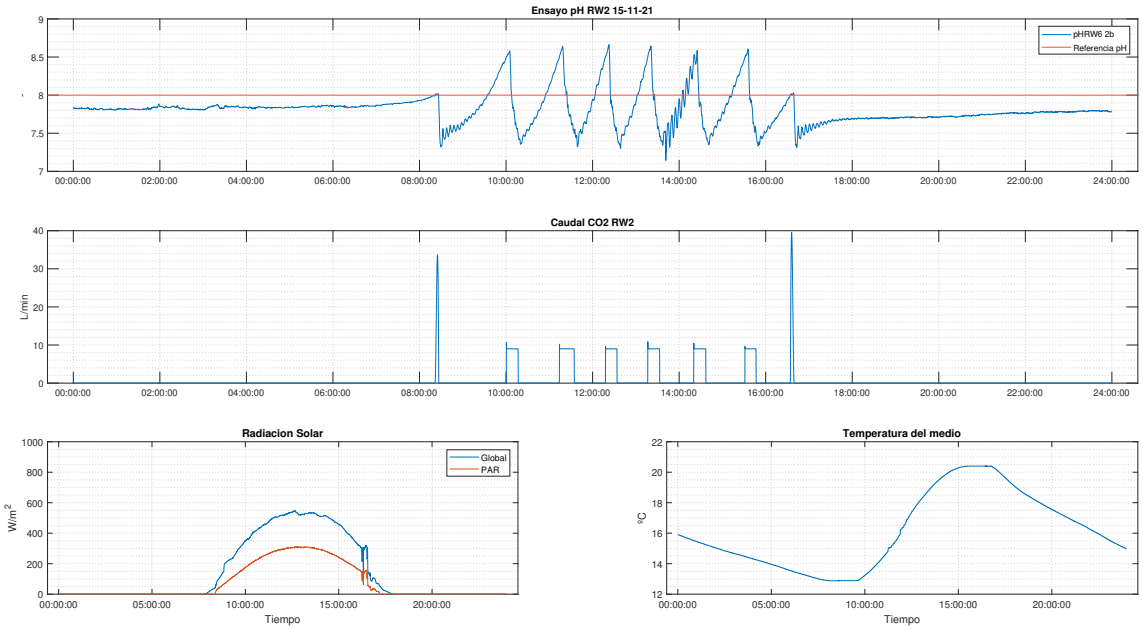


Figura 3.4: Ensayo para la obtención de modelos - 15 de noviembre de 2021

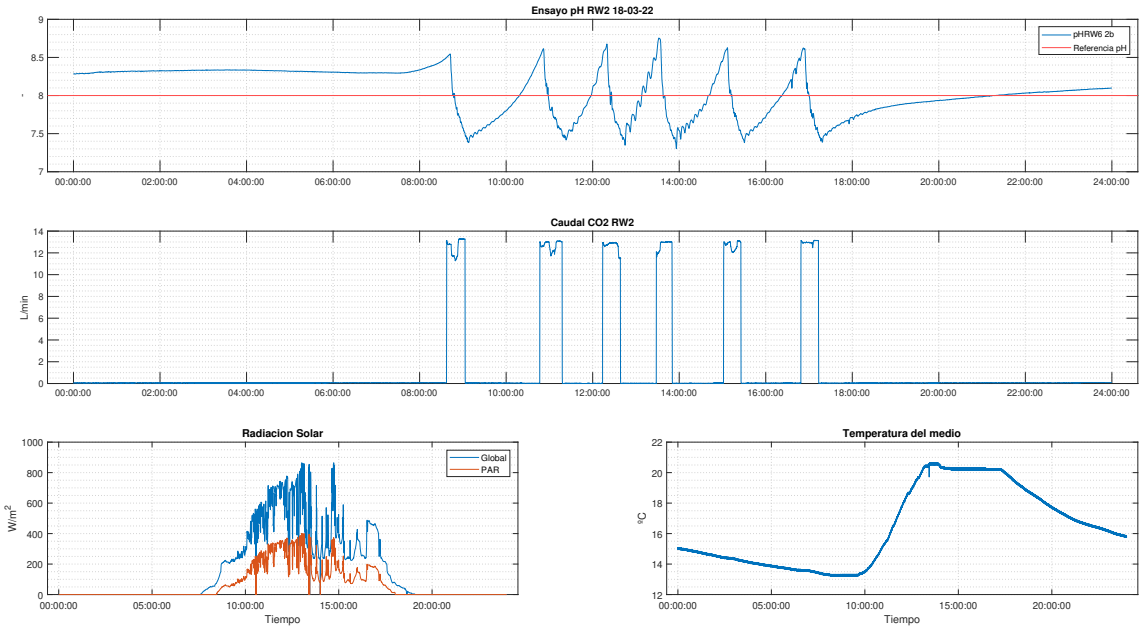


Figura 3.5: Ensayo para la obtención de modelos - 18 de marzo de 2022

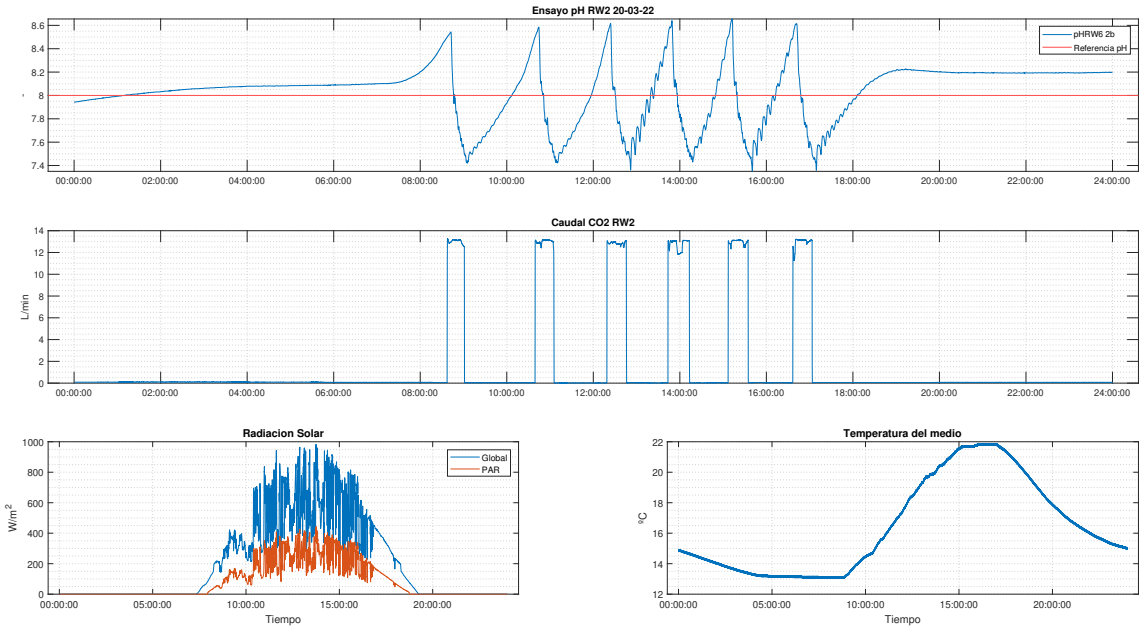


Figura 3.6: Ensayo para la obtención de modelos - 20 de marzo de 2022

3.2. Obtención de modelos

Una vez realizada una gran cantidad de ensayos para la obtención de datos, se procedió al análisis de datos para la obtención de los modelos. Al analizar el comportamiento del pH obtenido en los ensayos realizado, se pueden distinguir las dos dinámicas que fueron objeto de estudio en el presente trabajo: un descenso causado por la inyección de dióxido de carbono y un aumento del mismo al cesar la inyección, causada por la fotosíntesis realizada por los microorganismos. Además, se observa que la velocidad de evolución de la salida varía de un impulso de CO_2 a otro debido al cambio en las condiciones meteorológicas.

Los datos son extraídos del ordenador industrial descrito en la sección 2.1.4 en archivos de extensión .csv, donde cada uno corresponde a los datos de un día, desde las 00:00:00 hasta las 23:59:59 horas. En él se almacenan todas las variables de interés, tanto de los dos reactores disponibles como las variables que registran las condiciones meteorológicas. Antes de entrar en materia de obtención de modelos, fue imprescindible adecuar los datos. La tarea principal fue dividir los datos de todo el día en los intervalos de interés.

El PLC presenta un control por defecto, es decir, un controlador que funciona las 24 horas del día, salvo que se ejecute otro ensayo. Se trata de un controlador todo/nada que cuando el pH medido supera la referencia inyecta el caudal máximo de CO_2 que puede brindar la válvula. Algunos días de las campañas realizadas, como el de la figura 3.4, el ensayo para la obtención de modelos se encuentra comprendido entre dos actuaciones de la válvula correspondiente al control todo/nada sin histéresis. Por lo tanto, fue necesario identificar los intervalos en que se encuentra en acción el control diseñado para la obtención de modelos. Además, dentro del control diseñado para este trabajo, hay intervalos con inyección de CO_2 e intervalos sin, que permiten distinguir entre los dos tipos de respuestas que son objeto de estudios.

Es por esto que, el primer paso a realizar una vez obtenida una gran cantidad de datos, fue realizar un programa en MATLAB que permitiera separar fácilmente en tramos distintos según el comportamiento del sistema. Esto se podría realizar manualmente, pero al tratarse de una campaña prolongada de ensayos, se ha realizado un *script* que permitiera una mayor eficiencia (código facilitado en secciones A.1.1 y A.1.3). Una vez realizada la distinción de comportamientos, se pudo proceder a la obtención de modelos.

3.2.1. Modelos para respuesta forzada

Para modelar la respuesta forzada se ha utilizado una función de transferencia de primer orden de la forma mostrada en la ecuación 3.1. Por lo tanto, era necesario conocer el valor de tres parámetros: ganancia estática, constante de tiempo y retardo del sistema.

$$pH(s) = \frac{k}{\tau s + 1} e^{-t_r s} CO_2(s) \tag{3.1}$$

La ganancia y la constante de tiempo se encuentran fuertemente vinculadas a las condiciones ambientales, mientras que el retardo es un parámetro fijo que depende de las características físicas del sistema. En este caso específico, representa el tiempo que tarda la masa de cultivo en desplazarse desde el foso detrás de las palas al sensor de medida. Por lo tanto, dicho parámetro depende de la velocidad del fluido, que se fija gracias a la velocidad de las palas.

En primer lugar, se han obtenido modelos cuyo retardo pudiera variar, es decir, en cada modelo se calculaba el retardo. El cálculo se realizaba a partir de los datos y con unas pocas líneas de código que se muestran a continuación. El mismo consiste en tomar cada tramo de bajada y comparar las muestras de cuatro en cuatro, utilizando una "ventana deslizante". Así, en el momento en que las cuatro muestras presentaran un comportamiento descendiente, se tomaba el índice como tiempo de retardo.

```
for i = 1: length(Intervalos_pHRW23_1)
   for k = 3:(length(Intervalos_pHRW23_1{i})-2)
      if (Intervalos_pHRW23_1{i}(k) < Intervalos_pHRW23_1{i}(k-1)) && (
          Intervalos_pHRW23_1{i}(k) > Intervalos_pHRW23_1{i}(k+1))...
            && (Intervalos_pHRW23_1{i}(k-1) < Intervalos_pHRW23_1{i}(k-2));
         indice(i,k) = k-2;
      end
   end
   idx1(i,1)=find(diff(indice(i,:)>18),1);
   principio(i) = indice(i,(idx1(i,1)+1));
end
```

Así, se obtuvieron modelos para la primera campaña de ensayos, realizada en noviembre, con el retardo variable. Una vez que se obtuvo una gran cantidad de datos, dado que el retardo es fijo, se tomó como retardo el promedio de todos los obtenidos. Así, se llegó a la conclusión de que se debía trabajar con un retardo $t_r = 270$ s.

Posteriormente, se volvieron a obtener los modelos pero esta vez sin estimar el retardo, pues se utilizó el obtenido anteriormente. Una vez creados los vectores correspondientes a cada intervalo del cual se quería estimar un modelo, se utilizaron las siguientes líneas de código para realizar la estimación.

```
%% Obtencion del modelo de respuesta forzada con System Identification
Ts = 10;           % Tiempo de muestreo de 10 segundos
Tr = 27;           % Tiempo de retardo
for i = 1: length(TramosCO2(1,:))
   final = find(TramosCO2(:,i)==0,2);
   if size(final) == 1
      data = iddata(TramospH(:,i),TramosCO2(:,i),Ts);
   else
      data = iddata(TramospH(1:final(2)-1,i),TramosCO2(1:final(2)-1,i),Ts);
   end
   dataset{i} = data;          % Datos
   modelo = procest(data,'P1'); % Primer orden sin retardo
   modeloset{i} = modelo;      % Modelos
   Datos_Modelo = getpvec(modelo); % Obtencion de parametros
   kp(i) = Datos_Modelo(1);
   tau(i) = Datos_Modelo(2);
   modelos{i} = tf(kp(i),[tau(i) 1],'InputDelay',Tr*Ts); % Sistemas con
       retardo
end
```

En estas líneas se define el tiempo de muestreo utilizado (10 segundos en la primera campaña de ensayos y 1 segundo para la segunda) y, para cada tramo se crea el tipo de datos necesario para utilizar los comandos del *System Identification Toolbox*. Para esto, se utiliza el comando **iddata**, que necesita como parámetros el vector de salida, el vector de entrada y el tiempo de muestreo. Una vez hecho esto, se estima el modelo utilizando el comando **procest**, al cual se le brinda como parámetros los datos anteriormente creados y el tipo de modelo. Tal como se ha explicado en la sección 2.4.1, para obtener modelos de primer orden sin retardo el segundo parámetro será **'P1'**. Una vez estimado el modelo, con el comando **getpvec** se obtienen los parámetros de interés, en este caso, la ganancia y la constante de tiempo. En la línea 25 se construye el modelo con los parámetros obtenidos pero se le agrega el tiempo de retardo (variable "t_r"). A continuación, se puede validar el modelo gráficamente comparando su respuesta a escalón con los datos reales.

Para cada modelo obtenido se almacenan sus parámetros característicos en un archivo de extensión '.xls'. En la figura 3.7 se muestra una fracción de todos los datos obtenidos.

	A	B	C	D	E	F	G	H
1	**Dia**	**Mes**	**Caudal**	**Ganancia**	**CteTiempo**	**RadiacionMedia**	**TemperaturaMedia**	**NivelMedio**
2	5	10	2	-2.023971111	3549.842126	453.2453859	17.87260067	12.29850671
3	5	10	2	-0.76361311	1197.328983	617.1834846	21.36842179	14.95944972
4	5	10	2	-0.80960536	1592.940918	725.5964862	23.75663018	15.02133871
5	5	10	2	-0.859285989	1870.83893	753.2361551	25.91042722	15.00389451
6	5	10	2	-0.838852528	1703.202272	646.3732394	27.41357394	14.99597183
7	5	10	2	-0.929895945	1633.166027	450.7028631	27.98539106	15.03500279
8	6	10	2	-2.568427881	4580.773544	439.6354167	20.05888021	12.06651736
9	6	10	2	-0.709464425	1199.381328	584.6231796	22.88386529	13.85031553
10	6	10	2	-0.634521764	1352.779435	700.5929622	25.06411765	15.01694118
11	5	11	5	-0.267759579	930.5517247	433.0565625	12.67365	14.7262325
12	5	11	5	-0.242190283	835.0380237	556.6321429	15.39682143	14.40265
13	5	11	5	-0.248061081	929.283437	564.748981	17.91684783	14.46818841

Figura 3.7: Almacenamiento de datos de la respuesta forzada.

Como se puede observar, se guardan los datos de cada intervalo en una línea nueva y se almacenan los valores del caudal de CO_2, la ganancia y la constante de tiempo estimadas mediante *System Identification Toolbox* y los valores medios de radiación, temperatura del medio y nivel del medio. Estos últimos tres se guardan ya que los parámetros del modelo se intentarán estimar a partir de sus valores instantáneos.

Una vez analizadas las dos campañas de ensayos, se han obtenido 207 modelos, cantidad suficiente para poder estimar la ganancia y la constante de tiempo del sistema a partir de las condiciones ambientales. En las figuras 3.8 y 3.9 se puede observar el ajuste de los modelos obtenidos a los datos reales, siendo este considerablemente bueno. La primera corresponde a la primera campaña, concretamente al 5 de octubre de 2021, mientras que la segunda es del día 2 de abril de 2022, correspondiente a la segunda campaña de ensayos.

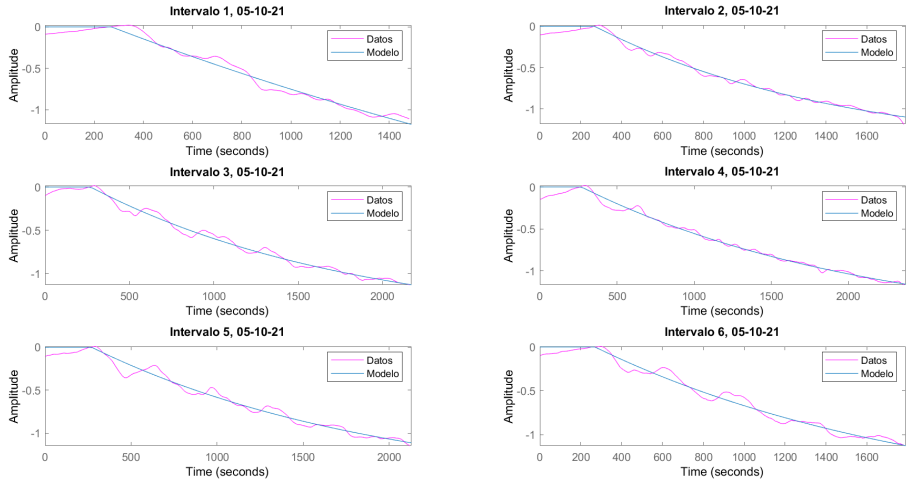


Figura 3.8: Comparación modelo-datos, 05 de octubre de 2021

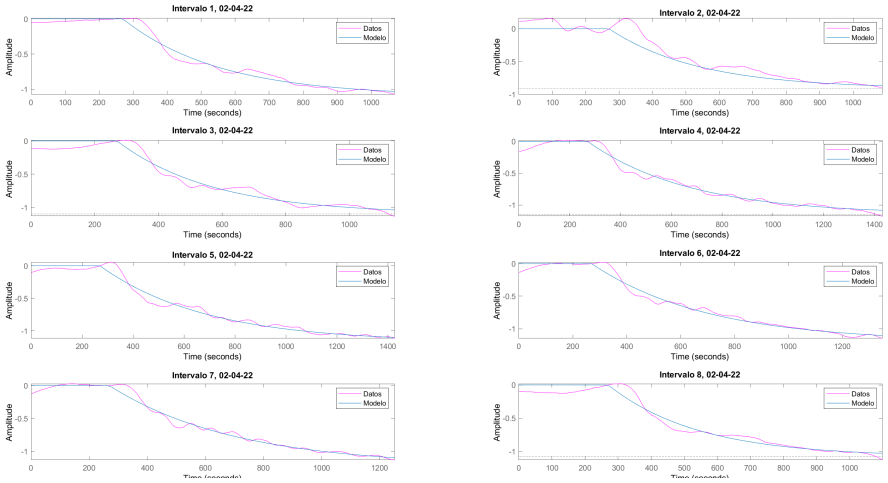


Figura 3.9: Comparación modelo-datos, 02 de abril de 2022

3.2.2. Modelos para respuesta libre

Para modelar la respuesta libre del sistema, que tiene lugar en ausencia de inyección de CO_2, se ha procedido de dos maneras distintas. En primer lugar, se ha modelado utilizando la técnica de mínimos cuadrados y, posteriormente, utilizando la aplicación *System Identification Toolbox*, descrita en la sección 2.4.1. En la presente sección, se describen detalladamente ambos procedimientos.

- **Modelado con ajuste por mínimos cuadrados:**

  En primer lugar, se ha intentado calibrar un modelo de primer orden, de la forma:

$$pH(k) = \theta_1 pH(k-1) \tag{3.2}$$

  donde se expresa que el pH del instante k depende del pH del instante anterior a través de una constante, θ_1. Sin embargo, no se conseguía un buen ajuste del modelo, por lo que se agregó un parámetro adicional, un término independiente, que permitía introducir un *offset* y ajustar así el modelo a los datos reales.

  Por lo tanto, la ecuación en diferencias que se debía calibrar era de la forma:

$$pH(k) = \theta_1 pH(k-1) + \theta_2 \tag{3.3}$$

  donde θ_1 y θ_2 son los parámetros cuyos valores se quieren calcular y $pH(k)$ y $pH(k-1)$ son el pH del instante actual y el anterior, ambos conocidos.

  Para realizar la estimación, se definieron los siguientes vectores:

$$Y = pH(k) \tag{3.4}$$

$$M = \begin{bmatrix} pH(k-1) & 1 \end{bmatrix} \tag{3.5}$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \tag{3.6}$$

  donde M es el vector de salidas (regresor) y θ es el vector de parámetros a calcular. Así, el cálculo de sus valores se consiguió realizando la siguiente operación:

$$\theta = (M^T \cdot M)^{-1} M^T Y \tag{3.7}$$

  A continuación, se muestra una porción del código, correspondiente a los cálculos de θ_1 y θ_2. En la misma se definen el vector Y, con los instantes actuales de pH (desde $k = 2$ hasta $k = end$) y la matriz M con una columna correspondiente a los instantes anteriores de pH (desde $k = 1$ hasta $k = end - 1$, correspondiente a θ_1) y otra rellena con unos (correspondiente a θ_2). Además, se calcula el vector de parámetros según la expresión de la ecuación (3.7).

```
n = 1;

for i = 1:size(pH_libre,2)
   Y(1:largo(i)-1,i) = pH_libre(n+1:largo(i),i);
   M(1:largo(i)-1,i) = pH_libre(n:largo(i)-n,i);
   M1(1:largo(i)-1,i) = ones;
   theta = [M(retardo:largo(i)-1,i) M1(retardo:largo(i)-1,i)]\Y(
       retardo:largo(i)-1,i);
   constantes2(i,:) = theta';
   ye(1:n,i) = pH_libre(retardo,i);
end
```

Una vez estimados los parámetros, se debe comprobar si el ajuste del modelo a los datos reales es adecuado. Para eso, se define la variable 'ye', que hace referencia a la salida (o pH) estimada. Su primer valor será igual al pH leído por el sensor, mientras que para los siguientes instantes se calcula según la expresión de la ecuación (3.3), puesto que ahora ya se conocen los valores de θ_1 y θ_2. Por último, se grafican ambas para comprobar el ajuste del modelo visualmente. A continuación, se muestra el código utilizado para dicha validación.

```
% Verifico si se hizo bien la identificacion
for i = 1:size(pH_libre,2)
   for k = n+1:largo(i)-retardo
      ye(k,i) = constantes2(i,1)*ye(k-1,i)+ constantes2(i,2);
   end
end

pH_libre = [pH_libre;zeros(1,length(pH_libre(1,:)))];
figure()
for i = 1:size(pH_libre,2)
   tiempo1 = 1:1:length(pH_libre(1:find(pH_libre(:,i)==0,1)-2,i));
   tiempo2 = 1+retardo:1:largo(i);
   subplot(round(size(pH_libre,2)/2),2,i)
   plot(tiempo1,(pH_libre(1:find(pH_libre(:,i)==0,1)-2,i)));
   hold on
   plot(tiempo2,ye(1:largo(i)-retardo,i));
   title('Respuesta libre '+string(i)+', '+string(day)+'-'+string(
       month)+'-'+string(year))
end
```

Una vez analizadas las dos campañas de ensayos, se han obtenido 202 modelos. En las figuras 3.10 y 3.11 se puede observar el ajuste de los modelos obtenidos a los datos reales, siendo este considerablemente bueno.

