UNIVERSITAT JAUME I

ESCOLA SUPERIOR DE TECNOLOGIA I CIÈNCIES EXPERIMENTALS

MÀSTER UNIVERSITARI EN ENGINYERIA INDUSTRIAL

DESARROLLO DE UN ALGORITMO DE CONTROL DE UNA ESTACIÓN DEPURADORA DE AGUAS RESIDUALES

TRABAJO DE FINAL DE MÁSTER

AUTOR

RUBÉN MOLINER HEREDIA

DIRECTORES

IGNACIO PEÑARROCHA ALÓS

ROBERTO SANCHIS LLOPIS

CASTELLÓN, OCTUBRE DE 2018

AGRADECIMIENTOS

A mis tutores, Ignacio Peñarrocha Alós y Roberto Sanchis Llopis, por el esfuerzo y dedicación que han aportado para ayudarme a sacar adelante este proyecto.

A mi familia, por todo el apoyo y ayuda que me han dado durante el transcurso del grado y del máster, y en especial, durante la elaboración de este Proyecto.

A mis compañeros, por haberme apoyado cuando lo necesitaba.

Muchas gracias.

ÍNDICE

1.- MEMORIA

1.1.- OBJETO

1.1.1.- OBJETIVO

El objetivo de este proyecto es el desarrollo de un algoritmo de control de una estación depuradora de aguas residuales (EDAR) que permita un mayor ahorro energético y económico que los actuales métodos de control de estas estaciones. Para ello, se van a utilizar técnicas de predicción de señales, así como métodos de control predictivo.

1.1.2.- JUSTIFICACIÓN

El proyecto se justifica por una necesidad de reducción del consumo eléctrico empleado en el control del proceso de tratamiento de aguas residuales, debido a que durante el funcionamiento habitual de una EDAR se consumen cantidades muy elevadas de energía, y el proceso de control podría ser más eficiente energéticamente.

1.2.- ALCANCE

El ámbito de aplicación del presente proyecto se encuentra en el campo del control predictivo y optimización de los recursos utilizados por las estaciones depuradoras. Para implementar dicho control, en este proyecto se han utilizado distintas técnicas procedentes del campo de la investigación del control de sistemas, tales como mecanismos de identificación de sistemas no lineales, de detección de patrones y fallos o de técnicas de optimización.

Para el diseño del sistema de control, se parte del modelo descrito en el Benchmark Simulation Model no. 1 (BSM1), de J. Álex, L. Benedetti et al., así como de los datos de caudal proporcionados en dicho documento.

El alcance del presente proyecto no incluye la implementación del algoritmo en una EDAR real.

1.3.- ANTECEDENTES

Las estaciones depuradoras de aguas residuales tienen un papel fundamental en el desarrollo de las sociedades actuales, así como en la preservación del medio ambiente y la protección de los recursos hídricos. Su función principal es la de procesar las aguas residuales producidas por la actividad humana y eliminar los residuos que contiene, de manera que las aguas resultantes puedan verterse a ríos y mares sin perjuicio de los seres vivos que habitan esos ecosistemas.

Es importante tener en cuenta que las EDAR son construcciones de gran envergadura, con depósitos que almacenan elevadas cantidades de aguas residuales, que necesitan ser procesadas y tratadas. Para lograrlo, las EDAR consumen cantidades ingentes de electricidad, hecho que comporta un elevado coste para la empresa que gestiona dicha EDAR.

Parte del consumo eléctrico de la EDAR se produce en los sopladores de oxígeno situados en el tratamiento secundario de las aguas residuales. El oxígeno se insufla en los reactores químicos

para lograr, entre otras reacciones químicas, la eliminación del amonio afluente a la EDAR, de manera que no se vierta en exceso al medio receptor.

Para controlar que no se superen ciertos límites de vertido de amonio, el sistema de control implementado en muchas EDAR consiste en un simple control de tipo relé (todo/nada) aplicado a los sopladores de oxígeno.

Este método de control dista bastante de ser óptimo, y no tiene en cuenta parámetros económicos relevantes como la tarificación horaria. Tampoco tiene en cuenta el impacto económico que comporta el transporte del fango extraído de los decantadores de este tratamiento, por lo que no se considera la cantidad de fango generado durante el proceso.

En este proyecto, por tanto, se abordará el diseño de un algoritmo de control que permita mantener la salida de componentes amoniacales bajo unos límites establecidos al mismo tiempo que limita el impacto económico causado por la depuración de las aguas residuales.

1.4.- NORMAS Y REFERENCIAS

1.4.1.- DISPOSICIONES LEGALES Y NORMAS APLICADAS

La redacción del presente proyecto se ha realizado en función de la norma UNE 157001-2014, de junio de 2004 "Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico".

1.4.2.- PROGRAMAS DE CÁLCULO

Los programas de cálculo utilizados para el desarrollo del presente proyecto se muestran a continuación:

Matlab. Este entorno de programación está diseñado para realizar operaciones con matrices y sistemas de manera rápida. Permite la escritura de programas y funciones realizadas, operar con transformadas de Laplace y espacios de estados, realizar análisis del espectro de frecuencias o realizar cálculos simbólicos, entre muchas otras complejas operaciones. También permite la descarga e instalación de aplicaciones (*apps*) enfocadas a realizar cálculos específicos orientados a diversas ramas de la ingeniería.

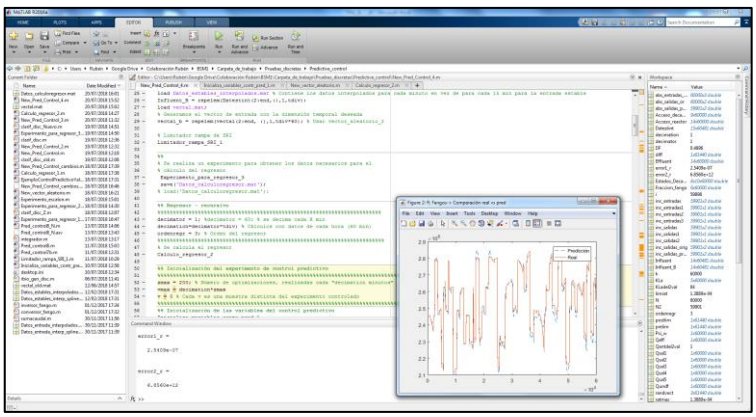

Imagen 1.- Entorno de programación Matlab

YALMIP. Consiste en un paquete de programas y funciones que se descarga para que Matlab pueda ejecutarlas. En el presente proyecto se utiliza YALMIP para la optimización de funciones, necesaria para la resolución del problema del control predictivo.

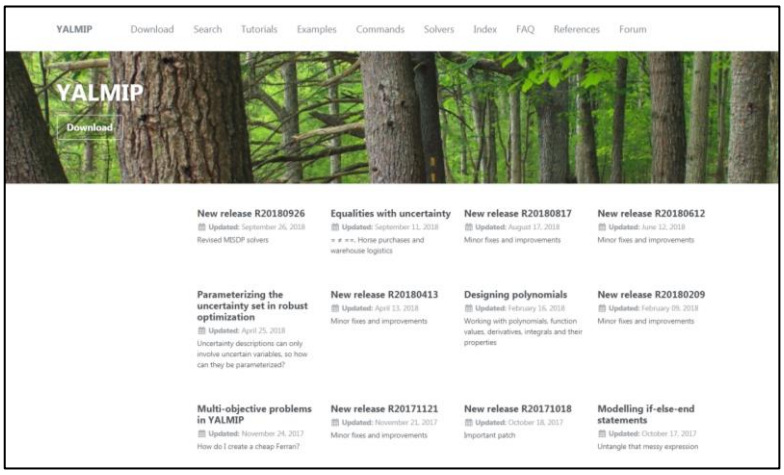

Imagen 2.- Visualización de la página web de YALMIP

Simulink. Se trata de una aplicación importante de Matlab que permite operar con señales y ondas, modificarlas y visualizar su comportamiento en tiempo real. Se caracteriza por funcionar mediante un lenguaje gráfico, utilizando bloques con funciones incorporadas personalizables, en vez de basarse exclusivamente en código.

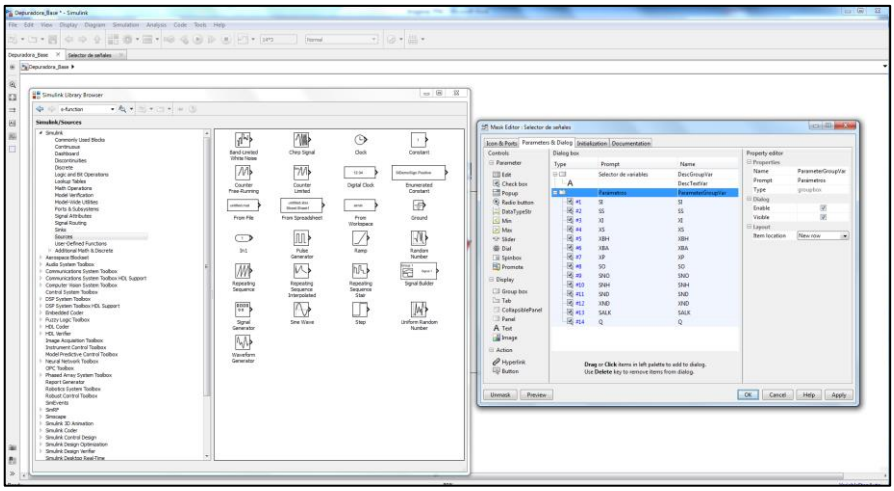

Imagen 3.- Entorno de programación Simulink

WEST. Es una plataforma que permite el modelado dinámico y la simulación del comportamiento de EDARs, así como de otros sistemas relacionados con la calidad del agua. Incluye una gran variedad de modelos matemáticos de los distintos procesos fisicoquímicos que se producen en las estaciones depuradoras, y suele ser un programa utilizado para comprobar cálculos en el ámbito de la ingeniería de fluidos. Esta plataforma dispone de herramientas para el intercambio de datos con Matlab.