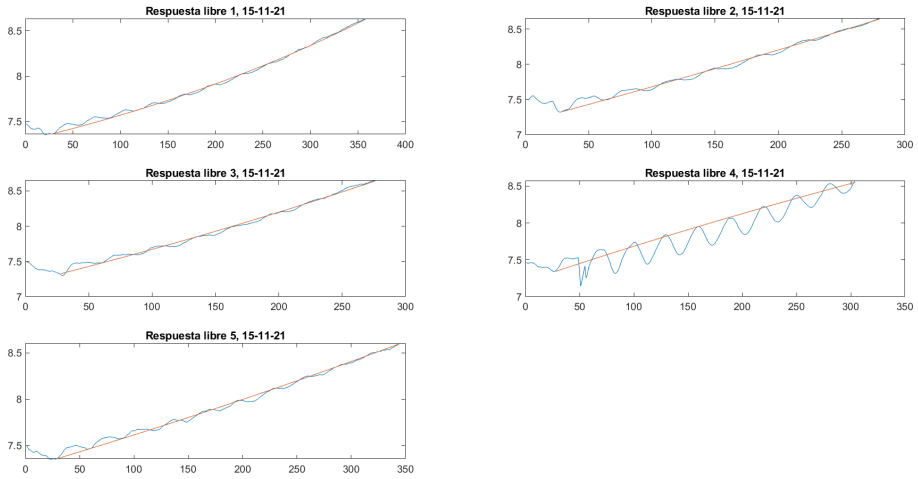


Figura 3.10: Comparación modelo-datos, 15 de noviembre de 2021

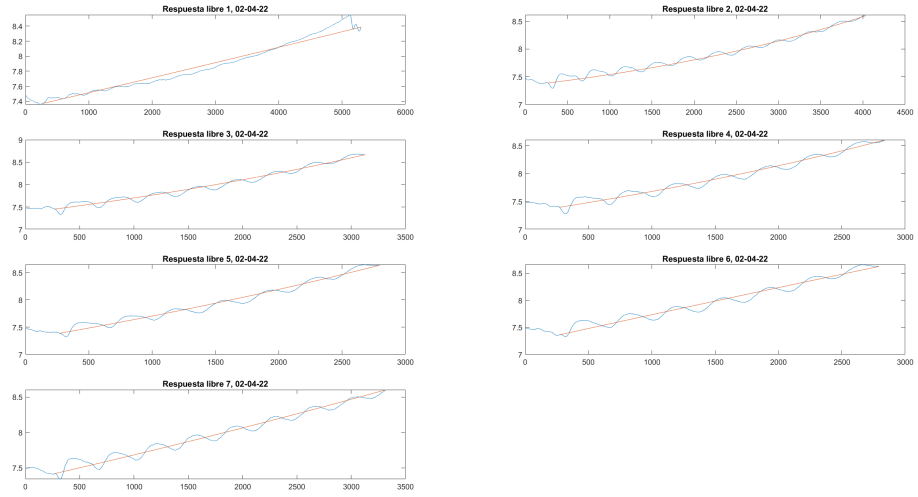


Figura 3.11: Comparación modelo-datos, 02 de abril de 2022

Se puede observar que los modelos se ajustan adecuadamente y que la respuesta del pH real presenta una dinámica oscilatoria que no es representada por el modelo obtenido. Estas oscilaciones son debidas al tiempo de residencia del fluido en el lazo y su período es el tiempo que tarda un volumen del fluido en dar una vuelta comple-

ta. Su modelado complicaría el control y no proporcionaría mejoras en términos de la productividad del sistema, por lo que el modelo obtenido sería adecuado para los objetivos perseguidos en el trabajo. Sin embargo, a la hora de probar el estimador se ha visto que los resultados proporcionados presentaban mucho error respecto al pH medido. Algunos comportamientos defectuosos del estimador se observan en las figuras 3.12 a 3.14 donde el pH estimado no solo se aleja considerablemente del pH medido sino que además su dinámica es incorrecta.

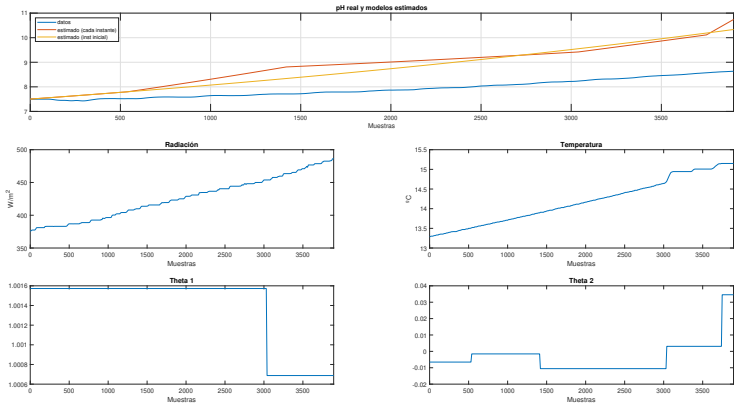


Figura 3.12: Estimación defectuosa de la respuesta libre - 1

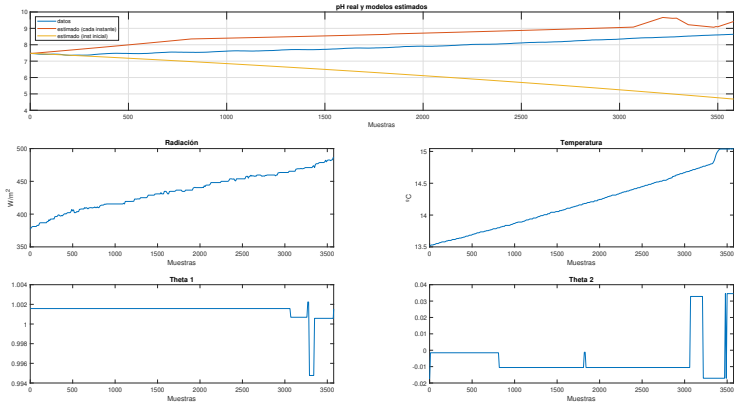


Figura 3.13: Estimación defectuosa de la respuesta libre - 2

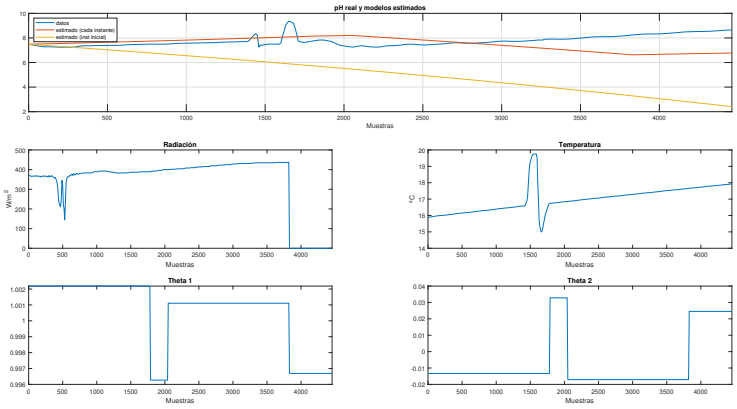


Figura 3.14: Estimación defectuosa de la respuesta libre - 3

Por esta razón se ha intentado obtener otro tipo de modelo para la respuesta libre haciendo uso del *System Identification Toolbox*.

- **Modelo con *System Identification Toolbox*:**

  En vista de que el modelo obtenido anteriormente por mínimos cuadrados no permitía obtener una buena estimación, se ha decidido obtener modelos de segundo orden haciendo uso de la aplicación *System Identification Toolbox* de MATLAB, descrita en la sección 2.4.1.

  Una vez más, como se debían tratar muchos datos, se realizó la estimación mediante unas pocas líneas de código, mostradas a continuación.

```
nx = 2;   % Orden del modelo
Ts = 10;  % Tiempo de muestreo

for i = 1:length(Intervalos_L_pH(1,:))
    final = find(Intervalos_L_pH(:,i)>0,1,'last');
    data = iddata(Intervalos_L_pH(tr:final,i),[],Ts);
    data.Tstart = 0;
    dataset{i} = data;
    %%% Opcion 1: PEM %%%
    modelo = pem(dataset{i},nx,'Ts',0);

    %%% Opcion 2: N4SID %%%
    % opt = ssestOptions('InitializeMethod','n4sid','Focus','prediction
        ','EnforceStability',false)
```

```
14      % modelo = ssest(data,nx,opt)
15
16      modeloset{i} = modelo;
17      figure(i+1)
18      compare(dataset{i},modeloset{i})
19  end
```

En primer lugar se define el objeto *iddata* del siguiente modo:

data = iddata (Salida, [], Ts)

donde Salida es la salida del sistema (pH) y Ts el tiempo de muestreo. El segundo parámetro corresponde a la entrada del sistema, al colocar [ ] se indica que no existe y por lo tanto se tratarán los datos como series temporales.

Una vez definidos los datos, se puede estimar el modelo utilizando los dos métodos de estimación explicados en la sección 2.4.1. Ambos consiguen un buen ajuste, tal como se muestra en las figuras 3.15 y 3.16.

Sin embargo, este método de modelado no es válido para el objetivo del trabajo, con él no se puede realizar un estimador de pH. Esto se debe a que, para series temporales el modelo en espacio de estados estimado es de la siguiente forma:

$$dx/dt = Ax(t) + ke(t) \tag{3.8}$$

$$y(t) = Cx(t) + e(t) \tag{3.9}$$

donde A y C son las matrices que se quieren calcular y k es la componente de perturbación, la cual no puede evitarse con los métodos de estimación utilizados cuando se trata de series temporales.

Esto genera el inconveniente de que el modelo se realimenta constantemente del error. Es decir, el pH se estimaría con la suma del pH estimado en el instante anterior más el error entre el pH estimado y el pH medido por el sensor. Por lo tanto, este método de modelado no ha podido desarrollarse con el fin de utilizarlo para el estimador de pH.

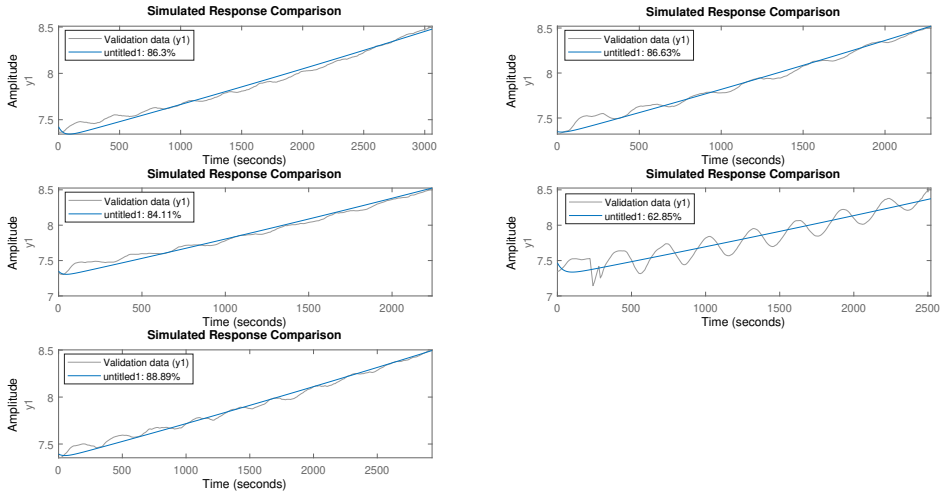


Figura 3.15: Validación de modelos de respuesta libre - Método PEM

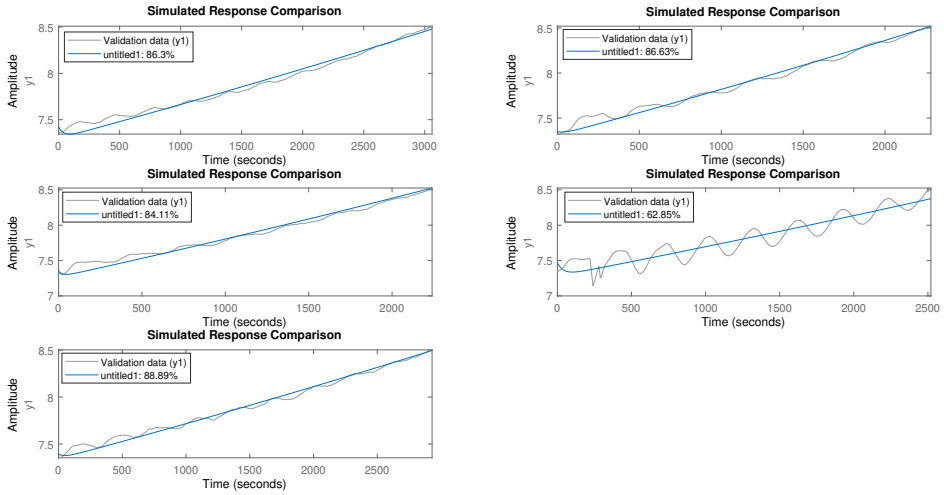


Figura 3.16: Validación de modelos de respuesta libre - Método N4SID

Una vez intentado obtener un modelo para la respuesta libre por dos vías distintas, se observa que para la implementación del estimador la opción que se debe utilizar es la obtenida mediante ajuste por mínimos cuadrados, a pesar de no haber conseguido un comportamiento óptimo.

3.3. Árboles de regresión

Una vez obtenidos todos los modelos comentados en la sección 3.2, se han entrenado los árboles de regresión necesarios para realizar la estimación de los parámetros de los modelos. Para dicho fin, se utilizaron las líneas de código descritas en la sección 2.4.2.

A continuación, se muestran las líneas de código utilizadas para obtener los árboles de regresión correspondientes a la ganancia estática y la constante de tiempo de los modelos para la respuesta forzada del sistema, durante la inyección de CO_2. Estas líneas son análogas a las utilizadas para la obtención de los parámetros de la respuesta libre del sistema.

```
%% SCRIPT PARA OBTENCION DE ARBOLES DE REGRESION DE RESPUESTA FORZADA
clear; close all; clc; warning off;

% Se lee el excel con todos los datos.
excel = strcat(pwd,'\Respuestas_Forzadas_Retardo_Fijo.csv');

comma2point_overwrite(excel)
opts = detectImportOptions(excel);
% Se seleccionan las variables de interes.
opts.SelectedVariableNames = {'Dia','Mes','Caudal','Ganancia','CteTiempo','
    RadiacionMedia','TemperaturaMedia','NivelMedio'};

% Se crea la tabla 'datos'.
datos = readtable(excel,opts);

% Se extraen de la tabla datos las variables de interes
caudal = datos.Caudal(1:end-1);
k = datos.Ganancia(1:end-1);
tau = datos.CteTiempo(1:end-1);
MediaRad = datos.RadiacionMedia(1:end-1);
MediaTemp = datos.TemperaturaMedia(1:end-1);
MediaNivel = datos.NivelMedio(1:end-1);

% Matriz de predictores
predictores = [MediaRad,MediaTemp,MediaNivel];

% Entrenamiento de los arboles
TreeModelk = fitrtree(predictores,k,'PredictorNames',{'Rad','Temp','Niv'},'
    ResponseName','K');
TreeModeltau = fitrtree(predictores,tau,'PredictorNames',{'Rad','Temp','Niv
    '},'ResponseName','tau');

```

```
30 % Se guardan los modelos para utilizar en la estimacion.
31 save('Modelos_Regresion.mat',"TreeModeltau","TreeModelk")
32
33 % Para ver la importancia de los predictores en cada modelo.
34 imp = predictorImportance(TreeModeltau);
35 figure;
36 subplot(1,2,1)
37 bar(imp);
38 title('Predictor Importance Estimates - tau');
39 ylabel('Estimates');
40 xlabel('Predictors');
41 h = gca;
42 h.XTickLabel = TreeModeltau.PredictorNames;
43 h.XTickLabelRotation = 45;
44 h.TickLabelInterpreter = 'none';
45
46 imp = predictorImportance(TreeModelk_sinCO2);
47 subplot(1,2,2)
48 bar(imp);
49 title('Predictor Importance Estimates - k');
50 ylabel('Estimates');
51 xlabel('Predictors');
52 h = gca;
53 h.XTickLabel = TreeModelk_sinCO2.PredictorNames;
54 h.XTickLabelRotation = 45;
55 h.TickLabelInterpreter = 'none';
56
57 % Para ver los modelos graficamente.
58 view(TreeModelk_sinCO2,'Mode','graph')
59 view(TreeModeltau_sinCO2,'Mode','graph')
```

En primer lugar, se lee el archivo *Excel* donde se han almacenado los datos correspondientes a los parámetros del modelo y las condiciones a las que se encuentra sometido el sistema en cada medio. Para el caso de la respuesta forzada, son los que se observan en la figura 3.7. A continuación, se crea la matriz de predictores, donde cada columna corresponde a un predictor distinto y cada fila es la combinación de los tres predictores para un momento dado. A continuación, se estiman los dos árboles de regresión con el comando explicado en la sección 2.4.2 (líneas 27 y 28) y se guardan, puesto que se necesitarán posteriormente para implementar el estimador de pH y el controlador de parámetros adaptativos. Por último, se observan gráficamente la importancia de cada predictor para cada árbol de regresión y el formato de cada árbol.

Para el caso de la respuesta forzada, los parámetros han sido estimados con dos árboles distintos. Por un lado, un árbol que utiliza como predictores la radiación global y la temperatura y el nivel del medio. Por otro, uno que utiliza, además de los predictores anteriores, el caudal de CO_2 inyectado. Como se verá en la sección 4.1.1, los resultados obtenidos con el segundo de ellos son mejores, presentando el pH estimado un menor error que el obtenido con el primer árbol.

Sin embargo, a la hora de implementar el estimador y el control adaptativo se busca realizar una estimación con un horizonte de aproximadamente 300 segundos, lo cual equivale a 10 muestras utilizando un tiempo de muestreo de 30 segundos. Además, tras estimar los parámetros se calculará el nuevo caudal de CO_2 que se debe inyectar para conseguir que el pH se mantenga en la referencia. Por estas dos razones expuestas, independientemente de cuál de los dos árboles brinde mejor resultado, para fines de control deberá ser utilizado aquel que no incluye el caudal de CO_2 como predictor, con el fin de evitar cambios permanentes en los parámetros del controlador..

A continuación, en las figuras 3.17 y 3.18 se muestran los árboles de regresión obtenidos para la constante de tiempo y la ganancia estática asociadas a la respuesta forzada, utilizando como predictores la radiación, la temperatura y el nivel del medio. Además, en la figura 3.19 se observa la importancia de cada predictor en la estimación de τ y k. Se puede observar que la temperatura y el nivel son cruciales para la predicción de la ganancia, mientras que para estimar el valor de la constante de tiempo presentan mayor importancia la radiación global y la temperatura del medio.

A continuación, en las figuras 3.20 y 3.21 se muestran los árboles de regresión obtenidos para θ_1 y la θ_2 asociadas a la respuesta libre. Además, en la figura 3.22 se observa la importancia de cada predictor en la estimación de θ_1 y θ_2. Se puede observar que, en ambos casos, el predictor más importante es la radiación global, seguido de la temperatura del medio.