Imagen 4.- Imagen de carga de WEST

Freepidtools. Este conjunto de aplicaciones desarrolladas en software libre contiene herramientas que permiten tanto la identificación de funciones de transferencia como de diseño de controladores de tipo PID. Estas aplicaciones han sido creadas por Roberto Sanchís Llopis, profesor de la Universitat Jaume I (Castellón).

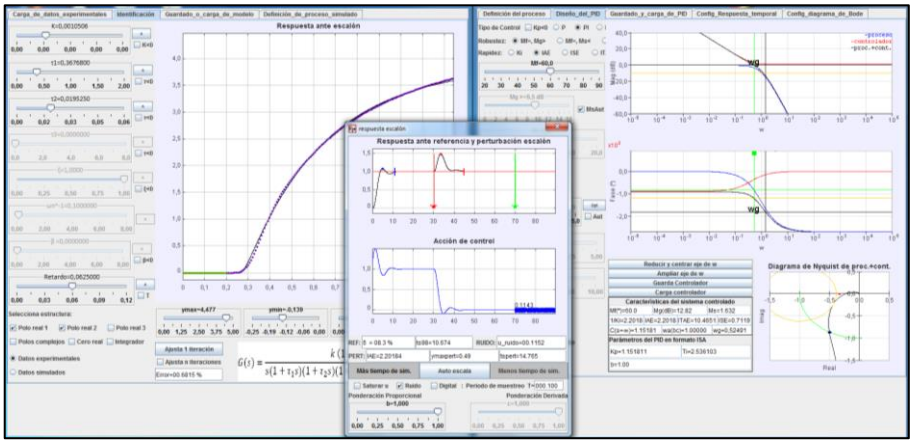

Imagen 5.- Herramientas de Freepidtools

1.4.3.- BIBLIOGRAFÍA

Alex, J., et al. "Benchmark simulation model no. 1 (BSM1)." Report by the IWA Taskgroup on Benchmarking of Control Strategies for WWTPs (2008). (Corrección de 2018)

Gruber, J. K., and Ignacio Peñarrocha Alós. "Soluciones alternativas al control predictivo basado en modelo de Volterra." (2016).

Briones, Juan Ignacio, et al. "Control de amonio en un reactor biológico tipo carrusel utilizando algoritmos matemáticos basados en lógica borrosa (sistema LOBO2)." (2014).

Apuntes de la asignatura ET1023 - Sistemas Automáticos.

Apuntes de la asignatura EE1038 - Regulación Industrial.

Apuntes de la asignatura SJA010 - Automatización y Control Avanzado de Procesos.

Apuntes de la asignatura SJA018 - Ampliación de Instalaciones Eléctricas.

Apuntes de la asignatura SJA002 - Análisis y Diseño de Procesos Químicos.

Apuntes de la asignatura ET1033 - Tecnologías del Medio Ambiente y Seguridad Industrial.

1.4.4.- OTRAS REFERENCIAS

http: //apps.ensic.inpl-nancy.fr/benchmarkWWTP/ - Página web del IWA de donde se ha obtenido tanto el documento del BSM1 como los datos necesarios para los experimentos.

https: //yalmip.github.io/ - Página web desde donde se puede descargar el paquete YALMIP para Matlab.

https: //sites.google.com/a/uji.es/freepidtools/ - Página web de donde se puede descargar el conjunto de aplicaciones Freepidtools.

http://www.omie.es/inicio – Página web de donde se pueden obtener los precios de la energía eléctrica.

1.5.- DEFINICIONES Y ABREVIATURAS

1.5.1.- ESPECÍFICOS DE LA ESTACIÓN DEPURADORA

Afluente. Se denomina afluente al caudal de entrada a la estación depuradora. Estas aguas residuales, que provienen del alcantarillado, se originan en las viviendas y en las industrias. Las aguas pluviales también forman parte del caudal afluente. Antes de entrar en el reactor biológico, al caudal afluente se le ha aplicado una serie de tratamientos físicos para reducir los sólidos presentes.