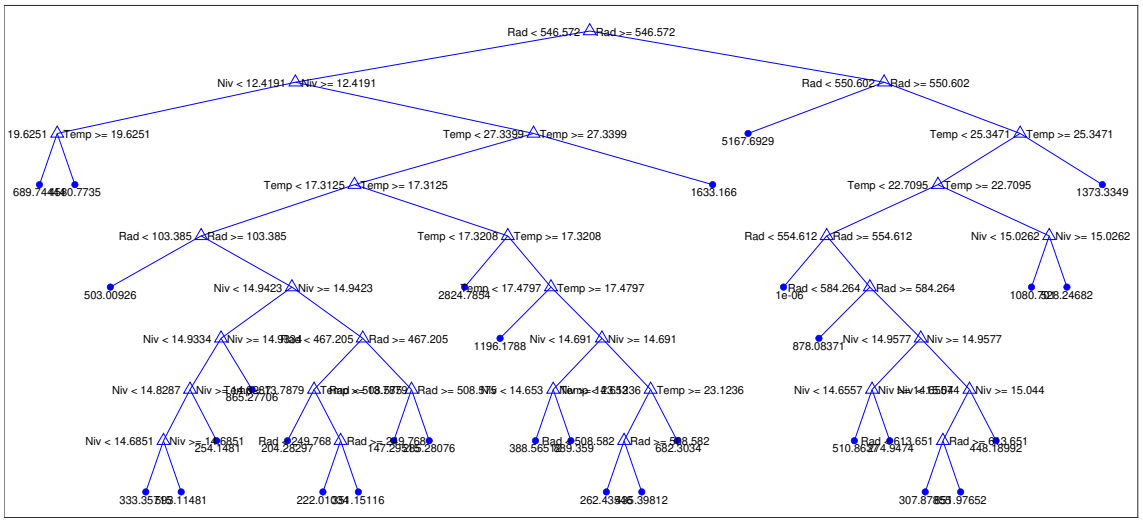


Figura 3.17: Árbol de regresión - Constante de tiempo

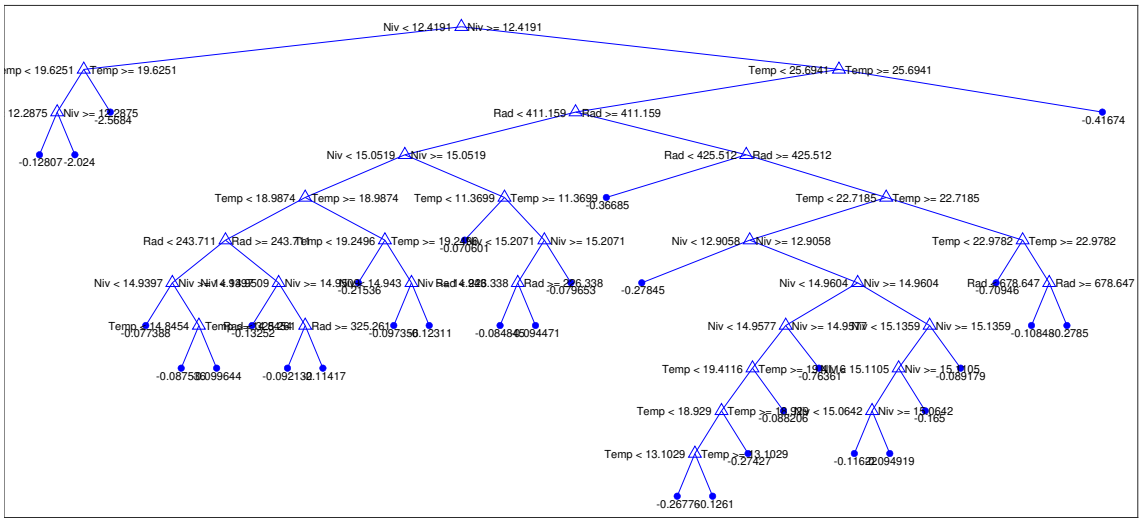


Figura 3.18: Árbol de regresión - Ganancia estática

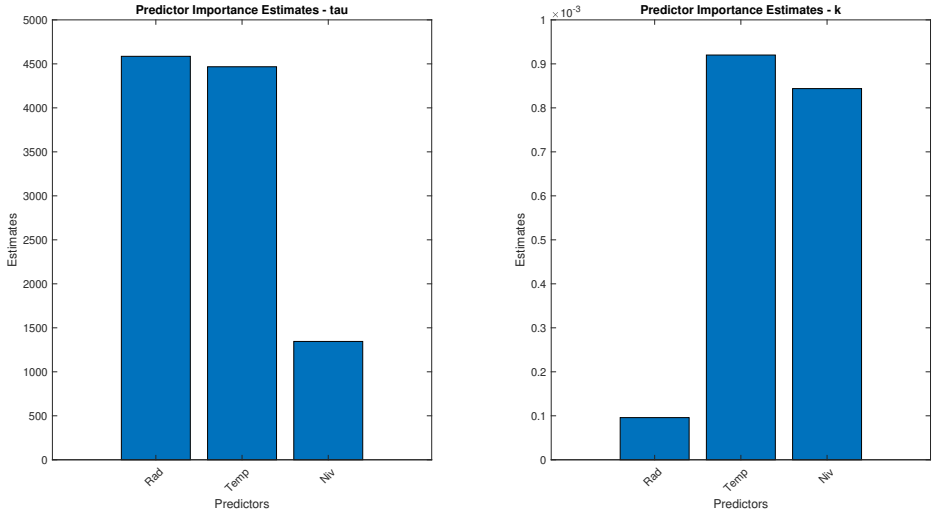


Figura 3.19: Importancia de los predictores - Respuesta forzada

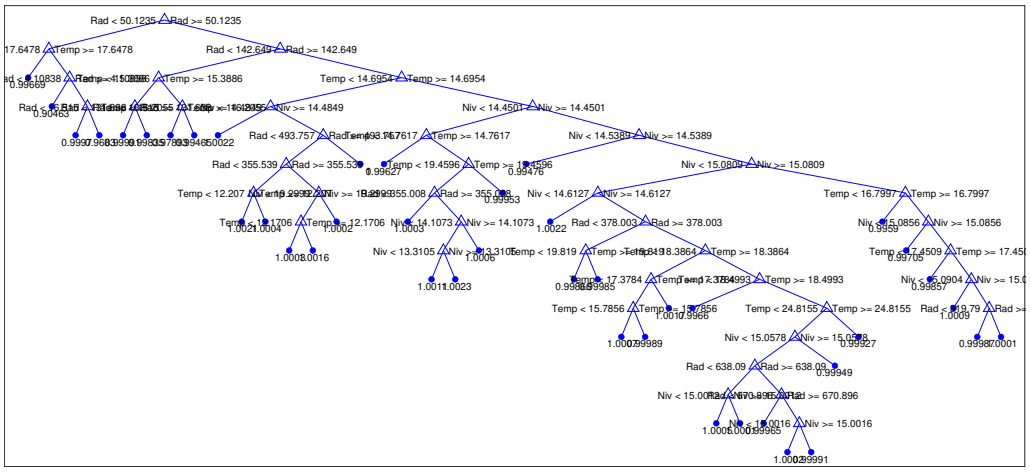


Figura 3.20: Árbol de regresión - θ_1

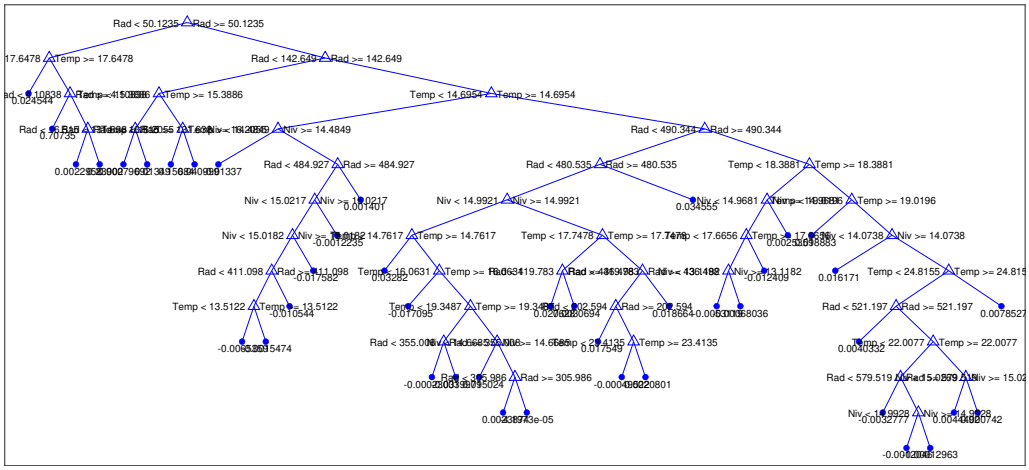


Figura 3.21: Árbol de regresión - θ_2

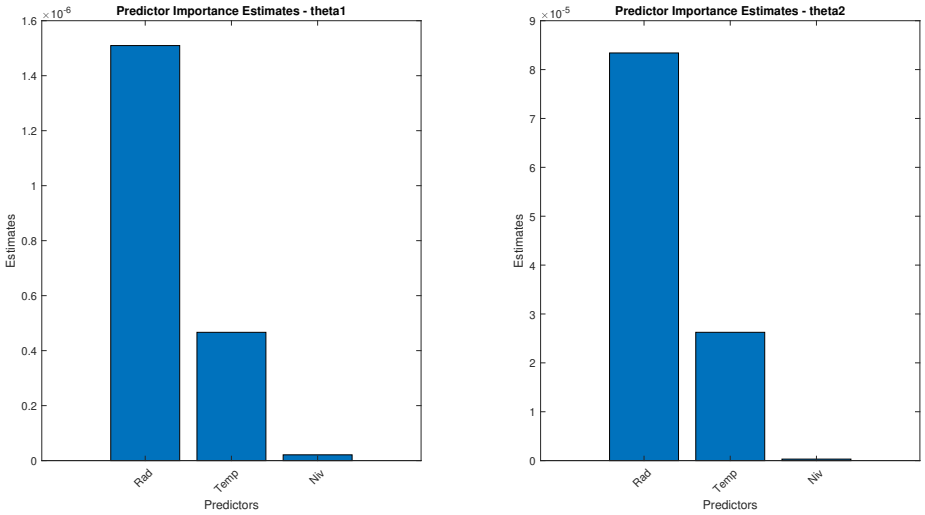


Figura 3.22: Importancia de los predictores - Respuesta libre

4

Análisis de resultados

4.1. Estimador de pH

Una vez entrenados los árboles de regresión para ambos tipos de respuesta, se implementó un algoritmo que, dados los valores de los predictores, fuera capaz de predecir el pH. De este modo, se conseguiría un sensor de pH en línea, que podría servir tanto para fines de control como para verificar el funcionamiento de las sondas.

Antes de llegar a implementar un estimador en línea en la planta real, se debieron seguir los siguientes pasos:

- En primer lugar, se intentó estimar el pH por tramos. Es decir, se buscaba estimar el pH en los mismos intervalos que se han utilizado para obtener los modelos.
- Una vez conseguido el punto anterior, se implementó el estimador para registros de datos de días completos anteriores. Es decir, se utilizaron los datos de los predictores de un día completo y se intentó estimar tanto la respuesta libre como la forzada, sabiendo los instantes y los valores de la inyección de CO_2.
- A continuación, se implementó el estimador en un simulador utilizando como planta virtual una red neuronal.
- Finalmente, se implementó en el reactor disponible en las instalaciones del Convenio UAL-IFAPA.

A continuación, se explica detalladamente el desarrollo de cada una de las etapas anteriormente enumeradas y se muestran los resultados correspondientes.

4.1.1. Estimación de pH por tramos

El objetivo de esta primer etapa era corroborar que con los modelos obtenidos se podría realizar una estimación en línea de los parámetros del modelo. Para ello, se han cogido los mismos intervalos utilizados para obtener los modelos y, tomando el valor inicial de pH y los valores instantáneos de radiación global, temperatura del medio y nivel del medio, se han intentado estimar dichas respuestas.

Con dicho fin, se ha realizado un *MATLAB script* que estime, por separado, cada tramo de respuesta libre y respuesta forzada para un día dado. Cabe destacar que esta prueba se ha realizado con los mismos días utilizados para obtener los modelos. A continuación, se desarrolla el procedimiento seguido y se muestran los resultados obtenidos para cada tipo de respuesta.

- **Respuesta forzada:**

  Se ha realizado la estimación de dos modos distintos. Por un lado, se ha realizado la estimación de los parámetros únicamente en el instante inicial de inyección y, con estos parámetros y los datos de la inyección de CO_2, se ha estimado la respuesta para todo el intervalo. Por otro lado, en cada muestra se ha realizado la estimación de los parámetros, de modo que, de acuerdo a los valores de los predictores en cada momento, los parámetros del modelo varían. Además, se han utilizado dos variantes de árboles de regresión:

  1. **Utilizando CO_2 como predictor:**
     Tal como se ha comentado en la sección 3.3 para la respuesta forzada, en primer lugar se ha realizado el estimador por tramos con un árbol entrenado utilizando como predictores la radiación global, la temperatura y el nivel del medio y el caudal de CO_2 inyectado. Se espera que las predicciones sean mejores puesto que el CO_2 presenta una gran importancia como estimador a la hora de predecir el valor de la constante de tiempo (figura 4.1).

     Los modelos obtenidos en este caso se ajustan considerablemente bien a los datos de pH reales, tanto calculando los parámetros instante a instante como calculándolos únicamente en el instante inicial. Por lo tanto, la opción más eficiente es estimar los parámetros una única vez, pues se reduce la complejidad y el tiempo de cómputo y los resultados son mejores. En las figuras 4.2 a 4.4 se observan los resultados de estimar el pH utilizando como predictores la radiación, la temperatura, el nivel y el CO_2 inyectado.

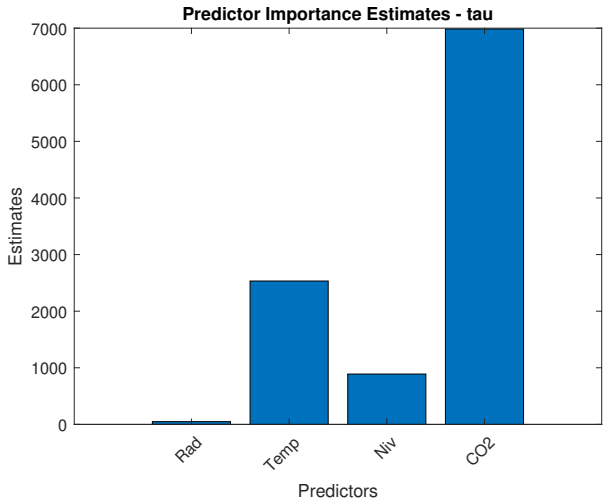


Figura 4.1: Importancia del CO_2 como predictor de la constante de tiempo

Se puede observar que estimando una única vez los parámetros el pH estimado se ajusta igual o mejor al pH real que estimándolos en cada instante de muestreo.

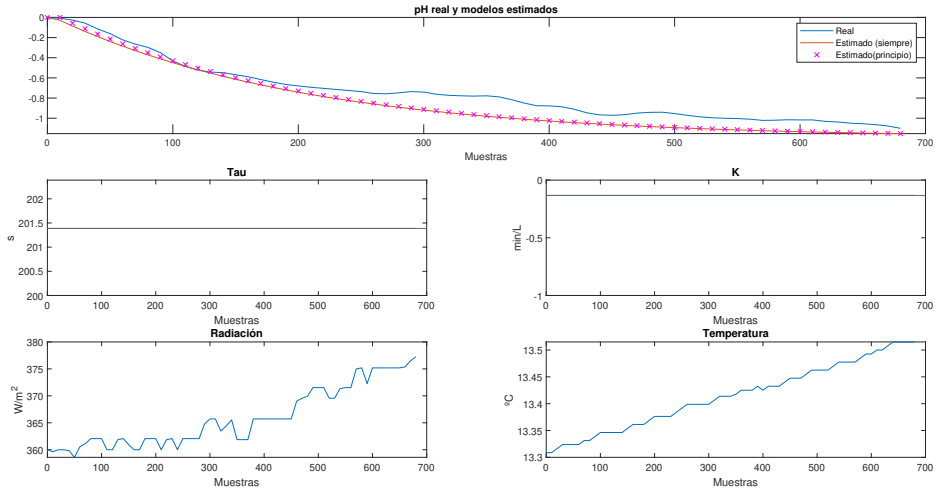


Figura 4.2: Estimación de la respuesta forzada utilizando CO_2 como predictor - 1

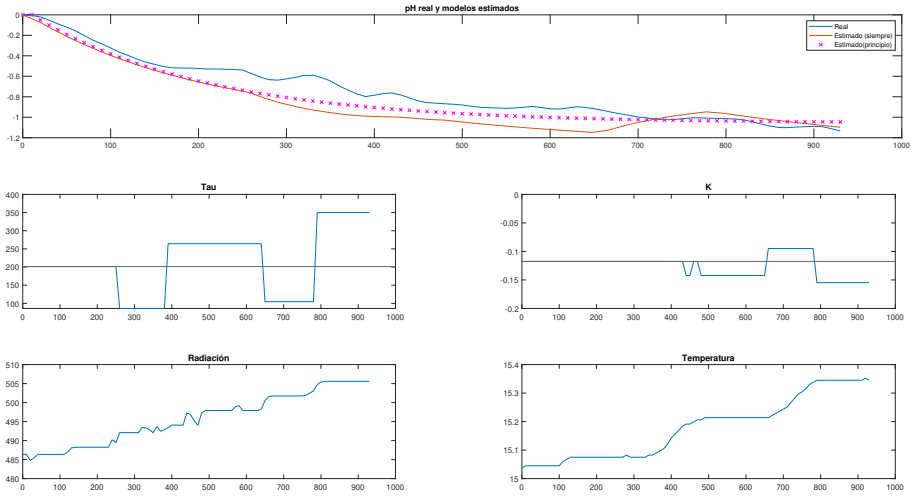


Figura 4.3: Estimación de la respuesta forzada utilizando CO_2 como predictor - 2

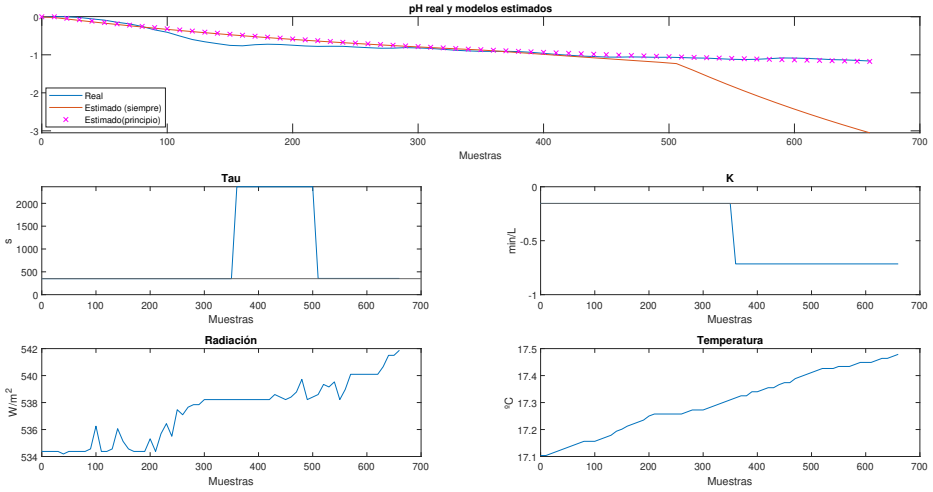


Figura 4.4: Estimación de la respuesta forzada utilizando CO_2 como predictor - 3

Sin embargo, como ya se ha comentado en la sección 3.3, es necesario realizar la estimación de los parámetros de forma independiente al CO_2. Esto se debe principalmente al objetivo final del trabajo, pues se busca realizar un control adaptativo cuyos parámetros se adapten a los del modelo según las condiciones, por lo que no puede haber dependencia con el CO_2.

Por lo tanto, una vez visto que es posible estimar la respuesta del sistema, se procede a realizar la estimación a tramos sin este predictor.

2. **Sin utilizar el CO_2 como predictor:**
   Una vez visto que es posible realizar la estimación del pH, se realizó el mismo procedimiento que en el punto 1, pero eliminando el caudal de CO_2 como parámetro predictor de k y τ. Se realizó, al igual que antes, comparando los resultados obtenidos al estimar únicamente al inicio y los obtenidos estimando instante a instante. Los resultados se muestran en las figuras 4.5 a 4.7, donde los intervalos son los mismos que las figuras 4.2 a 4.4.

   Se puede observar que los resultados empeoran respecto a la estimación obtenida utilizando el CO_2 como predictor, pero son suficientes para intentar implementarlo.

   El código utilizado para la implementación del estimador por tramos de la respuesta forzada se puede ver en la sección A.2.

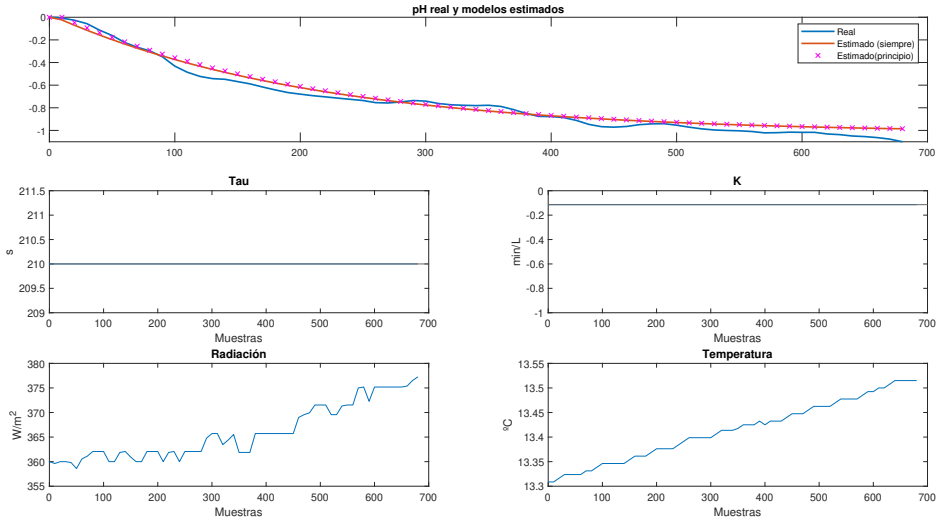


Figura 4.5: Estimación de la respuesta forzada sin CO_2 - 1

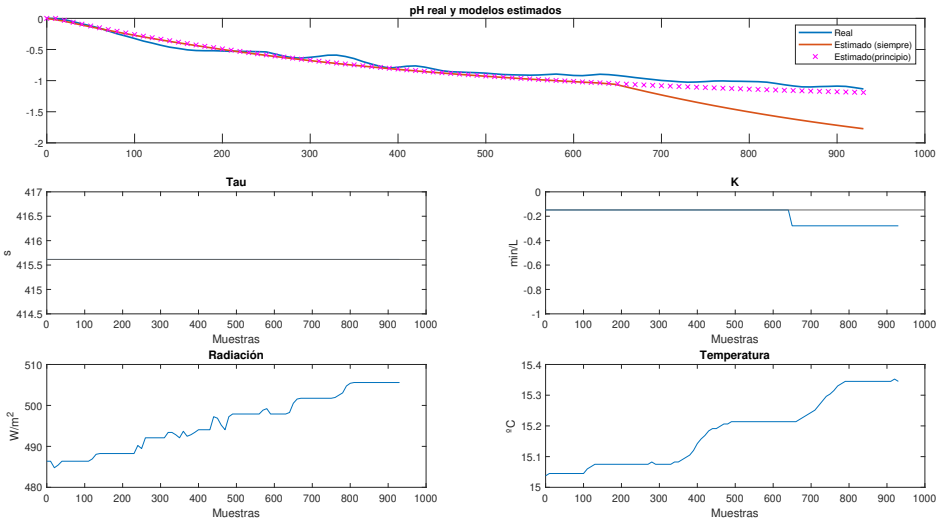


Figura 4.6: Estimación de la respuesta forzada sin CO_2 - 2

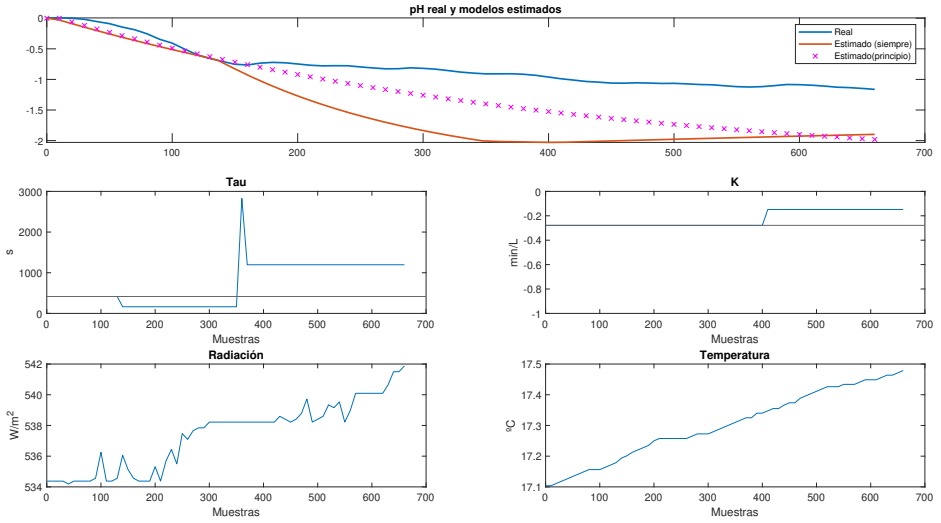


Figura 4.7: Estimación de la respuesta forzada sin CO_2 - 3

- **Respuesta libre:**

  Para estimar la respuesta libre se ha procedido del mismo modo que anteriormente, salvo que en este caso, no existe la posibilidad de incorporar el CO_2 como predictor, pues se pretende estudiar el comportamiento del sistema en su ausencia.

El código utilizado para implementar el estimador por tramos se presenta en la sección A.2.2 y los resultados de su ejecución se muestran en las figuras 4.8 a 4.10.

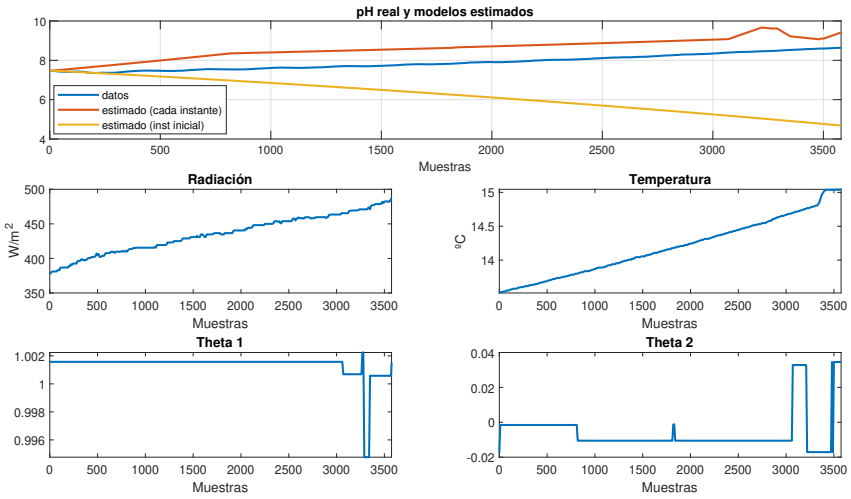


Figura 4.8: Estimación de la respuesta libre - 1

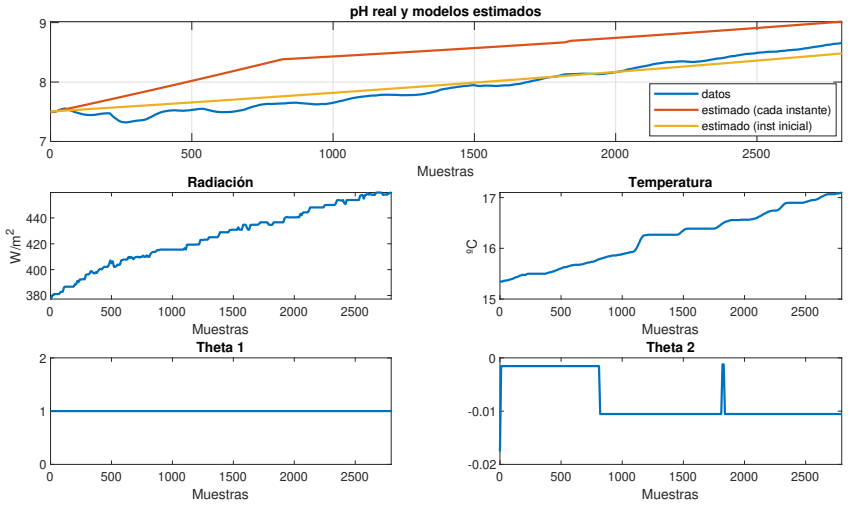


Figura 4.9: Estimación de la respuesta libre - 2

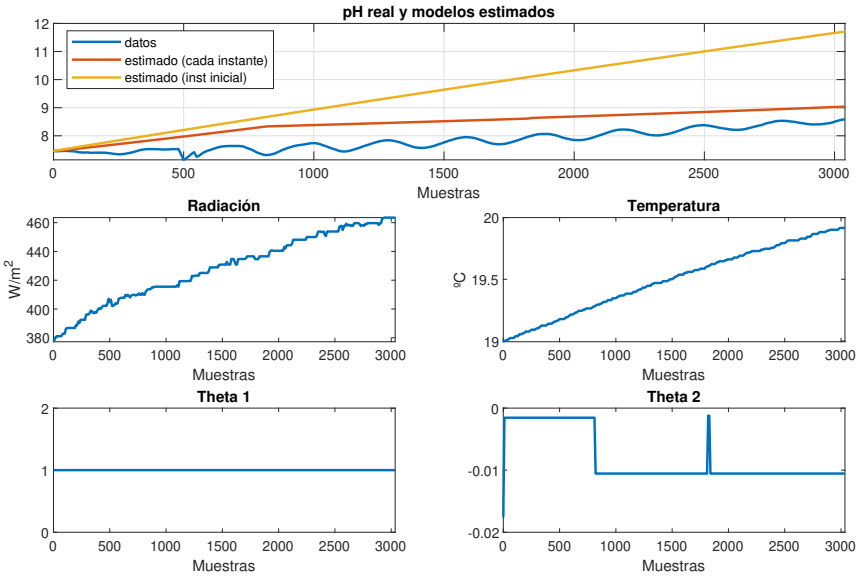


Figura 4.10: Estimación de la respuesta libre - 3

Se puede observar que las estimaciones obtenidas presentan mucho error respecto al pH medido por el sensor para ambos métodos de estimación. Por esta razón, se intentó obtener otro tipo de modelos para la respuesta libre (modelo en espacio de estados de segundo orden obtenido mediante el *System Identification Toolbox*) aunque, tal como se explicó en la sección 3.2, la estimación de la componente de perturbación imposibilita la estimación del pH tal como se ha planteado hasta el momento.

A pesar de los resultados insatisfactorios, se ha intentado realizar el estimador para registros de datos de días completos, cuyo procedimiento y resultados se describen en la sección 4.1.2.

4.1.2. Estimador de pH en registros de datos de días anteriores

En la presente sección, se muestran los resultados obtenidos al implementar el estimador en registros de datos correspondientes a días completos. Cabe destacar que la implementación se ha realizado por partes, las cuales se describen a continuación.