ASM1. *Activated Sludge Model nº 1*, o Modelo de Lodos Activados número 1, es el conjunto de ecuaciones que modeliza el comportamiento de un reactor biológico de lodos activados, permitiendo cálculos para la predicción de la demanda de oxígeno requerida, la degradación de la biomasa, el crecimiento de los microorganismos y la eliminación de la COD. Está desarrollado por el IWA (Asociación Internacional del Agua). Posteriores versiones, como el ASM2, permiten un mayor refinamiento de las ecuaciones, así como la inclusión de otras ecuaciones que determinan la eliminación de fósforo. El ASM1 utiliza un conjunto específico de variables de estado, correspondiente a 13 concentraciones de distintas sustancias presentes en las aguas residuales. Estas variables se muestran en el Anexo 2.2.2.2.- Reactor biológico.

BSM1. El *Benchmark Simulation Model nº 1*, o modelo de referencia número 1, consiste en unas series de ecuaciones (entre las que se incluye el ASM1), restricciones y condiciones que modelizan el funcionamiento del tratamiento secundario de una EDAR. Incluye también una serie de experimentos que sirven para verificar el correcto funcionamiento de la simulación.

DQO/COD. Demanda Química de Oxígeno o *Chemical Oxygen Demand*, es la medida de la cantidad de oxígeno que va a ser consumida por una serie de reacciones químicas en una disolución. La medida de la DQO de un agua residual sirve para comprobar la cantidad de microorganismos que hay en la misma, y la legislación suele establecer unos límites máximos de DQO para poder verter el agua al medio receptor.

EDAR/WWTP. Siglas de Estación Depuradora de Aguas Residuales, o *Wastewater Treatment Plant*. Una EDAR trata las aguas residuales procedentes de núcleos urbanos, eliminando suficientes componentes nocivos como para que puedan verterse de manera segura de vuelta al medio receptor.

Efluente. Se denomina efluente al caudal de agua que sale por la parte superior del decantador secundario de la estación depuradora. A veces, a este caudal se le aplican tratamientos fisicoquímicos adicionales (conocidos como tratamientos terciarios), aunque también puede ser directamente vertido al medio receptor en función de la calidad del agua.

$\boldsymbol{K_L a}$ – Coeficiente de transferencia líquido-gas. Se refiere a la facilidad de transferencia del oxígeno en el líquido del reactor biológico. El coeficiente consta de dos términos: a, que es proporcional al área de la interfase de transferencia (superficie de la burbuja), y K_L, que abarca otras variables (como la agitación del líquido del reactor, facilitando o dificultando el mezclado del oxígeno).

Medio receptor. Se trata de cualquier lugar de la naturaleza donde pueden ser vertidas las aguas depuradas, como ríos y mares.

RSU. Los Residuos Sólidos Urbanos son aquellos residuos generados por la actividad humana no directamente industrial. En el tratamiento primario de una EDAR se extraen estos residuos que ha arrastrado el agua.

SRI. Selector de Recirculación Interna. En este proyecto, parámetro definido por la válvula que permite la recirculación de las aguas residuales desde el último compartimento del reactor biológico hasta el primero. Cuanto mayor sea este parámetro, más caudal recircula de nuevo hacia la entrada del reactor.

WEST. Siglas de *Worldwide Engine for Simulation, Training and Automation*. Se trata de una plataforma de modelado y simulación de EDAR.

1.5.2.- SOBRE INGENIERÍA DE CONTROL

Actuador. Elemento físico que permite modificar la entrada a un sistema. Se caracterizan por amplificar el efecto de las señales recibidas, de manera que una modificación de una señal enviada por un sistema electrónico se puede traducir en una variación de una magnitud física muy superior. Por tanto, esto significa que requieren una fuente de energía adicional. Los actuadores tienen características de precisión, de dinamicidad y de rangos que pueden modelarse matemáticamente.

Autorregresivo. Tipo de sistema o modelo que se caracteriza porque el estado actual del sistema depende de sus estados anteriores, así como de otros parámetros. De no depender de estos anteriores estados, se denomina no autorregresivo o de media móvil.

Control en bucle cerrado. Un sistema de control en bucle cerrado tiene como objetivo lograr que las salidas del sistema controlado (o proceso a controlar) se mantengan cerca de unos valores determinados (referencias) o entre unos márgenes determinados mediante la variación de las entradas del sistema. Para calcular dichas variaciones, el sistema de control utiliza la información de las salidas recibida por los sensores, para después dar órdenes a los actuadores para que efectúen los cambios en las entradas.

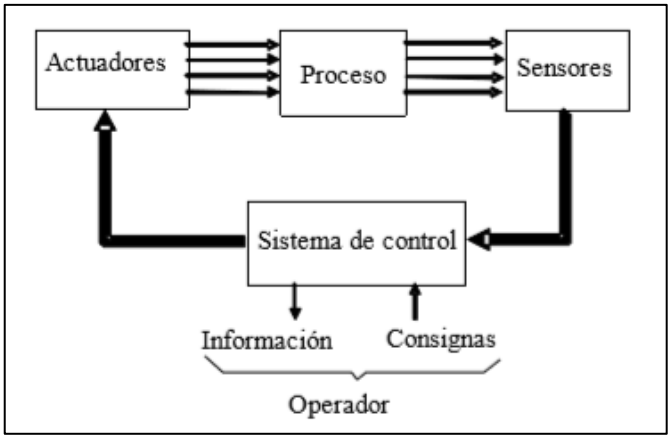

Imagen 6.- Control en bucle cerrado

Control PID. Este método de control se caracteriza por calcular la acción de control sobre el proceso de manera proporcional al error, así como a la integral y a la derivada de dicho error. Los parámetros que deben diseñarse en estos casos se denominan ganancia Proporcional, Integral y Derivativa, y dan nombre a este tipo de control.

Control tipo relé. Este tipo de control también se conoce como on/off o todo/nada. Se trata de un método de control de procesos en el que la acción de control toma el valor más elevado si la salida es inferior a la referencia, y el más reducido si la salida es superior a la referencia. Una variante es el control con **histéresis**, en el que se permiten ciertos márgenes de variación alrededor del punto de referencia para evitar oscilaciones continuas del valor de la acción de control.

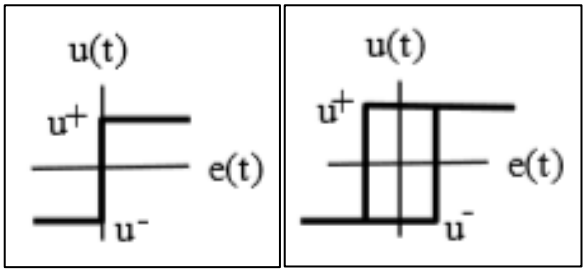

Imagen 7.- Control relé/control relé con histéresis

Discretización. En este Proyecto, se refiere a la conversión de funciones expresadas en tiempo continuo (ecuaciones diferenciales) en funciones expresadas en tiempo discreto (ecuaciones en diferencias).

Entrada. Una entrada es una variable externa a un sistema que se caracteriza porque una variación en el valor de ésta puede afectar al estado de uno o más de los elementos del sistema. Un sistema de control manipula las entradas de un sistema para obtener las salidas deseadas.

Entrada escalón. Se trata de una variación súbita del valor de la entrada de un sistema, con una posterior estabilización de dicha entrada en el nuevo valor. Las entradas escalón suelen utilizarse para caracterizar el comportamiento dinámico y estático de un sistema mediante el uso de determinadas técnicas de evaluación.

Entrada impulsional. Se trata de una variación súbita de la entrada de un sistema, pero a diferencia de la entrada escalón, en el instante de tiempo inmediatamente posterior, el valor vuelve a ser el original. Puede usarse para caracterizar el comportamiento de un sistema mediante técnicas de convolución.

Error. En un sistema de control de un proceso, el error es la diferencia entre el valor de una referencia y el valor de la variable que se está intentando controlar para que siga dicha referencia.

Espacio de estados. Es un modelo matemático descrito como una serie de variables de entrada, de salida e internas (variables de estado), relacionadas mediante ecuaciones diferenciales de primer orden o ecuaciones en diferencias, dependiendo de si están tratando funciones expresadas en forma continua o en discreta, respectivamente.

Fallo. Se entiende como fallo el cambio súbito del comportamiento de una señal. Puede deberse a un cambio en el comportamiento del sistema, o en la aparición de una señal de perturbación.

Filtro. En el ámbito de la ingeniería de control, un filtro es una operación realizada a una señal para eliminar unos ciertos niveles de frecuencia. En función de las bandas de frecuencia eliminadas, los filtros se pueden clasificar en paso bajo, paso alto o paso banda. El filtro paso bajo es comúnmente utilizado para eliminar el ruido de medida, dado que es de alta frecuencia.

Función de transferencia. Es un modelo matemático que caracteriza unívocamente a un sistema, y permite relacionar las variaciones de las variables de entrada con las variaciones de las variables de salida. Esta función se puede obtener tras aplicar transformadas de Laplace a las ecuaciones diferenciales que rigen la dinámica del sistema, o transformadas en Z si son ecuaciones en diferencias.

IIR. Infinite Impulse Response (Respuesta Impulsional Finita). Tipo de filtro que se caracteriza porque un cambio en la entrada (u) del filtro, por pequeño y rápido que sea, tiene un efecto permanente en la salida del filtro (y). Se trata de un modelo autorregresivo con la siguiente forma:

$$y_k = a \cdot y_{k-1} + (1 - a) \cdot u_k \quad \textbf{(1.1)}$$

MATLAB. (*En el resto de este documento,* Matlab, *por facilidad de lectura*) Es un software matemático con un lenguaje de programación propio (lenguaje M) optimizado para realizar complejos cálculos matemáticos mediante el uso de matrices. El nombre procede de MATrix LABoratory.