En primer lugar, se ha implementado la estimación del pH en presencia de inyección de CO_2, es decir, se ha intentado estimar la respuesta forzada para un día completo. Para eso, se han realizado las siguientes líneas de código.

```
%% Estimacion de la respuesta forzada %%
for i = 1 :size(datos,1)
   if (datos.Hora(i) < '08:00:00') || (datos.Hora(i) > '21:00:00') %
       Estimacion desde las 8 am hasta las 21 hs.
      pHEstimado(1,i) = datos.pHRW23(i);
   else
      % Si no hay inyeccion de CO2 se lee el dato del sensor.
      if caudalCO2(1,i-tr) == 0
          pHEstimado(1,i) = datos.pHRW23(i);
      % Si hay inyeccion de CO2 realizo la estimacion.
      else
         if caudalCO2(1,i-tr) - caudalCO2(1,i-tr-1) > 0.5
            k(i) = predict(TreeModelk,[datos.RadGlobal(i),datos.TempRW2(i),
                datos.NivelRW2(i),caudalCO2(i-tr)]);
            tau(i) = predict(TreeModeltau,[datos.RadGlobal(i),datos.TempRW2
                (i),datos.NivelRW2(i),caudalCO2(i-tr)]);
            pto_op = pHEstimado(1,i-1);
            Gc = tf(k(i),[tau(i) 1],'InputDelay',tr*Tm);
            Gz = c2d(Gc,Tm);
            a = Gz.Numerator{:}(2);
            b = Gz.Denominator{:}(2);
            pHEstimado(1,i) = datos.pHRW23(i);
         else
            k(i) = k(i-1);
            tau(i) = tau(i-1);
            pHEstimado(1,i) = a*caudalCO2(1,i-tr-1) - b*(pHEstimado(1,i-1)-
                pto_op) + pto_op;
         end
      end
   end
end
```

En el trozo de código mostrado, se observa cómo se ha realizado la estimación del pH durante la inyección de CO_2. Los pasos seguidos son:

- Antes de las 08:00:00 h y luego de las 21:00:00 h no se realiza estimación, sino que el pH se obtiene de la lectura del sensor. Esto se debe a que no es habitual el control nocturno del sistema dado que por la falta de radiación solar las microalgas no realizan la fotosíntesis y los cambios en el pH durante la noche son despreciables.

- Si durante las horas de estimación el caudal de CO_2 es nulo no se realiza la estimación, sino que se lee el pH del sensor. Esto se debe a que, en esta primera instancia, se quiere comprobar la correcta estimación del pH durante la inyección de CO_2. Cabe destacar que esta condición tiene en cuenta el retardo, por lo que para estimar el pH en el instante k se lee el caudal de CO_2 en el instante $k - t_r$, siendo t_r el tiempo de retardo del sistema.
- Si el caudal no es nulo, existen dos casos:
  - Si es el primer instante de inyección, es decir, en el momento en el que pasó de haber un caudal nulo a un caudal no nulo, se realiza la estimación de k y τ de acuerdo a los predictores (en este caso se ha utilizado el CO_2 como predictor). A partir de los parámetros se construye la función de transferencia y se discretiza teniendo en cuenta el tiempo de muestreo.
  - Si la inyección de CO_2 lleva más de un instante de muestreo, se utilizan los parámetros calculados al inicio del intervalo y se estima el pH.

De este modo, para controles tipo todo/nada, se realiza la estimación de los parámetros únicamente en el instante inicial de inyección ya que, como se ha visto en la sección 4.1.1, los resultados son mejores que si se realiza la estimación instante a instante y se reduce el tiempo de cómputo.

Los resultados de implementar dichas líneas de código en días con control todo/nada son los que se muestran en las figuras 4.11 a 4.13, donde se observa que, salvo casos excepcionales, el modelo estimado se ajusta sin demasiado error a los datos reales de pH. En las figuras se observa el caudal inyectado gracias a un controlador todo/nada, la respuesta del pH y su estimación, y la variación de los parámetros del modelo, k y τ.

Cabe destacar que en esta primera implementación se ha realizado la estimación utilizando como predictores la radiación global, la temperatura y el nivel del medio y el caudal de CO_2. Al tratarse únicamente del estimador, esto no genera ningún inconveniente pero sí permite obtener una mejor estimación.

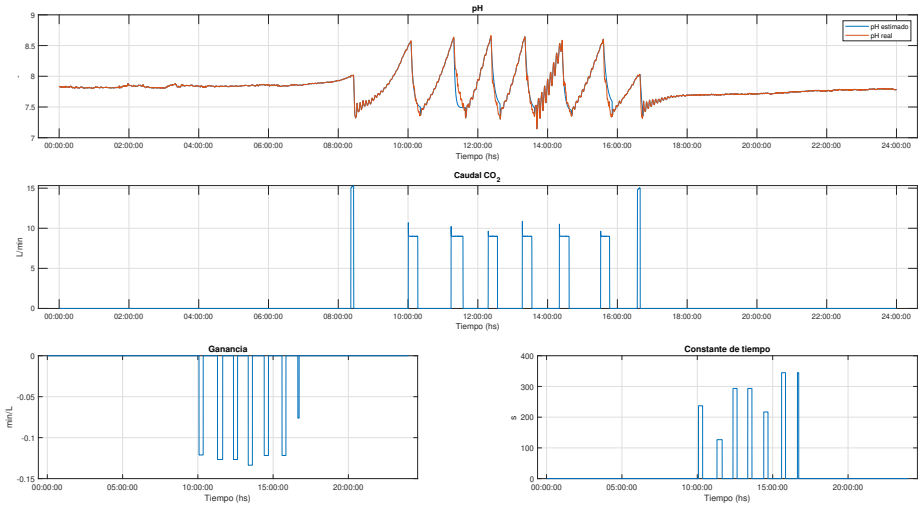


Figura 4.11: Estimación de la respuesta forzada - 1

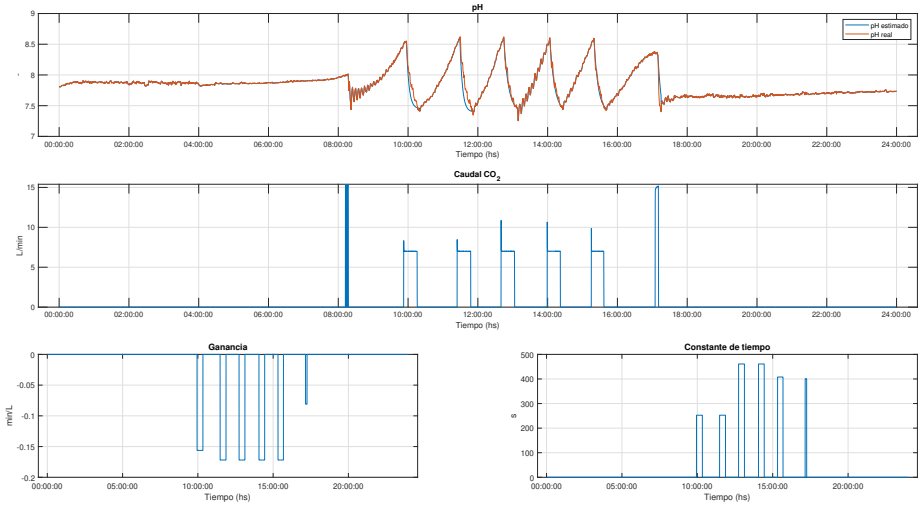


Figura 4.12: Estimación de la respuesta forzada - 2

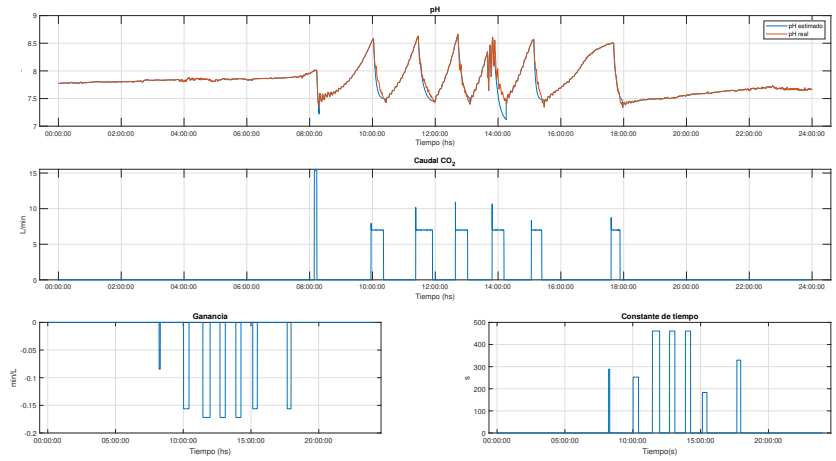


Figura 4.13: Estimación de la respuesta forzada - 3

A continuación, se muestran los resultados de implementar las mismas líneas de código pero utilizando como predictores de los parámetros únicamente la radiación global y la temperatura y el nivel del medio (figuras 4.14 a 4.16). Se observa que, para los mismos datos y las mismas condiciones a las que se encuentra sometido el sistema, las estimaciones empeoran considerablemente, pero siguen brindando resultados acordes al comportamiento del sistema.

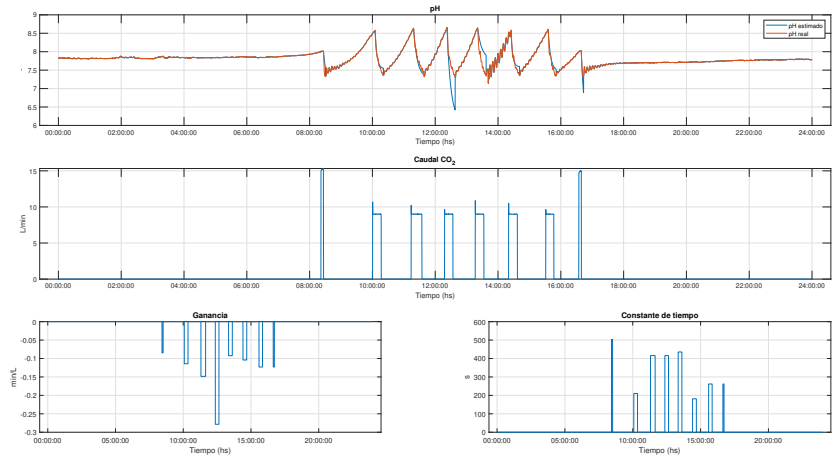


Figura 4.14: Estimación de la respuesta forzada sin CO_2 - 1

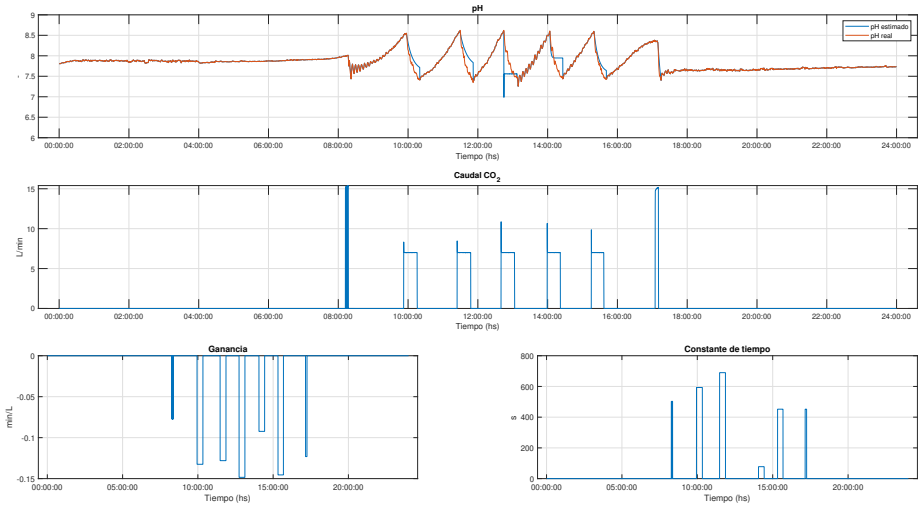


Figura 4.15: Estimación de la respuesta forzada sin CO_2 - 2

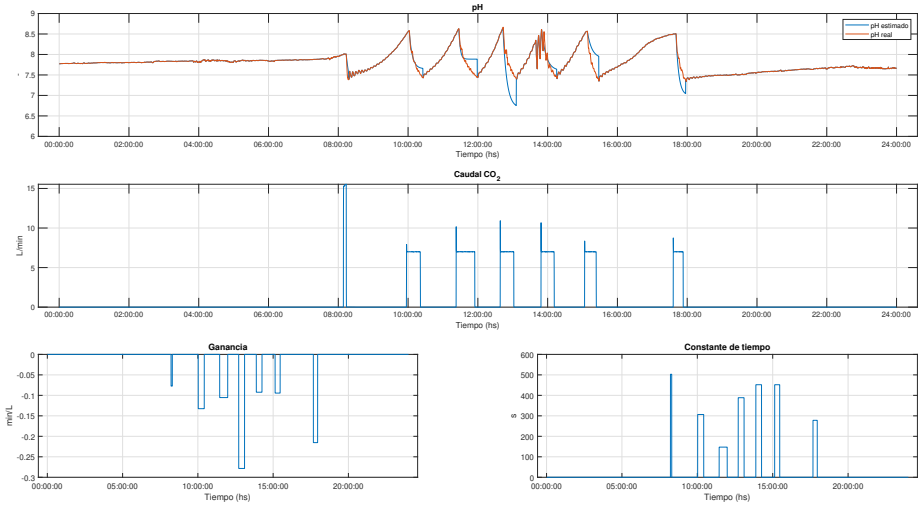


Figura 4.16: Estimación de la respuesta forzada sin CO_2 - 3

Una vez visto que, cuando se trata de controladores todo/nada, los resultados son mejores utilizando el CO_2 como predictor de los parámetros, se ha intentado estimar también la respuesta libre. Las líneas de código utilizadas para dicha implementación son las que se muestran en la sección A.3.1.

Sin embargo, tras multitud de pruebas no se ha conseguido una buena estimación de la respuesta libre. Esto se debe, probablemente, a la complejidad del sistema por tratarse de un sistema biológico con influencia de una gran cantidad de factores. En la figura 4.17 se muestra el resultado de implementar el estimador completo (se estiman ambas respuestas). Se observa que la respuesta forzada presenta una forma muy similar al pH real en casi todos los casos pero la respuesta libre presenta comportamientos que no pueden darse, como por ejemplo, el pH estimado alcanza valores por encima de su límite.

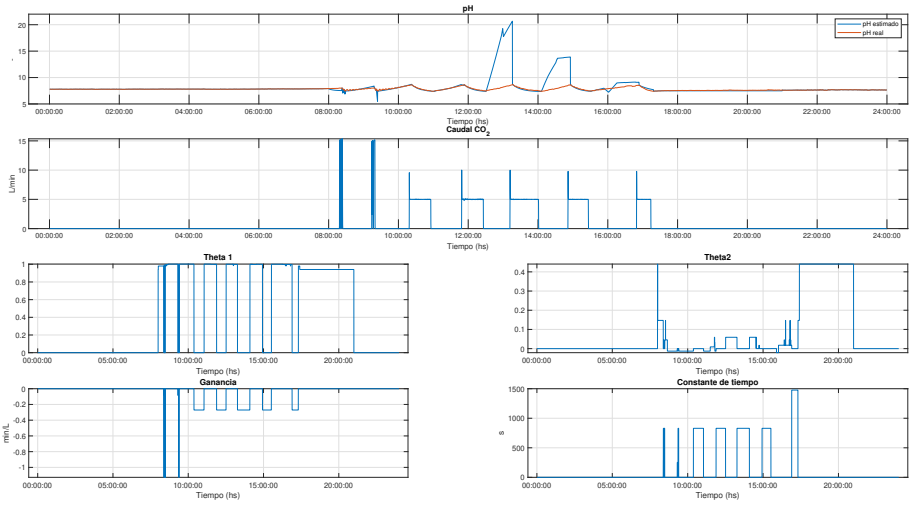


Figura 4.17: Estimación de la respuesta del pH a lo largo de un día completo - Controlador todo/nada

Es por esta razón que se ha intentado obtener un segundo modelo de segundo orden utilizando el *System Identification Toolbox* (sección 3.2). Sin embargo, como ya se ha explicado anteriormente, no sirve para estimación de pH en línea.

A continuación, se ha implementado el estimador en registros de días cuyo control fuera un controlador tipo PI, eliminando el caudal de CO_2 del conjunto de predictores, ya que el objetivo de esta etapa era comprobar si es posible realizar un control adaptativo con los parámetros del modelo que se estimen en cada instante.

El funcionamiento de estas estimaciones es el siguiente:

- Antes de las 06:00:00 h y luego de las 23:00:00 h no se realiza la estimación, al igual que en los momentos en que no hay inyección de CO_2 en el intervalo comprendido entre las 6 am y las 11 pm. En dichos momentos se lee el pH del sensor.
- Entre las 06:00:00 h y las 23:00:00 h se realiza la estimación del pH en los momentos donde hay inyección de CO_2 del siguiente modo:
  - Si hace t_r muestras había inyección de CO_2 se estima la ganancia y la constante de tiempo del sistema, se construye la ecuación que rige el comportamiento del pH y se realiza la estimación (t_r muestras hace referencia a lo que ocurría hace 270 segundos, tiempo de retardo del sistema).
  - Se mantiene la k y la τ estimadas durante *'refresh'* muestras y se calcula el pH con estos parámetros.
  - Una vez pasadas las muestras definidas por la variable *'refresh'*, se recalculan los parámetros y se repite el proceso.

La variable *'refresh'* se ha definido de forma que se estimen los parámetros cada 300 segundos, dicho valor se debe a los valores que toma la constante de tiempo de los modelos obtenidos (figura 4.21). Si se escoge un valor mayor, se observa con mayor claridad la forma de la respuesta de primer orden utilizada para el modelado.

El código utilizado se muestra en la sección A.3.2 y a continuación, se muestran los resultados obtenidos para cuatro días distintos (figuras 4.18 a 4.20).

Se puede observar que las estimaciones presentan valores y comportamientos muy similares a los del pH real en los tres casos. Además, se observa que los perfiles de radiación difieren de un día a otro y, aún así, el modelo estimado por los árboles de regresión se ajusta adecuadamente. En la zona inferior de la figura se puede observar cómo varían los parámetros del modelo a lo largo del día en función de las condiciones de las variables predictoras.

Los resultados obtenidos son muy satisfactorios y permiten proceder a implementar el estimador en un simulador basado en redes neuronales [13].

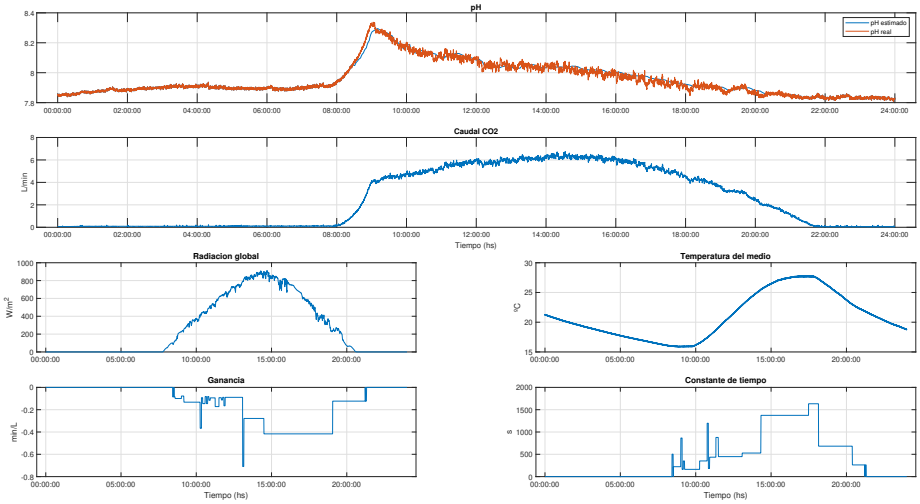


Figura 4.18: Estimación de pH con control tipo PI - 1

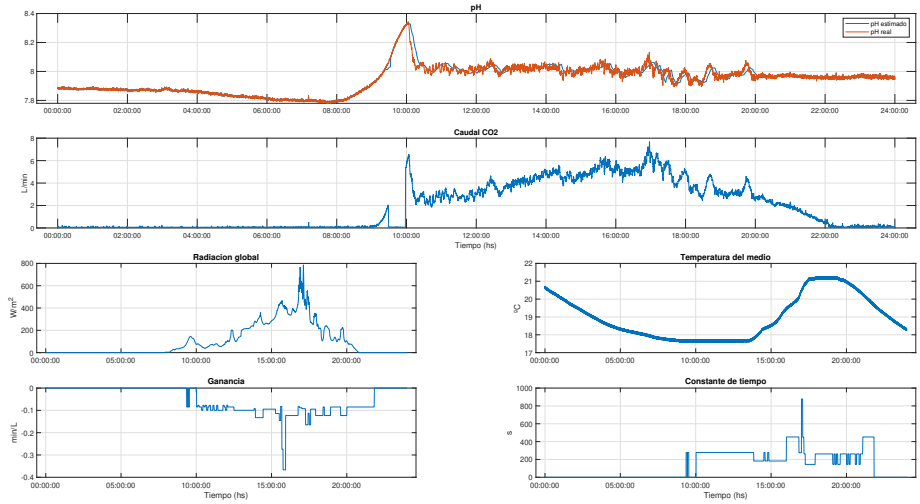


Figura 4.19: Estimación de pH con control tipo PI - 2

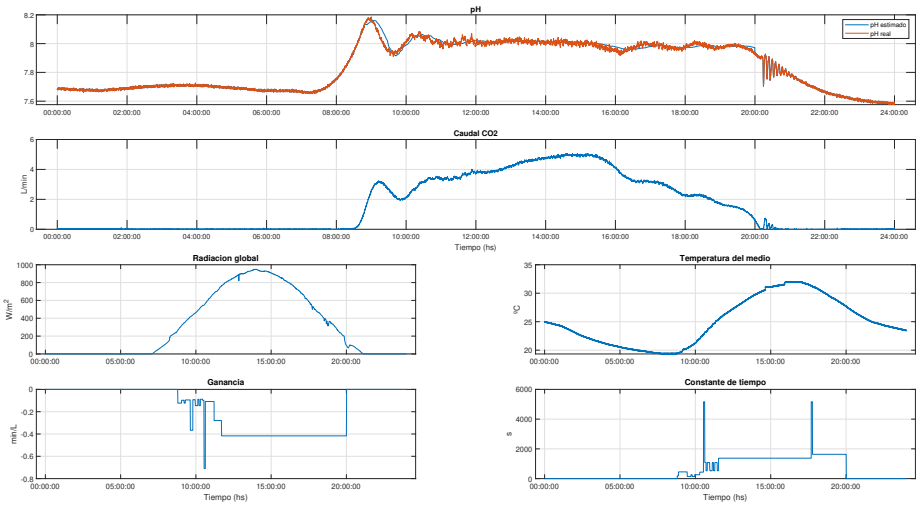


Figura 4.20: Estimación de pH con control tipo PI - 3

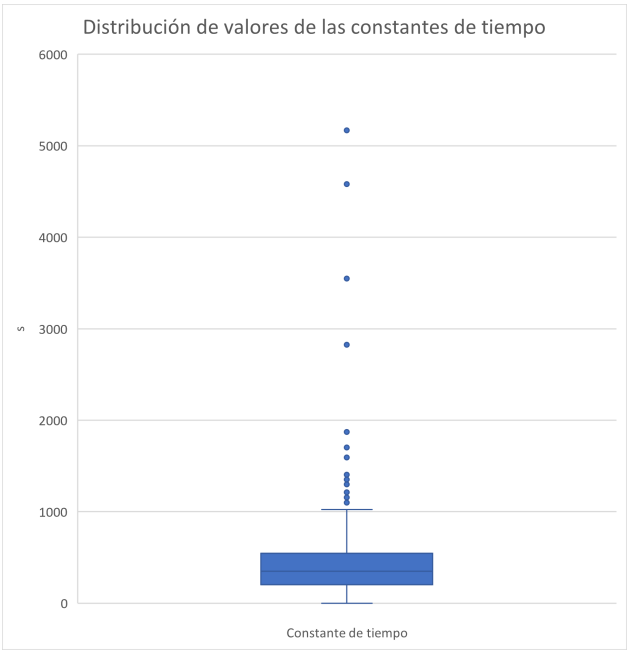


Figura 4.21: Valores tomados por las constantes de tiempo

4.1.3. Estimador de pH en simulador

A continuación, se ha implementado el estimador en un simulador que utiliza como planta virtual un modelo de red neuronal [13]. El código utilizado es análogo al utilizado en el estimador basado en registros con controlador tipo PI y se puede ver en detalle en la sección A.4.

Para dicha implementación se ha utilizado un controlador PI con parámetros fijos, obtenidos a partir de la ganancia y constante de tiempo medias de los modelos obtenidos en la primera fase del trabajo, por lo que el modelo de la planta utilizado es el que se muestra en la ecuación (4.1).

$$G(s) = \frac{k_{media}}{\tau_{media}s + 1}e^{-t_r s} = \frac{-0{,}16773}{524{,}2016s + 1}e^{-270s} \tag{4.1}$$

Una vez obtenido el modelo, se ha calculado el **tiempo de retardo medio** (o índice de controlabilidad). El mismo se calcula como se muestra en la ecuación (4.2), donde t_r es el tiempo de retardo del sistema y T_{rm} el tiempo de residencia medio ($T_{rm} = t_r + \tau$).

$$Q_{rn} = \frac{t_r}{T_{rm}} = \frac{t_r}{t_r + \tau} = \frac{270}{270 + 524{,}2016} = 0{,}34 \tag{4.2}$$

Un valor de Q_{rn} por debajo de 0.5 indica que se trata de un proceso con constante de tiempo dominante y que puede ser controlado mediante métodos clásicos. Por lo tanto, se ha diseñado un controlador con acción proporcional e integral por método λ, cuyo procedimiento se desarrolla en la sección 2.5. La función de transferencia de bucle cerrado es la mostrada en la ecuación 2.18, y los parámetros del controlador se calculan a partir de las expresiones recuadradas, de modo que resultan:

$$T_i = \tau = 524{,}2016 \text{ s} \tag{4.3}$$

$$\tau_{bc} = \frac{T_i}{kK_p} - t_r \Longrightarrow K_p = \frac{T_i}{k(\tau_{bc} + t_r)} \Longrightarrow K_p = -4{,}53 \text{ L/min} \tag{4.4}$$

donde, en la ecuación 4.4, $\tau_{bc} = 0{,}8\tau$.

El controlador resultante es el que se muestra en la ecuación 4.5.

$$C(s) = -4{,}53\left(\frac{524{,}2016s + 1}{524{,}2016s}\right) \tag{4.5}$$

El resultado de implementar el controlador anterior junto con el estimador de pH en el simulador se muestra en las figuras 4.22 a 4.24, donde se observa que se realiza un control adecuado del pH y los resultados de pH que brinda el estimador se asemejan suficientemente bien a los datos de pH predichos por la red neuronal.

Además, se puede observar cómo varían los parámetros del modelo (ganancia estática y constante de tiempo) a lo largo del día. Este trabajo es realizado por los árboles de regresión obtenidos, que actualizan la estimación de los parámetros cada 5 muestras (5 minutos).

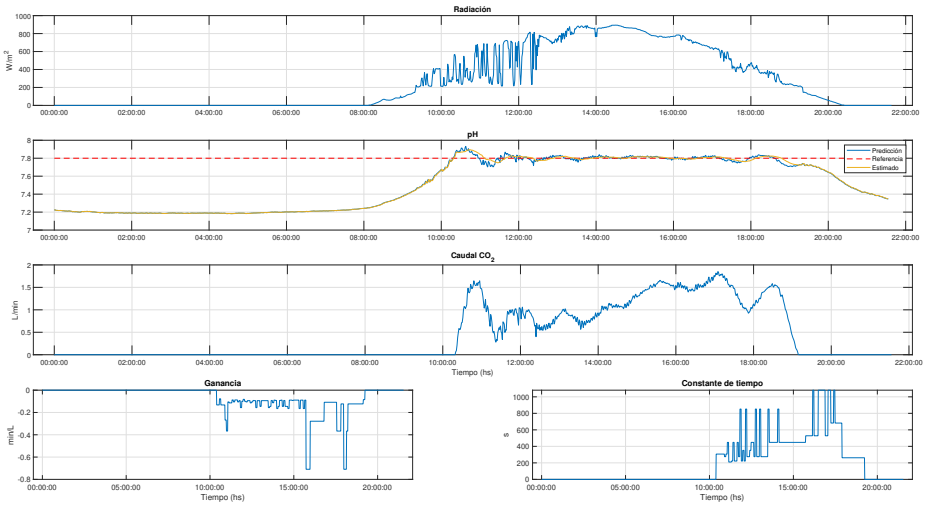


Figura 4.22: Estimación de pH en simulador con control tipo PI - 1

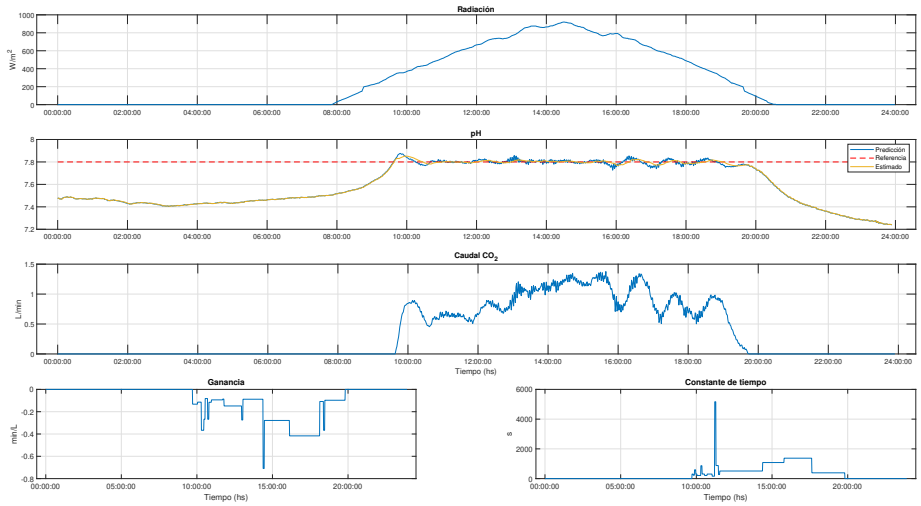


Figura 4.23: Estimación de pH en simulador con control tipo PI - 2

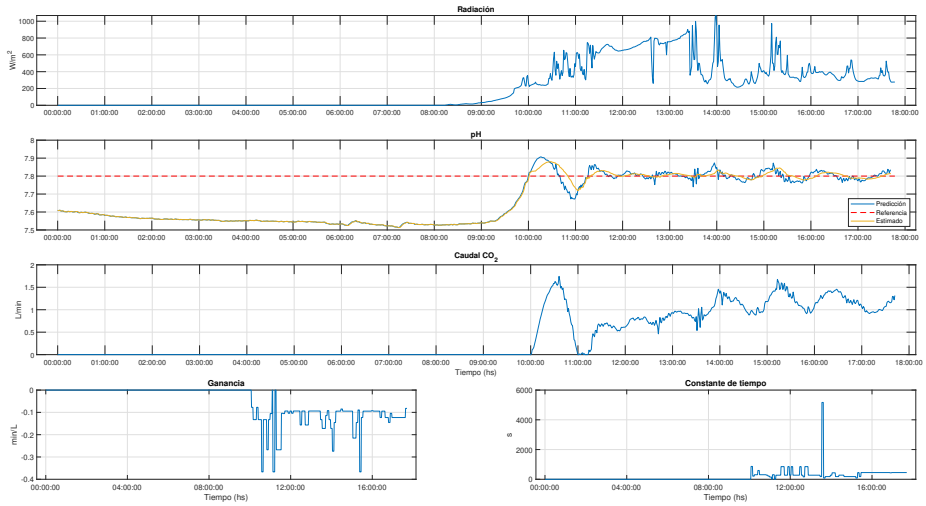


Figura 4.24: Estimación de pH en simulador con control tipo PI - 3

Una vez implementado el estimador y el control PI de parámetros fijos en el simulador, se calculan los índices clásicos de comportamiento (IAE, ISE, ITAE e ITSE) y el esfuerzo de control para analizar cuantitativamente los resultados obtenidos para rechazo a perturbaciones (tabla 4.1). En la tabla 4.2 se muestran el error cuadrático medio obtenido entre el pH estimado y el pH real.

Día	IAE	ISE	ITAE	ITSE	EC
1	18.4033	1.177	5.3646 e+03	343.0883	9.738
2	12.5410	0.6617	2.9095 e+03	153.5257	7.1776
3	11.1794	0.3493	3.488 e+03	108.9867	10.3465

Tabla 4.1: Índices de comportamiento de rechazo a perturbaciones para control fijo - Simulación

Día	RMSE
1	0.0208
2	0.0239
3	0.0180

Tabla 4.2: Error cuadrático medio entre pH estimado y pH real para control fijo - Simulación

4.1.4. Estimador de pH en el sistema real

Una vez que se comprobó el funcionamiento del estimador en simulación, se implementó el mismo controlador desarrollado en la sección 4.1.3 junto con el estimador de pH en el reactor real disponible en las instalaciones del Convenio UAL-IFAPA.

El código desarrollado para tal fin se muestra en la sección A.5. Se trata de una función que es llamada desde un *script* que se ejecuta continuamente, llamado "*startup*". Dicho script se ejecuta cada 30 segundos (tiempo de muestreo) y su funcionamiento es el siguiente:

- En primer lugar, se conecta mediante protocolo OPC DA al PLC y lee los datos correspondientes al estado de los reactores, la estación meteorológica y los actuadores.
- A continuación, se llama a todas las funciones que contengan algoritmos de control, ya sea control de pH, OD, nivel u otros parámetros. Todas ellas utilizan como entrada los datos obtenidos anteriormente.

- Cada función devuelve las referencias de caudal de CO_2 y de aire, un nombre representativo para su fácil identificación y tantas variables auxiliares resulte conveniente almacenar.
- Por último, se vuelve a realizar la conexión para escribir en el PLC las variables devueltas por las funciones anteriores.

Los resultados de implementar el controlador PI de parámetros fijos y el estimador de pH en el reactor se muestra en las figuras 4.25 y 4.26.

Se puede observar que el comportamiento de pH obtenido con el controlador diseñado es adecuado. Sin embargo, el pH estimado presenta mucho error con respecto al real, no pudiendo seguir su dinámica oscilatoria. A continuación, se muestran los índices de comportamiento obtenidos tanto para analizar el comportamiento de rechazo a perturbaciones (tabla 4.3) como la calibración del modelo respecto a la realidad (tabla 4.4).

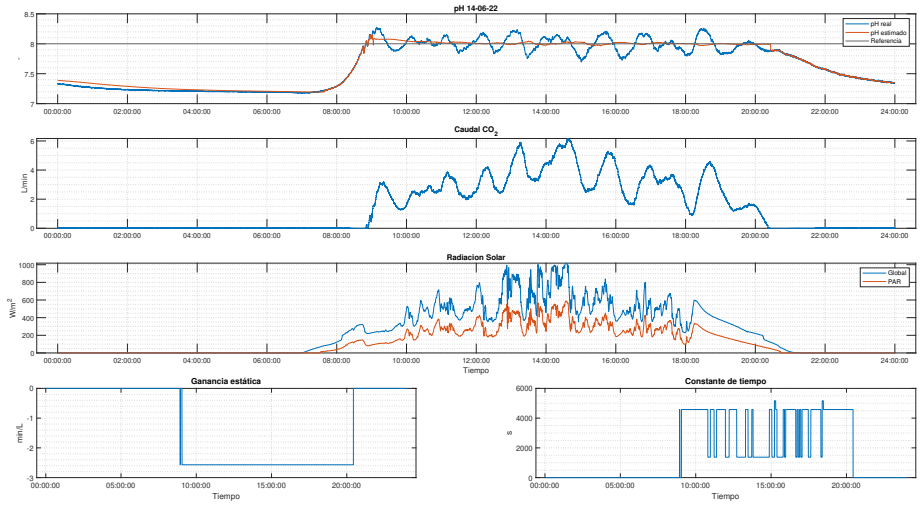


Figura 4.25: Estimación de pH y control PI fijo en reactor del Convenio UAL-IFAPA, 14 de Junio

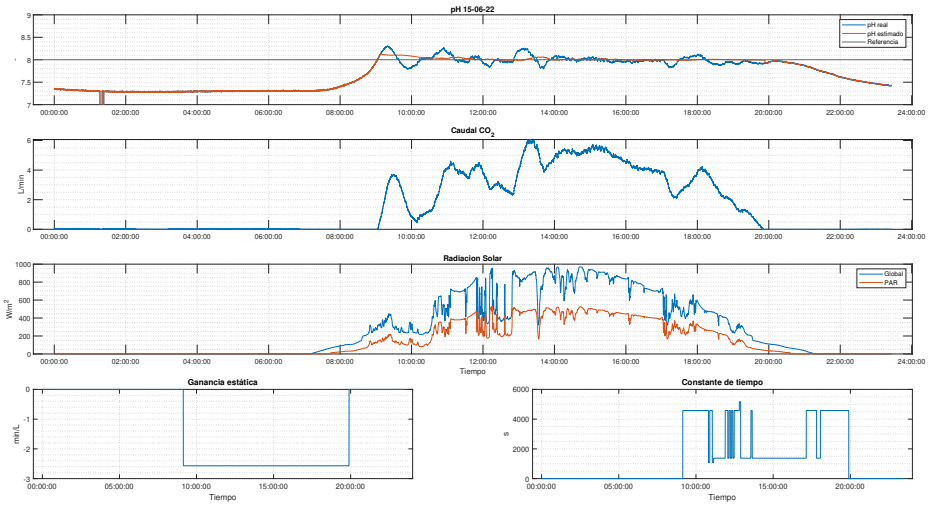


Figura 4.26: Estimación de pH y control PI fijo en reactor del Convenio UAL-IFAPA, 15 de Junio

Día	IAE	ISE	ITAE	ITSE	EC
14 junio	4010.2	589.4044	8.297 e+07	1.2525 e+07	525.8072
15 junio	2924.5	382.2602	4.821 e+07	4.8641 e+07	457.0744

Tabla 4.3: Índices de comportamiento de rechazo de perturbaciones - Controlador PI de parámetros fijos en reactor del Convenio UAL-IFAPA

Día	RMSE
14 de junio	0.1275
15 de junio	0.1

Tabla 4.4: Error cuadrático medio entre pH estimado y pH real para control fijo - Reactor del Convenio UAL-IFAPA

4.2. Algoritmos de control

Una vez comprobado el funcionamiento del estimador de pH se ha procedido a diseñar un algoritmo de control adaptativo, es decir, un controlador cuyos parámetros se ajusten al modelo que estima el árbol de regresión en cada momento.

Así, el controlador obtenido es de la misma forma que el obtenido en la sección 4.1.3. La diferencia reside en que ahora los valores de k y τ varían en función de las condiciones de radiación, temperatura del medio y nivel del medio, por lo que también variarán la ganancia proporcional y el tiempo integral del controlador.

$$G(s) = \frac{k}{\tau s + 1} e^{-270s} \tag{4.6}$$

$$C(s) = K_p \left(\frac{T_i s + 1}{T_i s} \right) = \frac{\tau}{k(0{,}8\tau + 270)} \left(\frac{\tau s + 1}{\tau} \right) \tag{4.7}$$

Dicha implementación se ha realizado tanto en simulación como en el reactor real, los resultados se muestran en las siguientes secciones.

4.2.1. Control adaptativo en simulación

El código utilizado es análogo al del estimador con controlador PI de parámetros fijos de la sección 4.1.3. La única diferencia radica en que, en este caso, se deberán eliminar las líneas 102 y 103 del código (sección A.4), de modo que K_p y T_i varíen en función de las variaciones sufridas por k y τ. Los resultados obtenidos se muestran en las figuras 4.27 a 4.29, teniendo en cuenta que los días en que se ha realizado el control son los mismos que los de la sección 4.1.3, por lo que se puede comparar el desempeño de ambos controladores.

Se puede observar que los resultados de control son muy buenos, pues se consigue obtener el pH en torno a la referencia establecida (pH de 7.8). Además, en la zona inferior de las figuras se refleja el cambio de ambos parámetros del controlador, implementado con su correspondiente transferencia sin saltos para que no haya cambios bruscos en la señal de control cuando hay una variación en el controlador. Los parámetros se actualizan cada cinco minutos.

En cuanto al estimador, su funcionamiento es similar al que se obtenía en simulación con un controlador PI de parámetros fijos.

Una vez visto que es posible realizar un control adaptativo a partir de los modelos de árbol de regresión obtenidos, se ha procedido a implementar esta misma estructura de control en el sistema real.

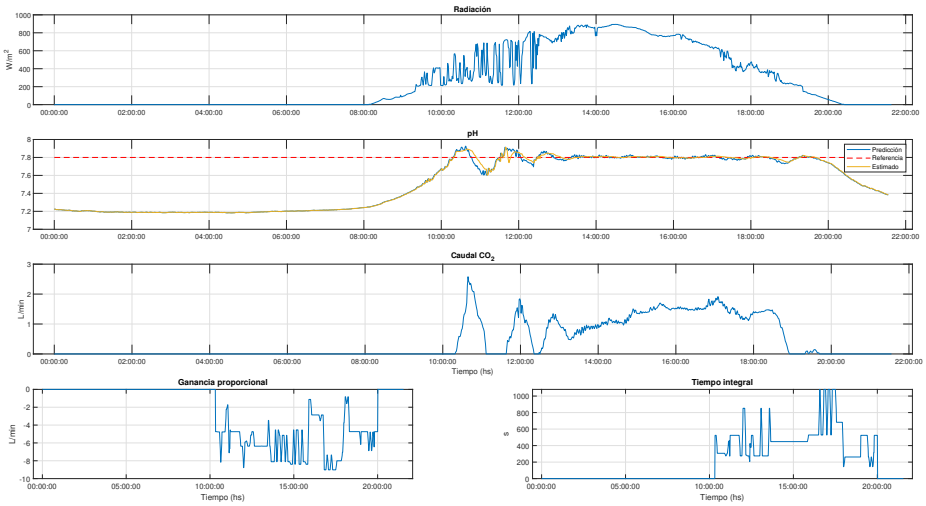


Figura 4.27: Control adaptativo en simulador - 1

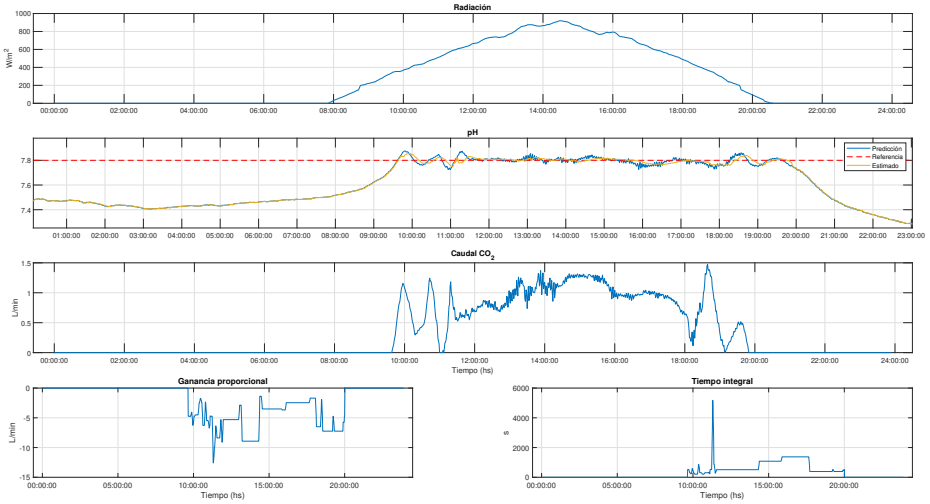


Figura 4.28: Control adaptativo en simulador - 2

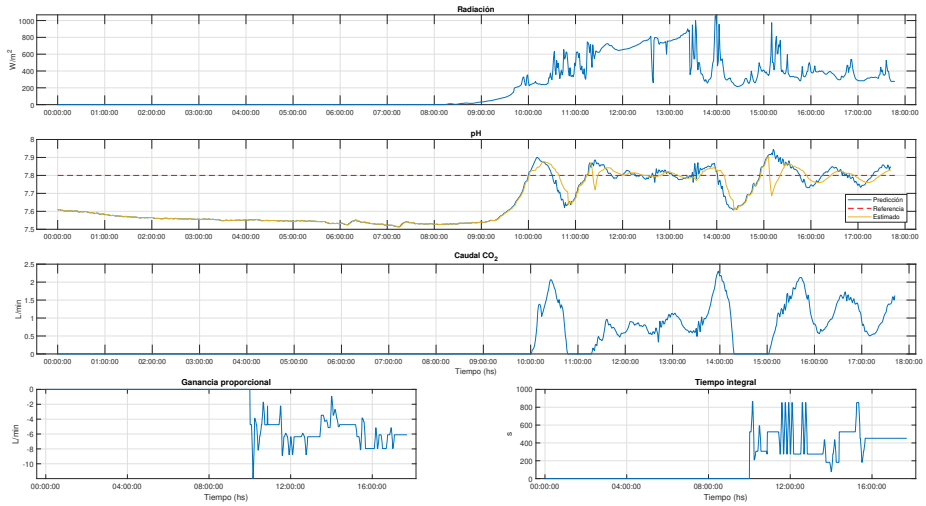


Figura 4.29: Control adaptativo en simulador - 3

Una vez implementado el controlador adaptativo en simulador para los mismos días que se ha implementado el controlador fijo, se han calculado los índices de comportamiento IAE, ISE, ITA e ITSE, además del esfuerzo de control, para cada uno de ellos para analizar de forma cuantitativa el comportamiento obtenido. Los mismos se observan en la tabla 4.5. Además, en la tabla 4.6 se muestra el error cuadrático medio del pH medido respecto al real para el control adaptativo implementado en simulación.

Día	IAE	ISE	ITAE	ITSE	EC
1	17.3405	1.3214	5.0548 e+03	385.1821	11.1577
2	23.9636	2.2505	5.5596 e+03	522.1106	11.1289
3	14.0494	0.5302	4.3834 e+03	165.4306	14.6339

Tabla 4.5: Índices de comportamiento de rechazo a perturbaciones para control adaptativo - Simulación

Día	RMSE
1	0.0274
2	0.0470
3	0.0226

Tabla 4.6: Error cuadrático medio entre pH estimado y pH real para control adaptativo - Simulación

Comparando las tablas 4.1 y 4.5, se puede observar que los índices de comportamiento del controlador presentan valores similares para ambos tipos de controles diseñados, pero siendo en su mayoría mejores para el controlador de parámetros fijos. Además, si se comparan las tablas 4.2 y 4.6 y se analizan los índices de comportamiento del estimador de pH, se puede observar que el pH estimado se acerca considerablemente más al pH real en el caso del control de parámetros fijos. No obstante, el control adaptativo presenta un comportamiento muy adecuado y la estimación de pH para el mismo también anima a realizar la implementación en el reactor disponible en las instalaciones del Convenio UAL-IFAPA.

4.2.2. Control adaptativo en el sistema real

Una vez obtenidos buenos resultados para el control adaptativo en simulación, se ha procedido a implementar en el reactor disponible en las instalaciones del Convenio UAL-IFAPA. El código desarrollado para tal fin se muestra en la sección A.6 y su funcionamiento es el mismo que el explicado en la sección 4.1.4. Los resultados obtenidos se muestran en las figuras 4.30 a 4.32.

La implementación del control adaptativo se realizó en la mañana del día 16 de junio, por lo que los resultados de dicho día se muestran a partir del instante en que comenzó a ejecutarse el mismo. Se puede observar que el pH comienza en aproximadamente 8.9 y el sistema tarda alrededor de 45 minutos en llevarlo al valor de la referencia. Al tratarse de un problema de regulación se podría haber escogido $\tau_{BC} = 0{,}2\tau$ de modo que la ganancia proporcional fuera mayor y la respuesta del sistema fuera más rápida. Sin embargo, en esta primera prueba ha querido desarrollarse un controlador más conservador.

Para evaluar cuantitativamente los resultados obtenidos, en la tabla 4.7 se muestran los índices de comportamiento obtenidos para el control implementados. Además, en la tabla 4.7 se muestra el $RMSE$ para el pH estimado respecto al pH medido.