MPC – Model Predictive Control/ Control Predictivo basado en Modelo. Método de control que utiliza aproximaciones del modelo que controla y perturbaciones futuras para optimizar una función de coste. Ver Anexo 2.3.

Perturbación. En un sistema de control, se trata de una entrada que no se puede controlar. En ocasiones también es imposible su medición. Las perturbaciones son muy comunes en los procesos industriales.

Referencia. En un sistema de control, es el valor al que se desea que llegue una determinada salida. Para ello se aplican variaciones en la entrada del sistema que se quiere controlar.

Ruido. El ruido es una señal de interferencia, generalmente de alta frecuencia, que afecta a las mediciones realizadas mediante un sensor. El ruido es inherente a cualquier medición física, y los métodos de control aplicados deben ser capaces de filtrar dicha señal para que no afecte en exceso a los cálculos y a las entradas (debido a que pueden provocar oscilaciones dañinas para los actuadores). La señal del ruido suele modelizarse con valores aleatorios siguiendo una distribución normal de media cero.

Salida. Es una variable procedente del sistema, cuyo valor depende de los estados del resto de los elementos del sistema y de su variación. El objetivo de un sistema de control básico es lograr que el valor de las salidas se mantenga entre unos márgenes deseados.

Sensor. Elemento físico de medición que transforma un tipo de señal en otra que pueda ser leída por elementos electrónicos (tanto digital como analógica). En un sistema de control, se colocan a las salidas del proceso que se desea controlar para poder informar de su estado. Los sensores tienen características de precisión, de rangos, de dinamicidad y ruido de medida que pueden modelarse matemáticamente.

Señal. Una señal puede definirse como el conjunto ordenado de todos los valores que toma una variable a lo largo del tiempo.

Sistema. Se denomina sistema a todo conjunto de elementos que interactúan entre sí. Debido a dicha interacción, un factor externo que afecte a uno o varios elementos del sistema puede acabar produciendo un cambio en el estado del resto de los elementos del sistema.

YALMIP. Siglas de Yet Another LMI Parser (*Lit. Otro Analizador de LMI*). Las características de este programa se han detallado en el Apartado 1.4.2.- Programas de cálculo.

1.5.3.- OTRAS DEFINICIONES Y ABREVIATURAS

UJI. Siglas de la Universitat Jaume I, localizada en Castellón de la Plana.

ESTCE. Siglas de la Escuela Superior de Tecnologías y Ciencias Experimentales de la Universitat Jaume I.

1.6.- REQUISITOS DE DISEÑO

Para que el algoritmo desarrollado pueda realizar un apropiado control de la EDAR, debe cumplir los siguientes requisitos:

- El método de ejecución del algoritmo debe ser capaz de realizar las predicciones y los cálculos con suficiente precisión. Esto significa que el programa de cálculo empleado debe ser capaz de soportar cálculos realizados con valores que tengan órdenes de magnitud muy diferenciados, así como que permita una discretización de las ecuaciones diferenciales con el menor error posible.

- Los ciclos de cálculo del algoritmo de control deben realizarse con una frecuencia proporcional a las constantes de tiempo de los procesos que se desean controlar en la EDAR. Deben ser inferiores, pero sin ser excesivamente frecuentes, para evitar que se produzcan errores de cálculo por exceso de datos.

- Para conseguir un óptimo funcionamiento del algoritmo del control predictivo, es indispensable que el algoritmo permita una adecuada obtención de modelos no lineales. Esto se debe a que las EDAR son sistemas con un elevado número de variables, y cuyo funcionamiento se basa en ecuaciones diferenciales en las que muchas de estas variables son interdependientes entre sí. Por tanto, el algoritmo deberá utilizar métodos distintos a los tradicionales.

- El algoritmo debe también ser capaz de realizar una previsión del caudal del afluente, dado que es imprescindible saber todas las entradas a la EDAR para que el control predictivo pueda realizar cálculos correctamente.

- Es interesante que el algoritmo pueda detectar la presencia de aguas procedentes de lluvia mezcladas en el afluente sin necesidad de una señal externa que lo indique, de manera que se pueda preservar una estimación correcta del patrón de caudal del afluente.

- El algoritmo debe diseñarse de tal manera que el sistema de control pueda garantizar que la composición de los vertidos (caudal efluente) se puedan mantener bajo los límites que establecen las leyes medioambientales.