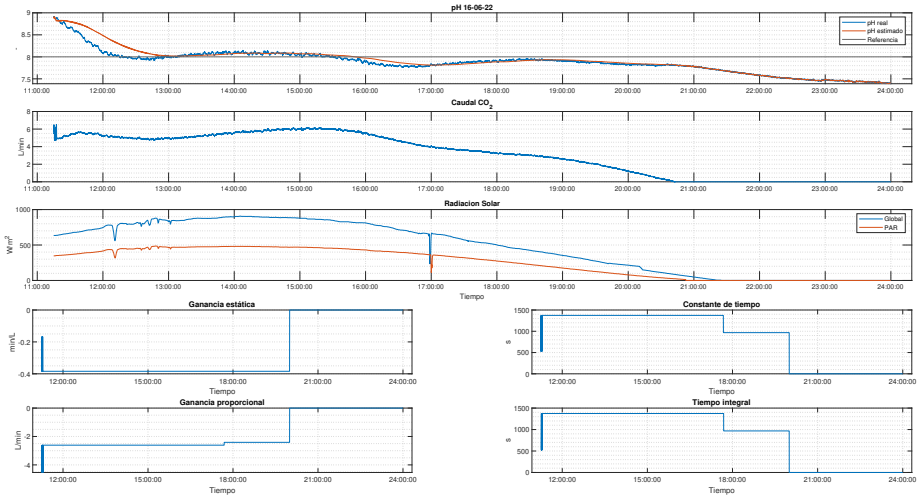


Figura 4.30: Control adaptativo en reactor del Convenio UAL-IFAPA, 16 de junio

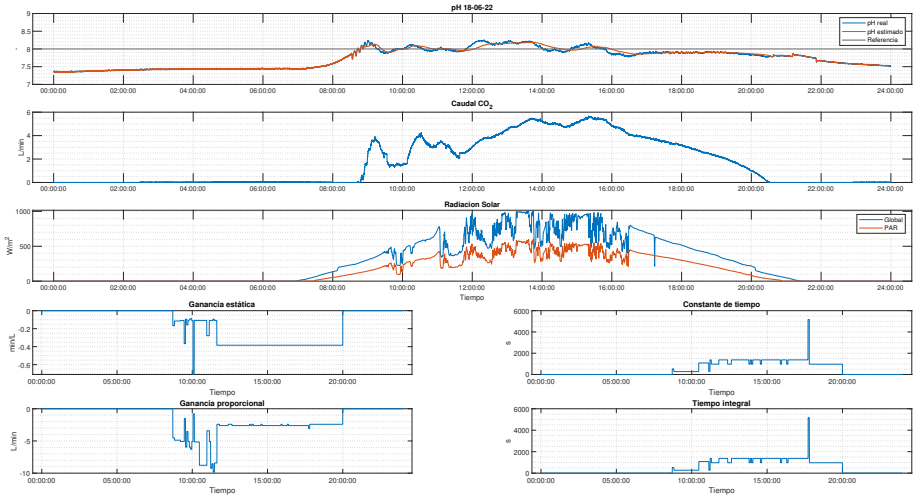


Figura 4.31: Control adaptativo en reactor del Convenio UAL-IFAPA, 18 de junio

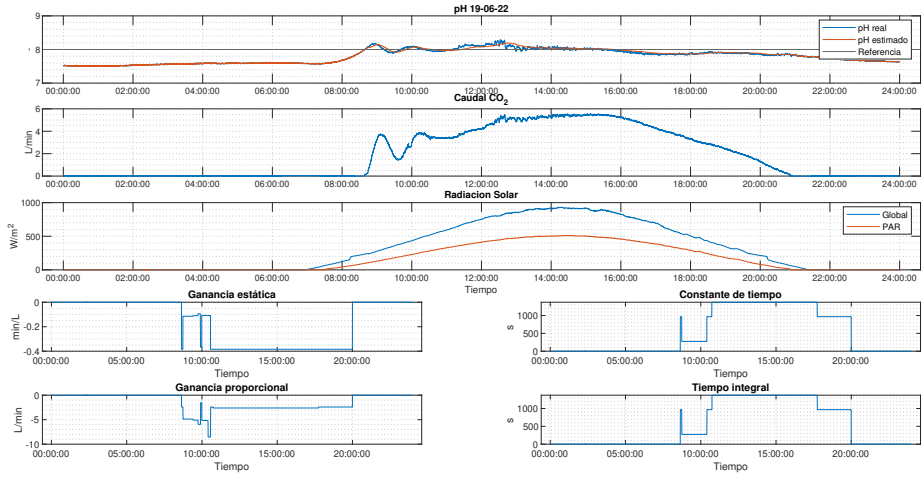


Figura 4.32: Control adaptativo en reactor del Convenio UAL-IFAPA, 19 de junio

Día	IAE	ISE	ITAE	ITSE	EC
16 de junio	4.1140 e+03	1.2282 e+03	5.4677 e+07	7.9564 e+06	522.6917
18 de junio	4.1221 e+03	558.9979	9.0100 e+07	1.1964 e+07	510.5508
19 de junio	3.3471 e+03	389.7199	7.5293 e+07	8.8794 e+06	554.9814

Tabla 4.7: Índices de comportamiento de rechazo de perturbaciones - Controlador PI de parámetros adaptativos en reactor del Convenio UAL-IFAPA

Día	RMSE
16 de junio	0.1085
18 de junio	0.0618
19 de junio	0.0451

Tabla 4.8: Error cuadrático medio entre pH estimado y pH real para control adaptativo-Reactor del Convenio UAL-IFAPA

Si se comparan los índices obtenidos para el control adaptativo con los obtenidos para el control de parámetros fijos (tablas 4.3 y 4.4), se puede observar que los índices de comportamiento en lo que respecta al control son similares en ambos casos, siendo generalmente mejores los obtenidos para el controlador fijo. En cuanto a los valores del *RMSE* obtenidos para los días en que se ha implementado el controlador adaptativo, estos son considerablemente mejores que los obtenidos para el controlador fijo.

Sin embargo, la comparación de estos índices solo sería completamente correcta si ambos controladores se hubieran implementado en las mismas condiciones de radiación, temperatura y nivel.

5

Conclusiones y trabajos futuros

En el presente capítulo se exponen las conclusiones extraídas en el desarrollo de este TFG y trabajos futuros que se pueden realizar en las instalaciones del Convenio UAL-IFAPA.

5.1. Conclusiones

Los resultados obtenidos durante la realización de este Trabajo Fin de Grado dejan de manifiesto el correcto funcionamiento del estimador de pH durante la inyección de CO_2 y del controlador PI de parámetros adaptativos. Se ha comprobado tanto en simulación como en su implementación real que el controlador diseñado es capaz de obtener un comportamiento adecuado del pH, manteniéndolo siempre entorno a la referencia establecida y con menores oscilaciones que al implementar un controlador PI clásico de parámetros fijos. Presente la enorme ventaja de que será capaz de brindar un buen funcionamiento independientemente de las condiciones a las que se encuentre sometido el sistema, en lo que respecta a radiación global, temperatura del medio y nivel del cultivo. Además, se ha comprobado que el estimador de pH brinda resultados muy satisfactorios cunado se encuentra implementado junto con el control adaptativo.

Por lo tanto, se puede concluir que la técnica utilizada para la estimación de los parámetros utilizando árboles de regresión funciona adecuadamente para este tipo de sistemas. Es necesario contar con una gran cantidad de datos para que todas las condiciones que se puedan presentar estén contempladas en los nodos de decisión de dicho árbol. Es por eso que, para un mejor comportamiento del controlador, sería conveniente realizar una campaña de ensayos adicional en épocas de verano, donde se cuenta con radiación y temperatura elevadas, y en épocas de invierno, donde la radiación y la temperatura decrecen.

Por otro lado, se ha encontrado una limitación en lo que respecta al modelado de la respuesta libre, cuando el sistema no se encuentra sometido a inyecciones de CO_2. A pesar del esfuerzo que se ha realizado en obtener un modelo que se ajuste a los datos y que sea capaz de realizar las estimaciones, no se ha llegado a resultados satisfactorios. Esto se debe, probablemente, a la simplicidad que presentaba la estructura de los modelos escogidos y a la falta de un mayor volumen de datos en el entrenamiento de los árboles..

Sin embargo, la respuesta libre representa el aumento de pH debido a la producción de la fotosíntesis en los intervalos donde el caudal de CO_2 inyectado es nulo. Por lo tanto, el modelo que se busca obtener es únicamente con fines de modelado, pues no se puede utilizar para realizar un control. Es por eso que debería continuarse realizando el esfuerzo de obtener modelos para dicha dinámica, sin importar cuánto se tenga que aumentar la complejidad del mismo.

5.2. Trabajos futuros

En base a las conclusiones obtenidas anteriormente, se pueden plantear los siguientes trabajos futuros a realizar para continuar mejorando la idea propuesta en el presenta TFG:

- Repetir los ensayos realizados, de modo que se pueda obtener una mayor cantidad de datos para entrenar los árboles.
- Implementar el controlador adaptativo en diferentes épocas del año para comprobar su correcto funcionamiento en condiciones diversas.
- Obtener un modelo para conseguir estimar la respuesta libre del sistema, en ausencia de inyección de CO_2.

Referencias

[1] Barceló-Villalobos, M., Gómez Serrano, C., Sánchez Zurano, A., Alameda García, L., Esteve Maldonado, S., Peña, J., and Acién Fernández, F. G. (2019). Variations of culture parameters in a pilot-scale thin-layer reactor and their influence on the performance of scenedesmus almeriensis culture. *Bioresource Technology Reports*, 6:190–197. https://doi.org/10.1016/j.biteb.2019.03.007.

[2] Berenguel, M., Rodríguez, F., Acién, F. G., and García, J. (2004). Model predictive control of pH in tubular photobioreactors. *Journal of Process Control*, 14:377–387. https://doi.org/10.1016/j.jprocont.2003.07.001.

[3] Caparroz, M., Otálora, P., Berenguel, M., and Guzmán, J. L. (2022). Modelado y control adaptativo del pH en reactores raceway para la producción de microalgas. *XLIII Jornadas de Automática, Logroño, España*.

[4] Ejea-Carbonell, D. G. (2017). *Árboles de Regresión. Algunos algoritmos y extensiones a métodos de consenso.* Trabajo Fin de Grado en Matemáticas. Universidad de Zaragoza.

[5] Fernández, I., Acién, F. G., Berenguel, M., and Guzmán, J. L. (2014a). First principles model of a tubular photobioreactor for microalgal production. *Industrial & Engineering Chemistry Research*, 53:11121–11136. https://doi.org/10.1021/ie501438r.

[6] Fernández, I., Acién, F. G., Berenguel, M., Guzmán, J. L., Andrade, G., and Pagano, D. J. (2014b). A lumped parameter chemical-physical model for tu-bular photobioreactors. *Chemical Engineering Science*, 112:116–129. https://doi.org/10.1016/j.ces.2014.03.020.

[7] Fernández, I., Acién, F. G., Guzmán, J. L., Berenguel, M., and Mendoza, J. L. (2016). Dynamic model of an industrial raceway reactor for microalgae production. *Algal Research*, 17:67–78. https://doi.org/10.1016/j.algal.2016.04.021.

[8] Fernández, I., Guzmán, J. L., Acién, F. G., and Berenguel, M. (2017). Dynamic modeling of microalgal production in photobioreactors. In *Prospects and Challenges in Algal*

Biotechnology, chapter 7, page 49–87. Springer. https://doi.org/10.1007/978-981-10-1950-0_2.

[9] Gonzalez, L. (2019). Árboles de decisión regresión - teoría. "`https://aprendeia.com/arboles-de-decision-regresion-teoria-machine-learning/`". Última consulta: 01/06/2022.

[10] Guzmán, J. L., Acién, F. G., and Berenguel, M. (2021). Modelling and control of microalgae production in industrial photobioreactors. *Revista Iberoamericana de Automática e Informática Industrial*, 18:1–18. https://doi.org/10.4995/riai.2020.13604.

[11] Hoyo, A., Rodríguez Miranda, E., Guzmán, J. L., G., A. F., Berenguel, M., and Moreno, J. C. (2022). A computer-based tool to simulate raceway photobioreactors for design, operation and control purposes. *Computers and Chemical Engineering*, 156:107472. https://doi.org/10.1016/j.compchemeng.2021.107572.

[12] Ljung, L. (2004). *System Identification Toolbox User's Guide*. The Mathworks Inc.

[13] Otálora Berenguel, P., Guzmán, J., Berenguel, M., and Acién, F. (2020). *Dynamic Model for the pH in a Raceway Reactor Using Deep Learning Techniques*, pages 190–199. DOI: 10.1007/978-3-030-58653-9_18.

[14] Schneider Electric (2019). *Guía de programación. EcoStruxure Machine Expert*. Schneider Electric.

[15] The MathWorks, Inc. (2022a). Fitrtree. "`https://es.mathworks.com/help/stats/fitrtree.html`". Última consulta: 01/06/2022.

[16] The MathWorks, Inc. (2022b). Predict. "`https://es.mathworks.com/help/stats/compactregressiontree.predict.html`". Última consulta: 01/06/2022.

[17] Åström, K. J. and Hägglund, T. (2006). *Advanced PID control*. Research Triangle Park, NC : ISA-The Instrumentation, Systems, and Automation Society.

Apéndices A

Anexos

A.1. Código para la obtención de modelos

A.1.1. Tratamiento de datos previo - Respuesta forzada

```
clear all; close all; clc; warning off;

 % Cambiar para obtener los datos del dia que se desee.
day = 30;
month = 04;
year = 22;
dia = day;
mes = month;

day = num2str(day,'%02.f');
month = num2str(month,'%02.f');
year = num2str(year,'%02.f');
actual = pwd;

 % Archivo de registro de los reactores.
fecha = strcat('C:\Users\Usuario\Desktop\UAL\Beca_Colaboracion_TFG\TFG\
    Ensayos\Resgistros_dias\',month,'\Registro_Reactores_',day,'_',month,'_'
    ,year,'.csv');
comma2point_overwrite(fecha)
opts = detectImportOptions(fecha);
opts.SelectedVariableNames = {'Hora','CaudalCO2RW6','pHRW62b','RadGlobal','
    RadPAR','TempRW6','TempAmbiente','NivelRW6'};
datos = readtable(fecha,opts);

 % Obtengo las variables de interes.
Tiempo = datos.Hora;
```

```
CaudalCO2RW2 = datos.CaudalCO2RW6;
pHRW23 = datos.pHRW62b;
RadGlobal = datos.RadGlobal;
TempRW2 = datos.TempRW6;
TempAmbiente = datos.TempAmbiente;
NivelRW2 = datos.NivelRW6;

% Filloutliers.
pHRW23 = filloutliers(pHRW23,'spline','movmean',800);
CaudalCO2RW2 = filloutliers(CaudalCO2RW2,'spline','movmean',350);

% Plots RW2.
fh1 = figure(1);
fh1.WindowState = 'maximized';
subplot(3,1,1)
plot(datos.Hora,pHRW23)
title('Ensayo pH RW2 '+string(day)+'-'+string(month)+'-'+string(year))
ylabel('-')
legend('pH23')
grid minor

subplot(3,1,2)
plot(datos.Hora,CaudalCO2RW2);
title('Caudal CO2 RW2')
ylabel('L/min');
grid minor

subplot(3,1,3)
plot(datos.Hora,[datos.RadGlobal, datos.RadPAR])
title('Radiacion Solar')
legend('Global','PAR')
ylabel('W/m^2')
xlabel('Tiempo')
grid minor

%% Obtencion de intervalos de respuesta forzada
% 0 = todo/nada
% 1 = Ensayo
Inyectando = double(CaudalCO2RW2>1);
index=find(Inyectando == 1);
idx=find(diff(index)~=1);
A=[idx(1);diff(idx);numel(index)-idx(end)];

% Construyo intervalos de tiempo, caudal de CO2, pH y radiacion global.
Intervalos_Tiempo_1 = mat2cell(Tiempo(Inyectando == 1),A,1);
Intervalos_CaudalCO2RW2_1 = mat2cell(CaudalCO2RW2(Inyectando == 1),A,1);
Intervalos_pHRW23_1 = mat2cell(pHRW23(Inyectando == 1),A,1);
```

```
71 Intervalos_RadGlobal = mat2cell(RadGlobal(Inyectando == 1),A,1);
72 Intervalos_TempRW2 = mat2cell(TempRW2(Inyectando == 1),A,1);
73 Intervalos_NivelRW2 = mat2cell(NivelRW2(Inyectando == 1),A,1);
74
75 % Elimino intervalos muy cortos (por fallos).
76 eliminar = [];
77 for i = 1 : length(Intervalos_CaudalCO2RW2_1)
78    if length(Intervalos_CaudalCO2RW2_1{i})<50
79       eliminar = [eliminar;i];
80    end
81 end
82
83 Intervalos_CaudalCO2RW2_1(eliminar) = [];
84 Intervalos_Tiempo_1(eliminar) = [];
85 Intervalos_pHRW23_1(eliminar) = [];
86 Intervalos_RadGlobal(eliminar) = [];
87 Intervalos_TempRW2(eliminar) = [];
88 Intervalos_NivelRW2(eliminar) = [];
89
90 % Tomo los datos para el System Identification a partir del retardo.
91 retardo = 270; % Promedio obtenido de obtencion de modelos con retardo
       variable.
92
93 % Definicion de vectores.
94 Intervalos_pHRW23 = {length(Intervalos_pHRW23_1)};
95 Intervalos_CaudalCO2RW2 = {length(Intervalos_pHRW23_1)};
96 Intervalos_Tiempo = {length(Intervalos_pHRW23_1)};
97 Intervalos_Radiacion = {length(Intervalos_pHRW23_1)};
98 Intervalos_TemperaturaRW2 = {length(Intervalos_pHRW23_1)};
99 Intervalos_NivelMedioRW2 = {length(Intervalos_pHRW23_1)};
100 % Definicion de vectores.
101
102 for i = 1:length(Intervalos_pHRW23_1)
103    Intervalos_pHRW23{i,1} = Intervalos_pHRW23_1{i}(retardo:1:end);
104    Intervalos_CaudalCO2RW2{i,1} = Intervalos_CaudalCO2RW2_1{i}(retardo:1
           :end);
105    Intervalos_Tiempo{i,1} = Intervalos_Tiempo_1{i}(retardo:1:end);
106    Intervalos_Radiacion{i,1} = Intervalos_RadGlobal{i}(retardo:1:end);
107    Intervalos_TemperaturaRW2{i,1} = Intervalos_TempRW2{i}(retardo:1:end);
108    Intervalos_NivelMedioRW2{i,1} = Intervalos_NivelRW2{i}(retardo:1:end);
109 end
110
111 % Busco el intervalo de ensayo mas largo.
112 mayor = 0;
113 for i = 1 : length(Intervalos_CaudalCO2RW2)
114    if length(Intervalos_CaudalCO2RW2{i})> mayor
115       mayor = length(Intervalos_CaudalCO2RW2{i});
```

```
116     end
117 end
118 
119 TramosCO2 = zeros(mayor,length(Intervalos_CaudalCO2RW2));
120 
121 % Definicion de vectores.
122 MediaRadiacion = zeros(length(Intervalos_CaudalCO2RW2),1);
123 MedianaRadiacion = zeros(length(Intervalos_CaudalCO2RW2),1);
124 ModaRadiacion = zeros(length(Intervalos_CaudalCO2RW2),1);
125 DesviacionRadiacion = zeros(length(Intervalos_CaudalCO2RW2),1);
126 MediaTempRW2 = zeros(length(Intervalos_CaudalCO2RW2),1);
127 MediaNivelRW2 = zeros(length(Intervalos_CaudalCO2RW2),1);
128 pHForzadoInicial = zeros(length(Intervalos_CaudalCO2RW2),1);
129 % Definicion de vectores.
130 
131 %% Contruccion de datos de respuesta forzada para obtencion de funciones de
        transferencia.
132 for i = 1: length(Intervalos_CaudalCO2RW2)
133     Intervalos_CaudalCO2RW2{i}(1,1) = 0;
134     Intervalos_pHRW23{i} = Intervalos_pHRW23{i}-Intervalos_pHRW23{i}(1);
135     TramosCO2(1:length(Intervalos_CaudalCO2RW2{i}),i) = cell2mat(
            Intervalos_CaudalCO2RW2(i,1));
136     TramosCO2(end,i) = 0;
137     final =find(TramosCO2(:,1)==0,2);
138     caudal(i) = mean(TramosCO2(2:final(2),i));
139     TramospH(1:length(Intervalos_pHRW23{i}),i) = cell2mat(Intervalos_pHRW23(
            i,1));
140     TramosRadiacion(1:length(Intervalos_Radiacion{i}),i) = cell2mat(
            Intervalos_Radiacion(i,1));
141     TramosTempRW2(1:length(Intervalos_TemperaturaRW2{i}),i) = cell2mat(
            Intervalos_TemperaturaRW2(i,1));
142     TramosNivelRW2(1:length(Intervalos_NivelMedioRW2{i}),i) = cell2mat(
            Intervalos_NivelMedioRW2(i,1));
143     MediaRadiacion(i) = mean(Intervalos_RadGlobal{i});
144     MedianaRadiacion(i) = median(Intervalos_RadGlobal{i});
145     ModaRadiacion(i) = mode(Intervalos_RadGlobal{i});
146     DesviacionRadiacion(i) = std(Intervalos_RadGlobal{i});
147     MediaTempRW2(i) = mean(Intervalos_TempRW2{i});
148     MediaNivelRW2 (i) = mean(Intervalos_NivelRW2{i});
149     pHForzadoInicial (i) = Intervalos_pHRW23{i}(1);
150 end
151 
152 TramosRadiacion = [TramosRadiacion;zeros(1,length(TramosRadiacion(1,:)))];
153 TramosTempRW2 = [TramosTempRW2;zeros(1,length(TramosTempRW2(1,:)))];
154 TramosNivelRW2 = [TramosNivelRW2;zeros(1,length(TramosNivelRW2(1,:)))];
```

A.1.2. Obtención de modelos - Respuesta forzada

```
%% Obtencion del modelo de repsuesta forzada con System Identification
Ts = 1; % Tiempo de muestreo de 1 segundo.

% Definicion de vectores.
dataset = {length(TramosCO2(1,:))};
modeloset = {length(TramosCO2(1,:))};
modelos = {length(TramosCO2(1,:))};
kp = zeros(length(TramosCO2(1,:)),1);
tau = zeros(length(TramosCO2(1,:)),1);
% Definicion de vectores.

for i = 1: length(TramosCO2(1,:))
   final = find(TramosCO2(:,i)==0,2);
   if size(final) == 1
      data = iddata(TramospH(1:size(TramosCO2(:,i)),i),TramosCO2(:,i),Ts);
   else
      data = iddata(TramospH(1:final(2)-1,i),TramosCO2(1:final(2)-1,i),Ts);
   end
   dataset{i} = data;       % DATOS DEL DIA - RESPUESTA FORZADA.
   modelo = procest(data,'P1'); % SIN RETARDO.
   modeloset{i} = modelo;   % MODELOS DEL DIA - RESPUESTA FORZADA.
   Datos_Modelo = getpvec(modelo);
   kp(i) = Datos_Modelo(1);
   tau(i) = Datos_Modelo(2);
   modelos{i} = tf(kp(i),[tau(i) 1],'InputDelay',retardo*Ts); % Sistemas
       con el retardo calculado.
end

% Grafica respuesta forzada vs modelo.
fh2 = figure(2);
fh2.WindowState = 'maximized';
for i = 1:length(Intervalos_pHRW23_1)
   subplot(round(length(Intervalos_pHRW23_1)/2),2,i)
   plot((1:1:length(Intervalos_pHRW23_1{i})),Intervalos_pHRW23_1{i}-
       Intervalos_pHRW23_1{i}(retardo),'m')
   hold on
   opt = stepDataOptions('StepAmplitude',caudal(i));
   step(modelos{i},opt,length(Intervalos_pHRW23_1{i}))
   legend('Datos','Modelo')
   title('Intervalo '+string(i)+', '+string(day)+'-'+string(month)+'-'+
       string(year))
end

```

```
% Guardar datos en archivo .xls.
dia = ones(length(Intervalos_pHRW23_1),1)*str2double(day);
mes = ones(length(Intervalos_pHRW23_1),1)*str2double(month);

for i = 1:length(caudal)
   caudal1(i,1) = caudal(i);
end

Tabla_Forzada = table(dia,mes,caudal',kp,tau,MediaRadiacion,MediaTempRW2,
    MediaNivelRW2,'VariableNames',{'Dia','Mes','Caudal','Ganancia','
    CteTiempo',...
      'RadiacionMedia','TemperaturaMedia','NivelMedio'});
writetable(Tabla_Forzada,strcat(pwd,'\Respuestas_Forzadas_Retardo_Fijo.xls'
    ),'WriteMode','append')
```