- El algoritmo debe poder minimizar la generación de fangos purgados con la EDAR, de manera que el consumo energético (y económico) derivado del transporte y/o tratamiento de los residuos sea lo más bajo posible. Este requisito podría entrar en conflicto con el anterior, por lo que el algoritmo debe ser capaz también de lograr el equilibrio correspondiente.

- El algoritmo debe ser capaz de realizar el control de la EDAR minimizando el consumo eléctrico, teniendo en cuenta los costes económicos que conllevan. Este requisito se traduce en que la prioridad del algoritmo será minimizar el uso de los actuadores que más consumen en función del coste proporcional de cada periodo horario.

1.7.- ANÁLISIS DE SOLUCIONES

1.7.1.- SELECCIÓN DEL TIPO DE CONTROL

A la hora de diseñar un algoritmo de control, los primeros pasos que deben realizarse son la determinación del tipo de control que se va a realizar, así como la selección de las variables que se desean controlar y de las que se puede actuar. Las opciones consideradas para el control del sistema son las siguientes:

- **Control tipo relé**. Este método de control consiste en que el actuador se activa a máxima potencia si la salida que se está controlando supera cierto umbral, y se apaga en cuanto el valor de la salida disminuye y queda por debajo de dicho umbral. Para evitar oscilaciones entorno al umbral, generalmente a este método de control se le añade una histéresis. Este método génerico es utilizado en la actualidad por algunas EDAR para controlar los límites de amonio en el efluente en función de los sopladores de oxígeno ($K_L a$).

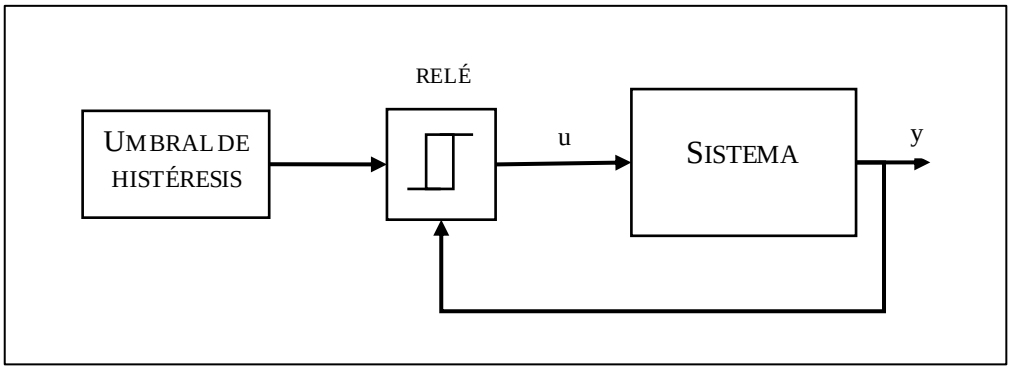

Imagen 8.- Control tipo relé

- **Control mediante el uso de PID**. Este método de control consiste en calibrar y aplicar un controlador PID para actuar sobre una entrada al sistema en función de la diferencia entre el valor de la salida a controlar y el valor de la salida deseada (referencia). A diferencia del caso anterior, los valores de la acción de control pueden encontrarse en un rango continuo de valores, por lo que el control es más preciso que el tipo relé. Sin embargo, a la hora de poner este control en práctica para controlar la salida de amonio en el efluente aparece el inconveniente de la necesidad de instalar un variador de frecuencia en los sopladores de oxígeno.

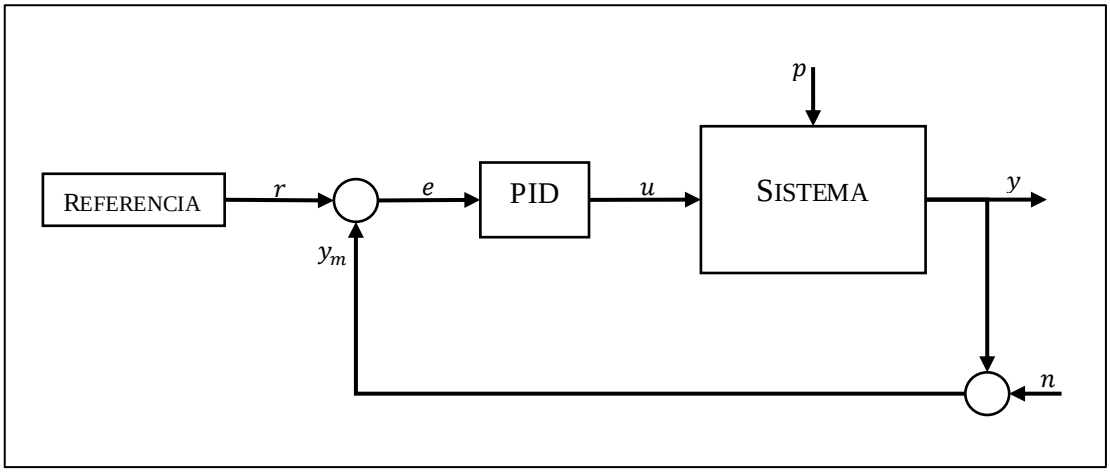

Imagen 9.- Control tipo PID

- **Control predictivo**. Este método consiste en el cálculo de las acciones de control teniendo en cuenta unos objetivos a minimizar y unas restricciones a cumplir. Esta opción se diferencia de los dos anteriores porque la acción de control no se calcula mediante operaciones aplicadas al error entre la referencia y la medida, sino que es el resultado de una optimización.