A.1.3. Tratamiento de datos previo - Respuesta libre

```
%% Obtencion de los intervalos de respuesta libre.

% Definicion de vectores.
index_inicio = zeros(length(Intervalos_Tiempo)-1);
index_final = zeros(length(Intervalos_Tiempo)-1);
largo = zeros(length(Intervalos_Tiempo)-1);
pH_medio = zeros(length(Intervalos_Tiempo)-1);
MediaRadiacionLibre = zeros(length(Intervalos_Tiempo)-1,1);
DesviacionRadiacionLibre = zeros(length(Intervalos_Tiempo)-1,1);
MediaTemperaturaLibre = zeros(length(Intervalos_Tiempo)-1,1);
pHLibreInicial = zeros(length(Intervalos_Tiempo)-1,1);
% Definicion de vectores.

for i = 1 : (length(Intervalos_Tiempo)-1)
   Inicio(i) = Intervalos_Tiempo{i}(end,1);
   index_inicio(i) = find(Tiempo==Inicio(i));
   Final(i) = Intervalos_Tiempo{i+1}(1,1);
   index_final(i) = find(Tiempo==Final(i));
   largo(i) = index_final(i)-index_inicio(i);
   pH_libre(1:largo(i)+1,i) = pHRW23(index_inicio(i):index_final(i));
   pH_medio(i) = mean(pHRW23(index_inicio(i):index_final(i)));
   Radiacion_libre(1:largo(i)+1,i) = RadGlobal(index_inicio(i):index_final(
       i));
   Temperatura_libre(1:largo(i)+1,i) = TempRW2(index_inicio(i):index_final(
       i));
   Nivel_libre(1:largo(i)+1,i) = NivelRW2(index_inicio(i):index_final(i));
```

```
25     MediaRadiacionLibre(i) = mean(RadGlobal(index_inicio(i):index_final(i)))
           ;
26     MediaTemperaturaLibre(i) = mean(TempRW2(index_inicio(i):index_final(i)))
           ;
27     MediaNivelLibre(i) = mean(NivelRW2(index_inicio(i):index_final(i)));
28     DesviacionRadiacionLibre(i) = std(RadGlobal(index_inicio(i):index_final(
           i)));
29     pHLibreInicial(i) = pHRW23(index_inicio(i));
30 end
31
32 Radiacion_libre = [Radiacion_libre;zeros(1,length(Radiacion_libre(1,:)))];
33 Temperatura_libre = [Temperatura_libre;zeros(1,length(Radiacion_libre(1,:))
       )];
34 Nivel_libre = [Nivel_libre;zeros(1,length(Radiacion_libre(1,:)))];
```

A.1.4. Obtención de modelos por mínimos cuadrados - Respuesta libre

```
1  %% Modelo por minimos cuadrados
2  % Saco el modelo de la respuesta libre por estimacion por minimos cuadrados
3  % x(k) = theta1 * x(k-1) + theta2
4
5  n = 1;  % primer orden
6  largolargo = 0;
7
8  for i = 2:size(pH_libre,2)
9     if largo(i)>largo(i-1)
10       largolargo = largo(i);
11    end
12 end
13
14 % Definicion de vectores
15 Y = zeros(largolargo-1,size(pH_libre,2));
16 M = zeros(largolargo-1,size(pH_libre,2));
17 M1 = zeros(largolargo-1,size(pH_libre,2));
18 constantes2 = zeros(size(pH_libre,2),2);
19 ye = zeros(1,size(pH_libre,2));
20 % Definicion de vectores
21
22 for i = 1:size(pH_libre,2)
23    Y(1:largo(i)-1,i) = pH_libre(n+1:largo(i),i);
24    M(1:largo(i)-1,i) = pH_libre(n:largo(i)-n,i);
25    M1(1:largo(i)-1,i) = ones;
26    theta = [M(retardo:largo(i)-1,i) M1(retardo:largo(i)-1,i)]\Y(retardo:
          largo(i)-1,i);
```

```
    constantes2(i,:) = theta';
    ye(1:n,i) = pH_libre(retardo,i);
end

% Verifico si se hizo bien la identificacion
for i = 1:size(pH_libre,2)
    for k = n+1:largo(i)-retardo
        ye(k,i) = constantes2(i,1)*ye(k-1,i)+ constantes2(i,2);
    end
end

pH_libre = [pH_libre;zeros(1,length(pH_libre(1,:)))];

fh4 = figure(4);
fh4.WindowState = 'maximized';
for i = 1:size(pH_libre,2)
    tiempo1 = 1:1:length(pH_libre(1:find(pH_libre(:,i)==0,1)-2,i));
    tiempo2 = 1+retardo:1:largo(i);
    subplot(round(size(pH_libre,2)/2),2,i)
    plot(tiempo1,(pH_libre(1:find(pH_libre(:,i)==0,1)-2,i)));
    hold on
    plot(tiempo2,ye(1:largo(i)-retardo,i));
    title('Respuesta libre '+string(i)+', '+string(day)+'-'+string(month)+'-
        '+string(year))
end


% Grafica de radiacion en intervalos de respuesta libre
fh5 = figure(5);
fh5.WindowState = 'maximized';
for i = 1:size(Radiacion_libre,2)
    subplot(round(size(Radiacion_libre,2)/2),2,i)
    plot(Radiacion_libre(1:find(Radiacion_libre(:,i)==0,1)-1,i))
    title('Radiacion libre '+string(i)+', '+string(day)+'-'+string(month)+'-
        '+string(year))
end


 Tabla_Libre = table(dia(1:end-1),mes(1:end-1),constantes2(:,1),constantes2
     (:,2),MediaRadiacionLibre,MediaTemperaturaLibre,MediaNivelLibre','
     VariableNames',{'Dia','Mes','Theta1','Theta2',...
    'RadiacionMedia','TemperaturaMedia','NivelMedio'});
  writetable(Tabla_Libre,strcat(pwd,'\Respuestas_Libres_Retardo_Fijo.xls'),'
      WriteMode','append')

save(strcat(pwd,'\Datos_Validacion\Datos_',day,'_',month,'_',year),'
    TramosCO2','TramosRadiacion','TramosTempRW2','TramospH','TramosNivelRW2'
```

```
        ...
68    ,'pH_libre','Radiacion_libre','Temperatura_libre','Nivel_libre','
          pHForzadoInicial','pHLibreInicial');
```

A.1.5. Obtención de modelos mediante *System Identification* - Respuesta libre

```
1  nx = 2;   % Orden del modelo
2  Ts = 10;  % Tiempo de muestreo
3
4  for i = 1:length(Intervalos_L_pH(1,:))
5     final = find(Intervalos_L_pH(:,i)>0,1,'last');
6     data = iddata(Intervalos_L_pH(tr:final,i),[],Ts);
7     data.Tstart = 0;
8     dataset{i} = data;
9     %%% Opcion 1: PEM %%%
10    modelo = pem(dataset{i},nx,'Ts',0);
11
12    %%% Opcion 2: N4SID %%%
13    % opt = ssestOptions('InitializeMethod','n4sid','Focus','prediction','
          EnforceStability',false)
14    % modelo = ssest(data,nx,opt)
15
16    modeloset{i} = modelo;
17    figure(i+1)
18    compare(dataset{i},modeloset{i})
19 end
```

A.2. Código para implementación de estimador a tramos

A.2.1. Predictor de la respuesta forzada

```
%% Validacion de los arboles de regresion - Respuesta forzada %%

clear all, clc, warning off, close all

% ELEGIR EL DIA.
day = 15;
month = 11;
year = 21;

day = num2str(day,'%02.f');
month = num2str(month,'%02.f');
year = num2str(year,'%02.f');

% Modelos de arboles de regresion obtenidos.
load('ModelosRegresion_Forzada_sinCO2.mat');

% Cargar datos.
load(strcat(pwd,'\Datos_Validacion\Datos_',day,'_',month,'_',year,'.mat'));
TramosCO2 = [TramosCO2;zeros(1,size(TramosCO2,2))];

%% ESTIMACION DE LA RESPUESTA FORZADA

for i = 1:size(TramosCO2,2)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % ESTIMACION INSTANTE A INSTANTE %
    % Vectores de radiacion, temperatura y nivel en cada instante
    Radiacion = TramosRadiacion(1:find(TramosRadiacion(:,i)==0,1)-1,i);
    Temperatura = TramosTempRW2(1:find(TramosTempRW2(:,i)==0,1)-1,i);
    Nivel = TramosNivelRW2(1:find(TramosNivelRW2(:,i)==0,1)-1,i);

    % Construyo los vectores necesarios para estimar k y tau en funcion de
        valores instantaneos de Rad, Temp y Niv.
    testX = [Radiacion,Temperatura,Nivel];
    testX = rmmissing(testX);
    pruebaInput.time = (0:size(testX,1)-1)';
    pruebaInput.signals(1).values = testX;
    pruebaInput.signals(1).dimensions = size(testX,2);

    vector_tiempo = 0:10:pruebaInput.time(end)*10;

    % Estimo k y tau para cada instante
```

```
k_estimada = predict(TreeModelk_sinCO2,[Radiacion Temperatura Nivel]);
tau_estimada = predict(TreeModeltau_sinCO2,[Radiacion Temperatura Nivel]);

% Vectores de tau y k estimadas y entrada instante a instante para validacion
vector_tau = [vector_tiempo',tau_estimada];
vector_k = [vector_tiempo',k_estimada];
cero = find(TramosCO2(:,i)==0,2);
vector_entrada =[vector_tiempo',TramosCO2(1:cero(2)-1,i)];

% Simulo la validacion
sim('C:\Users\Usuario\Desktop\UAL\Beca_Colaboracion_TFG\TFG\Ensayos\simulink_validacion.slx',vector_tiempo(end))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ESTIMACION SOLO CON EL DATO INICIAL %
tau_estimada1 = predict(TreeModeltau_sinCO2,[Radiacion(1) Temperatura(1) Nivel(1)]);
k_estimada1 = predict(TreeModelk_sinCO2,[Radiacion(1) Temperatura(1) Nivel(1)]);

vector_tiempo1 = 0:10:(cero(2)-2)*10;
vector_entrada =[vector_tiempo1',TramosCO2(1:cero(2)-1,i)];

% Le meto esa entrada a un modelo con la k y la tau estimadas
% anteriormente
G = tf(k_estimada1,[tau_estimada1 1]);
entrada = TramosCO2(1:cero(2)-1,i);
tfinal = vector_tiempo1(end);

[pH,t] = lsim(G,entrada,vector_tiempo1');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Graficas
set(gcf, 'Position', get(0, 'Screensize'));
figure(i)

subplot(3,2,[1:2])
plot(vector_tiempo',TramospH(1:cero(2)-1,i),'LineWidth',1.5);
hold on
plot(pH_estimado,'LineWidth',1.5)
plot(vector_tiempo1',pH,'xm')
title('pH real y modelos estimados')
xlabel('Muestras')
```

```
83      legend('Real','Estimado (siempre)','Estimado(principio)')
84
85      subplot(3,2,3)
86      plot(vector_tiempo',tau_estimada)
87      yline(tau_estimada1)
88      title('Tau')
89      xlabel('Muestras')
90      ylabel('s')
91
92      subplot(3,2,4)
93      plot(vector_tiempo',k_estimada)
94      yline(k_estimada1)
95      ylim([-1 0])
96      title('K')
97      xlabel('Muestras')
98      ylabel('min/L')
99
100     subplot(3,2,5)
101     plot(vector_tiempo',TramosRadiacion(1:find(TramosRadiacion(:,i)==0,1)-1,
            i));
102     title('Radiacion')
103     xlabel('Muestras')
104     ylabel('W/m^2')
105
106     subplot(3,2,6)
107     plot(vector_tiempo',TramosTempRW2(1:find(TramosTempRW2(:,i)==0,1)-1,i));
108     title('Temperatura')
109     xlabel('Muestras')
110     ylabel('C')
111 end
```

A.2.2. Predictor de la respuesta libre

```
1  %% Validacion de los arboles de regresion - Respuesta libre %%
2
3  clear all, clc, warning off, close all
4
5  % ELEGIR EL DIA.
6  day = 15;
7  month = 11;
8  year = 21;
9
10 day = num2str(day,'%02.f');
11 month = num2str(month,'%02.f');
```

```
12 year = num2str(year,'%02.f');
13
14 % Modelos de arboles de regresion obtenidos.
15 load('Modelos_Regresion_Libre_14Marzo.mat')
16
17 % Cargar datos.
18 load(strcat(pwd,'\Datos_Validacion\Datos_',day,'_',month,'_',year,'.mat'));
19
20
21 %% ESTIMACION DE LA RESPUESTA LIBRE
22
23 for i = 1:size(pH_libre,2)
24    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25    % ESTIMACION DE PARAMETROS USANDO UNICAMENTE EL VALOR INICIAL %
26    Radiacion = Radiacion_libre(1,i);
27    Temperatura = Temperatura_libre(1,i);
28    Nivel = Nivel_libre(1,i);
29
30    theta1_estimada1(i) = predict(ModeloTheta1,[Radiacion,Temperatura,Nivel
          ]);
31    theta2_estimada1(i) = predict(ModeloTheta2,[Radiacion,Temperatura,Nivel
          ]);
32    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
33
34    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
35    % ESTIMACION DE PARAMETROS INSTANTE A INSTANTE %
36    for j = 1:find(pH_libre(:,i)==0,1)
37       theta1_estimada2(j,i) = predict(ModeloTheta1,[Radiacion_libre(j),
             Temperatura_libre(j),Nivel_libre(j)]);
38       theta2_estimada2(j,i) = predict(ModeloTheta2,[Radiacion_libre(j),
             Temperatura_libre(j),Nivel_libre(j)]);
39    end
40    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41
42    % Estimacion de ambos pH.
43    pH_estimado_libre1(1,i) = pH_libre(1,i);
44    pH_estimado2(1,i)= pH_libre(1,i);
45
46    for j = 2:find(pH_libre(:,i)==0,1)
47       pH_estimado_libre1(j,i) = theta1_estimada1(i)*pH_estimado_libre1(j-1,
             i) + theta2_estimada1(i);
48       pH_estimado2(j,i) = theta1_estimada2(j-1,i)*pH_estimado2(j-1,i) +
             theta2_estimada2(j-1,i);
49    end
50
51    % Graficas
52    tiempo = 0:10:find(pH_libre(:,i)==0,1)*10-1;
```

```
53      figure(i+size(TramosCO2,2))
54
55      subplot(3,2,[1:2])
56      plot(tiempo(1:end-1)',pH_libre(1:find(pH_libre(:,i)==0,1)-1,i),'
            LineWidth',1.5)
57      hold on
58      plot(tiempo',pH_estimado2(1:find(pH_libre(:,i)==0,1),i),'LineWidth',1.5)
59      plot(tiempo',pH_estimado_libre1(1:find(pH_libre(:,i)==0,1),i),'LineWidth
            ',1.5)
60      xlim([0 find(pH_libre(:,i)==0,1)*10-20])
61      grid on
62      legend('datos','estimado (cada instante)','estimado (inst inicial)','
            Location','northwest')
63      title('pH real y modelos estimados')
64
65      subplot(3,2,3)
66      plot(tiempo',Radiacion_libre(1:find(pH_libre(:,i)==0,1),1),'LineWidth'
            ,1.5)
67      title('Radiacion')
68      xlim([0 find(pH_libre(:,i)==0,1)*10-25])
69
70      subplot(3,2,4)
71      plot(tiempo',Temperatura_libre(1:find(pH_libre(:,i)==0,1),i),'LineWidth'
            ,1.5)
72      title('Temperatura')
73      xlim([0 find(pH_libre(:,i)==0,1)*10-25])
74
75      subplot(3,2,5)
76      plot(tiempo',theta1_estimada2(1:find(pH_libre(:,i)==0,1),i),'LineWidth'
            ,1.5)
77      title('Theta 1')
78      xlim([0 find(pH_libre(:,i)==0,1)*10-25])
79
80      subplot(3,2,6)
81      plot(tiempo',theta2_estimada2(1:find(pH_libre(:,i)==0,1),i),'LineWidth'
            ,1.5)
82      title('Theta 2')
83      xlim([0 find(pH_libre(:,i)==0,1)*10-25])
84  end
```

A.3. Código para la implementación de estimador en registros de datos

A.3.1. Estimación de la respuesta para controladores todo/nada

```
%% Implementacion del estimador a partir de registro de datos %%
clear all; close all; clc; warning off;

% ELEGIR EL DIA.
day = 05;
month = 11;
year = 21;

day = num2str(day,'%02.f');
month = num2str(month,'%02.f');
year = num2str(year,'%02.f');
actual = pwd;

% Cargar datos.
fecha = strcat('C:\Users\Usuario\Desktop\UAL\Beca_Colaboracion_TFG\TFG\
    Ensayos\Resgistros_dias\Registro_CALRESI_',day,'_',month,'_',year,'.csv'
    );
comma2point_overwrite(fecha)
opts = detectImportOptions(fecha);
opts.SelectedVariableNames = {'Hora','CaudalCO2RW1','CaudalCO2RW2','pHRW13'
    ,'pHRW23','RadGlobal','RadPAR','TempRW1','TempRW2','TempAmbiente','
    NivelRW1','NivelRW2'};
datos = readtable(fecha,opts);

% Cargar arboles de regresion.
load('Modelos_Regresion_Libre_14Marzo.mat');
load('ModelosRegresion_Forzada.mat');

% Obtengo las variables de interes.
Tiempo = datos.Hora;
CaudalCO2RW1 = datos.CaudalCO2RW1;
CaudalCO2RW2 = datos.CaudalCO2RW2;
pHRW13 = datos.pHRW13;
pHRW23 = datos.pHRW23;
RadGlobal = datos.RadGlobal;
TempRW1 = datos.TempRW1;
TempRW2 = datos.TempRW2;
TempAmbiente = datos.TempAmbiente;
NivelRW1 = datos.NivelRW1;
```

```
36 NivelRW2 = datos.NivelRW2;
37
38 % Creo los vectores necesarios.
39 pHEstimado = zeros(1,size(datos,1));
40 tau = zeros(1,size(datos,1));
41 k = zeros(1,size(datos,1));
42 theta1 = zeros(1,size(datos,1));
43 theta2 = zeros(1,size(datos,1));
44
45 % Parametros del sistema.
46 tr = 27; % Muestras de retardo
47 Tm = 10; % Tiempo de muestreo de los datos
48
49 caudalCO2 = CaudalCO2RW2';
50
51 %% ESTIMACION
52 for i = 1 :size(datos,1)
53    if (datos.Hora(i) < '08:00:00') || (datos.Hora(i) > '21:00:00')
54       pHEstimado(1,i) = datos.pHRW23(i);
55    % Empiezo a estimar a las 8 am hasta las 9 pm.
56    else
57       % Si no hay inyeccion de CO2 estimo la respuesta libre.
58       if caudalCO2(1,i-tr) < 0.5
59          % El primer instante que no hay inyeccion toma el dato del sensor.
60          if caudalCO2(1,i-tr-1) - caudalCO2(1,i-tr) > 0.01
61             pHEstimado(1,i) = datos.pHRW23(i);
62             j = 1;
63          % A partir de aqui comienza a estimar.
64          else
65             theta1(i) = predict(ModeloTheta1,[datos.RadGlobal(i),datos.
                   TempRW2(i),datos.NivelRW2(i)]);
66             theta2(i) = predict(ModeloTheta2,[datos.RadGlobal(i),datos.
                   TempRW2(i),datos.NivelRW2(i)]);
67             pHEstimado(1,i) = theta1(i)*pHEstimado(1,i-1) + theta2(i);
68          end
69       % Si hay inyeccion de CO2
70       else
71          % En el primer instante de inyeccion calculo los parametros y tomo
               como pH estimado anterior el dato leido por el sensor en el
              instante anterior, para no arrastrar errores de estimacion de
              una dinamica a otra.
72          if caudalCO2(1,i-tr) - caudalCO2(1,i-tr-1) > 0.5
73             k(i) = predict(TreeModelk,[datos.RadGlobal(i),datos.TempRW2(i),
                   datos.NivelRW2(i),caudalCO2(i)]);
74             tau(i) = predict(TreeModeltau,[datos.RadGlobal(i),datos.TempRW2
                   (i),datos.NivelRW2(i),caudalCO2(i)]);
75             Gc = tf(k(i),[tau(i) 1],'InputDelay',tr*Tm);
```

```
76              Gz = c2d(Gc,Tm);
77              a = Gz.Numerator{:}(2);
78              b = Gz.Denominator{:}(2);
79              pHEstimado(1,i) = datos.pHRW23(i);
80              pHEstimado(1,i-1) = datos.pHRW23(i-1);
81              pto_op = datos.pHRW23(i-1);
82          else
83              k(i) = k(i-1);
84              tau(i) = tau(i-1);
85          end
86          pHEstimado(1,i) = a*caudalCO2(1,i-tr-1) - b*(pHEstimado(1,i-1)-
                pto_op) + pto_op;
87      end
88   end
89 end
90
91 %% GRAFICAS
92 subplot(4,2,1:2)
93 plot(datos.Hora,pHEstimado,'LineWidth',1.2)
94 grid on
95 hold on
96 plot(datos.Hora,datos.pHRW23,'LineWidth',1.2)
97 title('pH')
98 legend('pH estimado','pH real')
99 ylim([5 22])
100 ylabel('-')
101 xlabel('Tiempo (hs)')
102
103 subplot(4,2,3:4)
104 plot(datos.Hora,datos.CaudalCO2RW2,'LineWidth',1.2)
105 title ('Caudal CO_2')
106 xlabel('Tiempo (hs)')
107 ylabel('L/min')
108 grid on
109
110 subplot(4,2,5)
111 plot(datos.Hora,theta1)
112 title('Theta 1')
113 xlabel('Tiempo (hs)')
114 grid on
115
116 subplot(4,2,6)
117 plot(datos.Hora,theta2)
118 title('Theta2')
119 xlabel('Tiempo (hs)')
120 grid on
121
```

```
subplot(4,2,7)
plot(datos.Hora,k)
title('Ganancia')
ylabel ('min/L')
xlabel('Tiempo (hs)')
grid on

subplot(4,2,8)
plot(datos.Hora,tau)
title('Constante de tiempo')
ylabel('s')
xlabel('Tiempo (hs)')
grid on
```

A.3.2. Estimación de la respuesta forzada con controlador PI

```
%% Estimador de pH en algoritmos de control PI/PID %%
clear all; close all;
clc; warning off;

% Cambiar para obtener los datos del dia que se desee.
day = 10;
month = 04;
year = 22;

day = num2str(day,'%02.f');
month = num2str(month,'%02.f');
year = num2str(year,'%02.f');
actual = pwd;

% Cargar datos
fecha = strcat('C:\Users\Usuario\Desktop\UAL\Beca_Colaboracion_TFG\TFG\
    Ensayos\Resgistros_dias\',month,'\Registro_Reactores_',day,'_',month,'_'
    ,year,'.csv');
comma2point_overwrite(fecha)
opts = detectImportOptions(fecha);
opts.SelectedVariableNames = {'Hora','CaudalCO2RW5','CaudalCO2RW6','pHRW52b
    ','pHRW62b','RadGlobal','RadPAR','TempRW5','TempRW6','TempAmbiente','
    NivelRW5','NivelRW6'};
datos = readtable(fecha,opts);
% Cargar arboles de respuesta forzada.
load('ModelosRegresion_Forzada_sinCO2.mat');

% Obtengo las variables de interes
```

```
Tiempo = datos.Hora;
CaudalCO2RW1 = datos.CaudalCO2RW5;
CaudalCO2RW2 = datos.CaudalCO2RW6;
CaudalCO2RW2 = filloutliers(CaudalCO2RW2,"nearest","movmean",1000);
pHRW13 = datos.pHRW52b;
pHRW23 = datos.pHRW62b;
pHRW23 = filloutliers(pHRW23,"nearest","movmean",1000);
RadGlobal = datos.RadGlobal;
TempRW1 = datos.TempRW5;
TempRW2 = datos.TempRW6;
TempAmbiente = datos.TempAmbiente;
NivelRW1 = datos.NivelRW5;
NivelRW2 = datos.NivelRW6;

% Variables y vectores auxiliares
pHEstimado = zeros(1,size(datos,1));
estimando = zeros(1,size(datos,1));
tau = zeros(1,size(datos,1));
k = zeros(1,size(datos,1));
theta1 = zeros(1,size(datos,1));
theta2 = zeros(1,size(datos,1));
tr = 270;                   % Tiempo de retardo
Tm = 1;                      % Tiempo de muestreo de los datos
caudalCO2 = CaudalCO2RW2';
j=1;
refresh = 300;

for i = 1 :size(datos,1)
   if (datos.Hora(i) < '06:00:00') || (datos.Hora(i) > '23:00:00')
      pHEstimado(1,i) = datos.pHRW62b(i);
   % Empiezo a estimar a las 6 am hasta las 23 pm
   else
       % NO hay inyeccion de CO2 -> Leo directamente el dato
       if caudalCO2(1,i-tr) <= 0.5
            pHEstimado(1,i) = datos.pHRW62b(i);
            estimando(1,i) = 0;
       % Si hay inyeccion de CO2
       else
          if caudalCO2(1,i-tr) > 0.5
             if j == 1
                % Realizo la estimacion de k y tau cada 'refresh'
                % muestras
                caudal(i) = caudalCO2(1,i-tr) - caudalCO2(1,i-tr-1);
                k(i:i+refresh) = predict(TreeModelk_sinCO2,[datos.RadGlobal(
                    i),datos.TempRW6(i),datos.NivelRW6(i)]);
                tau(i:i+refresh) = predict(TreeModeltau_sinCO2,[datos.
                    RadGlobal(i),datos.TempRW6(i),datos.NivelRW6(i)]);
```

```
70                  pto_op = pHEstimado(1,i-2);
71                  pto_op = datos.pHRW62b(i-2);
72                  Gc = tf(k(i),[tau(i) 1],'InputDelay',tr*Tm);
73                  Gz = c2d(Gc,Tm);
74                  a = Gz.Numerator{:}(2);
75                  b = Gz.Denominator{:}(2);
76                  estimando(1,i) = 1;
77                  pHEstimado(1,i) = a*(caudalCO2(1,i-tr)-caudalCO2(1,i-tr-1))
                        - b*(pHEstimado(1,i-1)-pto_op) + pto_op;
78              end
79              if j<=refresh
80                  pHEstimado(1,i) = a*(caudalCO2(1,i-tr)-caudalCO2(1,i-tr-1))
                        - b*(pHEstimado(1,i-1)-pto_op) + pto_op;
81                  j = j+1;
82              end
83              if j == refresh
84                  j = 1;
85              end
86          end
87      end
88  end
89  end
90
91  %% Graficas
92  subplot(3,2,1:2)
93  plot(datos.Hora,pHEstimado,'LineWidth',1.2)
94  grid on
95  hold on
96  plot(datos.Hora,datos.pHRW62b,'LineWidth',1.2)
97  title('pH')
98  legend('pH estimado','pH real')
99  subplot(3,2,3:4)
100 plot(datos.Hora,datos.CaudalCO2RW6,'LineWidth',1.2)
101 xlabel('Tiempo (hs)')
102 title('Caudal CO2')
103 ylabel('L/min')
104 grid on
105 subplot(3,2,5)
106 plot(datos.Hora,k,'LineWidth',1.2)
107 title('Ganancia')
108 xlabel('Tiempo (hs)')
109 ylabel('min/L')
110 grid on
111 subplot(3,2,6)
112 plot(datos.Hora,tau,'LineWidth',1.2)
113 title('Constante de tiempo')
114 xlabel('Tiempo (hs)')
```

```
115 ylabel('s')
116 grid on
```

A.4. Código para la implementación de estimador en simulador

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Estimador de pH en simulador con control PI de parametros fijos %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; clc;

%% 1. Carga de datos
load('tramos_3.mat');    % Cargar el dataset completo (muestreo 1 min)
load('red_bc1.mat');     % Cargar el modelo
numdia = 6;              % Seleccionar el dia con el que se trabajara
data = Total_tramos{1,numdia}; % Datos del dia
T = Total_tramos{2,numdia}; % Vector de tiempos para representacion
retardo = net.numInputDelays; % Numero de delays del modelo

load('ModelosRegresion_Forzada_sinCO2.mat'); % Cargar los arboles de
    regresion

%% 2. Parametros del controlador
tr = floor(270/60);        % Tiempo de retardo
Tm = 60;                   % Tiempo de muestreo
pHref = 7.8*ones(length(data),1); % Referencia de pH

% Inicializacion de auxiliares
j = 1;
flag = 0;
suma = 0;
alfa = 1;
refresco = 5;

%% 3. Simulacion
pHk = [data(1:retardo,1)]; % Vector para guardar los valores de salida de
    pH
CO2k = [data(1:retardo,4)]; % Vector para guardar los valores de entrada,
    CO2
Buffer = [];
t_com = zeros(length(data),1);
pHEstimado = zeros(length(data),1);
k = zeros(length(data),1);
tau = zeros(length(data),1);
Kp = zeros(length(data),1);
Ti = zeros(length(data),1);
pHEstimado(1:retardo) = pHk;

for i = 1+retardo:length(data)-retardo
```

```
41
42      % Actualizo buffer de regresores
43      Buffer.Nivel = data(i-retardo:i-1,5);  % Nivel del reactor, columna 5.
44      Buffer.Temp = data(i-retardo:i-1,6);   % Temperatura del reactor, columna
            6.
45      Buffer.TA = data(i-retardo:i-1,7);     % Temperatura ambiente, columna 7.
46      Buffer.Rad = data(i-retardo:i-1,8);    % Radiacion global, columna 8.
47      Buffer.pH = pHk(i-retardo:i-1);        % Buffer para pH
48      Buffer.CO2 = CO2k(i-retardo:i-1);      % Buffer para CO2
49
50      % Lecturas de los sensores, nuevos datos del instante k
51      Nivel_new = data(i,5);      % Nivel del reactor, columna 5.
52      Temp_new = data(i,6);       % Temperatura del reactor, columna 6.
53      TA_new = data(i,7);         % Temperatura ambiente, columna 7.
54      Rad_new = data(i,8);        % Radiacion global, columna 8.
55
56      k(i) = -0.167732093;        % Ganancia media de todos los modelos.
57      tau(i) = 524.2016532;       % Constante de tiempo media de todos los
            modelos.
58      if data(i,8)>0
59         % Estimacion de pH
60         if CO2k(i-tr) > 0
61            if j == 1
62               caudal(i) = CO2k(i-tr) - CO2k(i-tr-1);
63               k1(i) = predict(TreeModelk_sinCO2,[Buffer.Rad(end),Buffer.Temp
                     (end),Buffer.Nivel(end)]);
64               tau1(i) = predict(TreeModeltau_sinCO2,[Buffer.Rad(end),Buffer.
                     Temp(end),Buffer.Nivel(end)]);
65               pto_op = pHk(i-1);%(i-2);
66               Gc = tf(k(i),[tau(i) 1],'InputDelay',tr*60);
67               Gz = c2d(Gc,Tm);
68               a = Gz.Numerator{:}(2);
69               b = Gz.Denominator{:}(2);
70               pHEstimado(i) = a*(CO2k(i-tr)-CO2k(i-tr-1)) - b*(pHEstimado(i
                     -1)-pto_op) + pto_op;
71               estimando(i) = 1;
72               alfa = 1;
73            end
74            if j>1 && j <= refresco
75               k1(i) = k(i-1);
76               tau1(i) = tau(i-1);
77               pHEstimado(i) = a*(CO2k(i-tr)-CO2k(i-tr-1)) - b*(pHEstimado(i
                     -1)-pto_op) + pto_op;
78               estimando(i) = 1;
79               alfa = alfa - 1/refresco;
80            end
81            if j == refresco
```

```
82              j = 0;
83              alfa = 1;
84          end
85          j = j+1;
86       else
87          pHEstimado(i) = pHk(end);
88          estimando (i) = 0;
89       end
90    else
91       pHEstimado(i) = pHk(end-1);
92       estimando (i) = 0;
93    end
94
95    if T(i) > '20:00:00'
96       flag = 0;
97    elseif pHk(end) > pHref(1) && flag == 0
98       flag = 1;
99    end
100
101   if flag == 1
102      k(i) = -0.167732093;        % Ganancia media de todos los modelos.
103      tau(i) = 524.2016532;
104      Ti(i) = tau(i-j)*alfa + tau(i)*(1-alfa);
105      Tt = sqrt(Ti(i));
106      Kp(i) = tau(i-j)*alfa/(k(i-j)*(0.8*tau(i-j)+tr*Tm)) + tau(i)*(1-alfa)
             /(k(i)*(0.8*tau(i)+tr*Tm));
107      e(i) = pHref(i) - pHk(end);
108      suma = suma + Kp(i)/Ti(i) * Tm * e(i);
109      CO2(i) = Kp(i)*e(i) + suma;
110   elseif flag == 0
111      CO2(i) = 0;
112   end
113
114   % Saturacion
115   if CO2(i)<0
116      CO2_new = 0;
117   elseif CO2(i)>13
118      CO2_new = 13;
119   else
120      CO2_new = CO2(i);
121   end
122
123   if flag == 1
124      % Antiwindup
125      es = CO2_new - CO2(i);
126      suma = suma + es/Tt;
127   end
```

```
128     % Concatenacion de regresores y nuevos valores -- Llamada al modelo/
            Planta real
129     pHk_new = modelo_pH(Buffer.pH,[Buffer.CO2; CO2_new],...
130         [Buffer.Nivel; Nivel_new],[Buffer.Temp; Temp_new],[Buffer.TA; TA_new
                ],...
131         [Buffer.Rad; Rad_new]); % La funcion devuelve como salida tantos
                instantes de pH como filas de cada variable menos 5. Ver funcion
                para mas detalles.
132
133     pHk = [pHk; pHk_new];  % Actualizo vector de medidas de pH
134     CO2k = [CO2k; CO2_new]; % Actualizo vector de medidas de CO2
135
136
137 end
138
139 %% 4. Representacion grafica.
140 figure
141 subplot(4,2,1:2)
142 plot(T(:,1),data(:,8),'LineWidth',1.2);
143 hold on
144 title('Radiacion');
145 ylabel('W/m^2'); grid;
146
147 subplot(4,2,3:4)
148 plot(T(1:end-retardo,1),pHk,'LineWidth',1.2);
149 hold on
150 plot(T,pHref,'r--');grid
151 plot(T(1:end-retardo,1),pHEstimado(1:end-retardo,1),'r',LineWidth',1.2)
152 legend('Prediccion','Referencia','Estimado');
153 title('pH');
154
155 subplot(4,2,5:6)
156 plot(T(1:end-retardo,1),CO2k,'LineWidth',1.2);
157 title('Caudal CO_2');
158 ylabel('L/min');
159 xlabel('Tiempo (hs)')
160 grid;
161
162 grid;
163 subplot(4,2,7)
164 plot(T(1:end-retardo,1),k1(1:end-retardo,1),'LineWidth',1.2);
165 title('Ganancia')
166 ylabel('min/L');
167 xlabel('Tiempo (hs)'); grid;
168
169 subplot(4,2,8)
170 plot(T(1:end-retardo,1),tau1(1:end-retardo,1),'LineWidth',1.2);
```

```
171 title('Constante de tiempo')
172 ylabel('s');
173 xlabel('Tiempo (hs)'); grid;
```

A.5. Código para la implementación de estimador en el reactor

```
function [Refs,Control,Aux1,Aux2,Aux3] = Control_Malena_Fijo(
    datos_reactores,pHRW6)

persistent SPCaudalCO2RW6 SPCaudalAireRW6 suma_RW6 alfa es flag j CaudalCO2
     pto_op pHEstimado k tau a b;
 % Cargar arboles de regresion
load('ModelosRegresion_Forzada_sinCO2_menosnodos.mat')

 %%% Definicion e inicializacion de variables necesarias %%%
 % Predictores
Rad = datos_reactores(15).Value;  % Radiacion global.
Temp = datos_reactores(19).Value;  % Temperatura RW6.
Niv = datos_reactores(29).Value;  % Nivel RW6.

 % Setpoints RW6
SP_pHRW6 = datos_reactores(35).Value;  % SetPoint de pH en el RW6.
SP_ODRW6 = datos_reactores(37).Value;  % SetPoint de OD en el RW6.

CaudalAire = 60;                      % Caudal de Aire.
CaudalCO2_new = datos_reactores(26).Value;  % Caudal de CO2 inyectado en
    instante actual.

 % Tiempos del sistema
Tm = 30;              % Tiempo de muestreo - 1 minuto.
Tr = floor(270/Tm);   % Tiempo de retardo.
refresco = 10;

 % Alfa (para transferencia sin saltos)
if isempty(alfa)
   alfa = 1;
end

 % Flag (indicador de control)
if isempty(flag)
   flag = 0;
end

 % J (para estimacion cada x muestras)
if isempty(j)
   j = 1;
end

 % Error de saturacion (para antiwindup).
```

```
41 if isempty(es)
42    es = 0;
43 end
44
45 % Suma (para calcular senal de control).
46 if isempty(suma_RW6)
47    suma_RW6 = 0;
48 end
49
50 % Caudales de CO2
51 if isempty (CaudalCO2)
52    CaudalCO2 = zeros(Tr+5,1);           % Defino vector de ceros para
          almacenar caudal.
53 end
54 CaudalCO2 = [CaudalCO2(2:end); CaudalCO2_new]; % Vector de ventana
       deslizante para caudal.
55
56 % Vector de pHEstimado
57 if isempty(pHEstimado)
58    pHEstimado = zeros(2,1);
59 end
60
61 % Datos necesarios de OD
62 ODRW6_2a = datos_reactores(8).Value;
63
64 %%% ESITIMADOR DE pH DURANTE INYECCION DE CO2 CADA 10 MUESTRAS %%%
65 if CaudalCO2(end-Tr) > 0.1
66    caudal = CaudalCO2(end-Tr) - CaudalCO2(end-Tr-1);
67    if j == 1
68       % Prediccion de parametros del modelo
69       k = predict(TreeModelk_sinCO2,[Rad,Temp,Niv]);
70       tau = predict(TreeModeltau_sinCO2,[Rad,Temp,Niv]);
71       pto_op = pHRW6;
72       Gc = tf(k,[tau 1],'InputDelay',Tr*Tm);
73       Gz = c2d(Gc,Tm);
74       a = Gz.Numerator{:}(2);
75       b = Gz.Denominator{:}(2);
76       pHe = a*caudal - b*(pHEstimado(2)-pto_op) + pto_op;
77       alfa = 1;
78    end
79    if j>1 && j <= refresco
80       pHe = a*(caudal) - b*(pHEstimado(2)-pto_op) + pto_op;
81       alfa = alfa - 1/refresco;
82    end
83    if j == refresco
84       j = 0;
85       alfa = 1;
```

```
    end
    j = j+1;
    pHEstimado = [pHEstimado(2);pHe];
else
    pHEstimado = [pHEstimado(2);pHRW6];
    pHe = pHRW6;
    k = 0;
    tau = 0;
end

%%% CONTROL de pH %%%
% Parametros medios del modelo, para implementar un PI de parametros fijos
k_fija = -0.167732093;    % Ganancia estatica media
tau_fija = 524.2016532;   % Constante de tiempo medio

% Parametros controlador - Metodo Lambda
Ti = tau_fija;                        % Tiempo integral.
Tt = sqrt(Ti);                        % Constante de tracking para antiwindup
    .
Kp = Ti/(k_fija*(0.8*tau_fija+Tr*Tm)); % Ganancia proporcional.

% Condiciones para control de pH
if Rad < 50
    flag = 0;
elseif pHRW6 > SP_pHRW6 && flag == 0
    flag = 1;
end

% Senal de control
if flag == 1
    e = SP_pHRW6 - pHRW6;
    suma_RW6 = suma_RW6 + Kp/Ti * Tm * e;
    CO2 = Kp * e + suma_RW6;
elseif flag == 0
    CO2 = 0;
end

% Saturacion
if CO2 < 0
    u = 0;
elseif CO2 > 13
    u = 13;
else
    u = CO2;
end

% Antiwindup
```

```
132 if flag == 1
133    es = u - CO2;
134    suma_RW6 = suma_RW6 + es/Tt;
135 end
136 
137 % Setpoints caudales
138 SPCaudalCO2RW6 = u;
139 SPCaudalAireRW6 = 0;
140 
141 % Si no se esta inyectando CO2 se puede inyectar aire en caso de estar el
142 % OD por encima de su Setpoint.
143 try
144    if isempty(ODRW6_2a) == True
145       ODRW6_2a = 0;
146    end
147 catch
148 end
149 try
150    if (flag == 0) && (ODRW6_2a > SP_ODRW6)
151       SPCaudalAireRW6 = CaudalAire;
152    elseif (flag == 1) || (ODRW6_2a < SP_ODRW6)
153       SPCaudalAireRW6 = 0; % Esto se cumple tanto si se esta inyectando CO2
           como si el OD esta por debajo del SP.
154    end
155 catch
156 end
157 
158 [Refs] = [SPCaudalCO2RW6,SPCaudalAireRW6];
159 Control = 'Control_Malena_RW6_Fijo';
160 Aux1 = k;
161 Aux2 = tau;
162 if isempty(Aux1)
163   Aux1 = 0;
164 end
165 if isempty(Aux2)
166    Aux2 = 0;
167 end
168 Aux3 = pHe;
```

A.6. Código para la implementación del control adaptativo en el reactor

```
function [Refs,Control,Aux1,Aux2,Aux3,Aux4,Aux5] =
    Control_Malena_Adaptativo(datos_reactores,pHRW6)

persistent SPCaudalCO2RW6 SPCaudalAireRW6 suma_RW6 alfa es flag j CaudalCO2
     pto_op pHEstimado k tau a b Kp Ti tau_old k_old;
% Cargar arboles de regresion
load('ModelosRegresion_Forzada_sinCO2_25nodos.mat')

%%% Definicion e inicializacion de variables necesarias %%%
% Predictores
Rad = datos_reactores(15).Value; % Radiacion global.
Temp = datos_reactores(19).Value; % Temperatura RW6.
Niv = datos_reactores(29).Value; % Nivel RW6

CO2 = 0;

% Setpoints RW6
SP_pHRW6 = datos_reactores(35).Value; % SetPoint de pH en el RW6.
SP_ODRW6 = datos_reactores(37).Value; % SetPoint de OD en el RW6.

CaudalAire = 60;                      % Caudal de Aire.
CaudalCO2_new = datos_reactores(26).Value; % Caudal de CO2 inyectado en
    instante actual.

% Tiempos del sistema
Tm = 30;              % Tiempo de muestreo - 1 minuto.
Tr = floor(270/Tm);   % Tiempo de retardo.
refresco = 10;

% Alfa (para transferencia sin saltos)
if isempty(alfa)
   alfa = 1;
end

% Flag (indicador de control)
if isempty(flag)
   flag = 0;
end

% J (para estimacion cada x muestras)
if isempty(j)
```

```
39      j = 1;
40  end
41
42  % Error de saturacion (para antiwindup).
43  if isempty(es)
44      es = 0;
45  end
46
47  % Suma (para calcular senal de control).
48  if isempty(suma_RW6)
49      suma_RW6 = 0;
50  end
51
52  % Caudales de CO2
53  if isempty (CaudalCO2)
54      CaudalCO2 = zeros(Tr+5,1); % Defino vector de ceros para almacenar
            caudal.
55  end
56  CaudalCO2 = [CaudalCO2(2:end); CaudalCO2_new]; % Vector de ventana
        deslizando para caudal.
57
58
59  % Vector de pHEstimado
60  if isempty(pHEstimado)
61      pHEstimado = zeros(2,1);
62  end
63
64  % Vector de ganancias estaticas
65  if isempty(k_old)
66      k_old = -0.167732093; %*ones(refresco+5,1);
67      k = -0.167732093;
68  end
69
70  % Vector de constantes de tempo
71  if isempty (tau_old)
72      tau_old = 524.2016;
73      tau = 524.2016;
74  end
75
76  % Datos necesarios de OD.
77  ODRW6_2a = datos_reactores(8).Value;
78
79  %%% ESITIMADOR DE pH DURANTE INYECCION DE CO2 %%%
80  if CaudalCO2(end-Tr) > 0.1
81      caudal = CaudalCO2(end-Tr) - CaudalCO2(end-Tr-1);
82      if j == 1
83          % Prediccion de parametros del modelo
```

```
84        k = [k(2:end) ; predict(TreeModelk_sinCO2,[Rad,Temp,Niv])];
85        tau = [tau(2:end) ; predict(TreeModeltau_sinCO2,[Rad,Temp,Niv])];
86        pto_op = pHRW6;
87        Gc = tf(k,[tau 1],'InputDelay',Tr*Tm);
88        Gz = c2d(Gc,Tm);
89        a = Gz.Numerator{:}(2);
90        b = Gz.Denominator{:}(2);
91        pHe = a*caudal - b*(pHEstimado(2)-pto_op) + pto_op;
92        alfa = 1;
93     end
94     if j>1 && j <= refresco
95        k = [k(2:end) ; k(end)];
96        tau = [tau(2:end) ; tau(end)];
97        pHe = a*(caudal) - b*(pHEstimado(2)-pto_op) + pto_op;
98        alfa = alfa - 1/refresco;
99     end
100    if j == refresco
101       j = 0;
102       alfa = 1;
103    end
104    j = j+1;
105    pHEstimado = [pHEstimado(2);pHe];
106 else
107    pHEstimado = [pHEstimado(2);pHRW6];
108    pHe = pHRW6;
109 %    k = 0;
110 %    tau = 0;
111 end
112
113 %%% CONTROL pH %%%
114
115 % Condiciones para control de pH
116 if Rad < 50
117    flag = 0;
118 elseif pHRW6 > SP_pHRW6 && flag == 0
119    flag = 1;
120 end
121
122 % Parametros controlador - Metodo Lambda con transferencia sin saltos.
123    Ti = tau_old*alfa + tau*(1-alfa);
124    Tt = sqrt(Ti);
125    Kp = tau_old*alfa/(k_old*(0.8*tau_old+Tr*Tm)) + tau*(1-alfa)/(k*(0.8*tau
           +Tr*Tm));
126
127 % Senal de control
128 if flag == 1
129    e = SP_pHRW6 - pHRW6;
```

```
130     suma_RW6 = suma_RW6 + Kp/Ti * Tm * e;
131     CO2 = Kp * e + suma_RW6;
132 elseif flag == 0
133     CO2 = 0;
134 end
135 
136 % Saturacion
137 if CO2 < 0
138     u = 0;
139 elseif CO2 > 13
140     u = 13;
141 else
142     u = CO2;
143 end
144 
145 % Antiwindup
146 if flag == 1
147     es = u - CO2;
148     suma_RW6 = suma_RW6 + es/Tt;
149 end
150 
151 % Setpoints caudales
152 SPCaudalCO2RW6 = u;
153 SPCaudalAireRW6 = 0;
154 
155 tau_old = tau;
156 k_old = k;
157 
158 % Si no se esta inyectando CO2 se puede inyectar aire en caso de estar el
159 % OD por encima de su Setpoint.
160 try
161     if isempty(ODRW6_2a) == True
162         ODRW6_2a = 0;
163     end
164 catch
165 end
166 try
167     if (flag == 0) && (ODRW6_2a > SP_ODRW6)
168         SPCaudalAireRW6 = CaudalAire;
169     elseif (flag == 1) || (ODRW6_2a < SP_ODRW6)
170         SPCaudalAireRW6 = 0; % Esto se cumple tanto si se esta inyectando CO2
                como si el OD esta por debajo del SP.
171     end
172 catch
173 end
174 
175 if isnan(SPCaudalCO2RW6)
```

```
176     SPCaudalCO2RW6 = 0;
177 end
178 
179 [Refs] = [SPCaudalCO2RW6,SPCaudalAireRW6];
180 Control = 'Control_Malena_RW6_Adaptativo';
181 Aux1 = k
182 Aux2 = tau
183 if isempty(Aux1)
184   Aux1 = 0;
185 end
186 if isempty(Aux2)
187     Aux2 = 0;
188 end
189 Aux3 = pHe
190 Aux4 = Kp
191 Aux5 = Ti
192 if isempty(Aux4) || isnan(Aux4)
193   Aux4 = 0;
194 end
195 if isempty(Aux5)
196     Aux5 = 0;
197 end
```

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

Las algas son microorganismos con un gran potencial en la producción de biomasa con alto valor añadido y en el tratamiento de aguas residuales. Su capacidad de crecimiento presenta una fuerte dependencia del pH, el oxígeno disuelto (OD), la radiación global y la temperatura del medio. El pH y el OD pueden controlarse mediante la inyección de CO2 y aire respectivamente, mientras que la radiación y la temperatura (en general) son factores que en reactores abiertos no pueden ser controlados.

La tasa de crecimiento de las microalgas viene determinada por un factor principal, que es la disponibilidad de luz y depende en gran medida de la concentración del cultivo y el nivel del medio. Además, este termino viene ponderado por otros tres que representan la temperatura, el pH y el oxigeno disuelto y cuyos valores se encuentran normalizados entre 0 y 1.

La influencia de la temperatura y el pH son muy similares, ambos presentan un valor óptimo entorno al cual la tasa de crecimiento es máxima, pero fuera de ese valor la tasa decrece drásticamente. En cuanto al oxígeno disuelto, hay un valor a partir del cual la tasa de crecimiento desciende. El pH es, entre todas las variables a controlar, la más crítica y complicada, pues la producción de la fotosíntesis provoca variaciones continuas.

Actualmente, es necesario llegar a un balance entre la complejidad del modelo de pH y la dificultad para desarrollar algoritmos de control. Por un lado, modelos muy complejos reflejan con mucha exactitud el comportamiento del sistema tanto espacial como temporalmente, pero dificultan el diseño de controladores. Por el otro, modelos muy simples facilitan el diseño de los algoritmos de control pero su excesiva sencillez provoca un modelado del sistema poco exacto y la pérdida de eficacia de los controladores cuando se presentan condiciones ambientales distintas a las que había a la hora de obtener el modelo, por lo que estos modelos tendrían que ser constantemente calibrados.

Es por las razones expuestas anteriormente que se han desarrollado modelos de árboles de regresión que sean capaces de calcular los parámetros de un modelo simple. Se buscaba conseguir un modelo sencillo pero que, en cierto modo, sea capaz de auto calibrarse en función de las condiciones a las que se encuentre sometido el sistema. Este modelo ha sido utilizado tanto para fines de modelado como para fines de control.

Finalmente, tras la realización de los ensayos se han obtenido resultados satisfactorios que indican un correcto funcionamiento tanto del estimador de pH como del controlador adaptativo.