Este método de control es mucho más complejo que los dos anteriores, y necesita una cantidad de información muy superior. Sin embargo, permite predecir el comportamiento del sistema con suficiente antelación, de manera que puede adelantarse a acontecimientos que los otros métodos no podrían asumir ni controlar, así como minimizar los costes de control causados por los actuadores. Éstos, además, pueden funcionar en un rango de valores o simplemente en modo on-off, beneficiándose el control predictivo de las ventajas de los dos otros tipos de control anteriores.

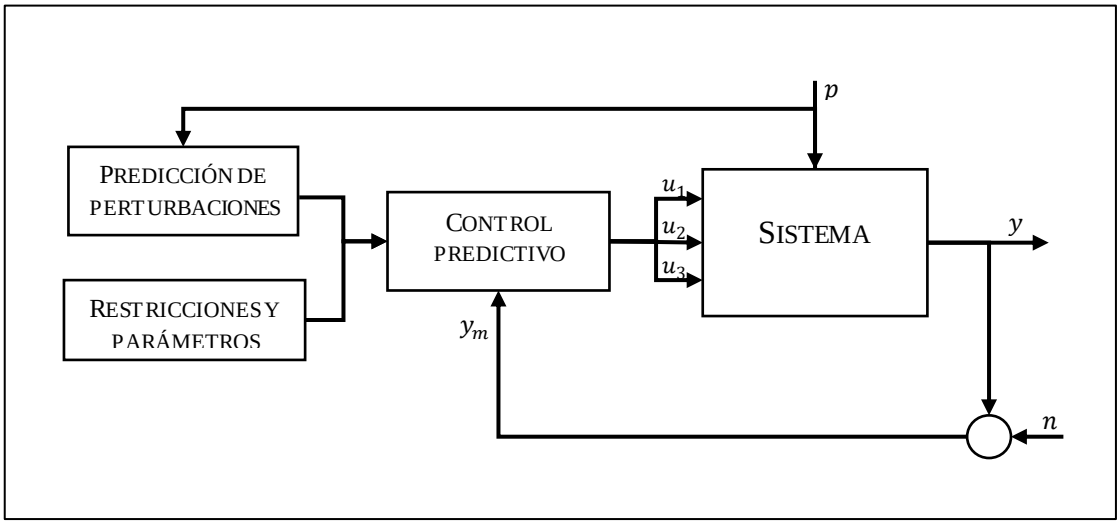

Imagen 10.- Control predictivo

El sistema de control seleccionado es el control predictivo, debido a que cumple con la mayor parte de requisitos, como ahorro energético o adelantamiento al efecto de variaciones en las perturbaciones (cambios en el caudal afluente), hecho que no cumplen los otros dos métodos de control. La herramienta matemática utilizada para diseñar el control predictivo es el optimizador YALMIP.

1.7.2.- SELECCIÓN DE SALIDAS Y ENTRADAS A CONTROLAR

El modelo de la EDAR tiene múltiples salidas a controlar, y un número limitado de entradas controlables. Siguiendo las instrucciones del BSM1, las dos entradas controlables del sistema son los sopladores de oxígeno (en concreto, el parámetro $K_L a$ del último compartimento del reactor biológico) y la válvula de recirculación interna (SRI, en este proyecto).

Las salidas que deben controlarse, según los requisitos para este proyecto, son el valor de la concentración de amonio en el efluente y la cantidad de fango extraído del decantador (denominado Ψ en el proyecto). Las opciones de control son las siguientes:

- **Control monovariable $K_L a$ – Amonio**. Esta opción consiste en controlar el amonio (bien sea para que siga una referencia concreta, bien sea para que no supere unos ciertos límites) mediante el control exclusivo del parámetro $K_L a$. Es una opción muy simple y directa, especialmente si se aplica un control PID; no obstante, si se intenta minimizar el consumo de energía (es decir, minimizar el uso de $K_L a$) con un control predictivo apenas se obtiene ninguna mejora (sin tener en cuenta las tarifas) ya que, ante cualquier aumento del amonio entrante, el $K_L a$ es el único que puede mantener bajo control al amonio efluente.

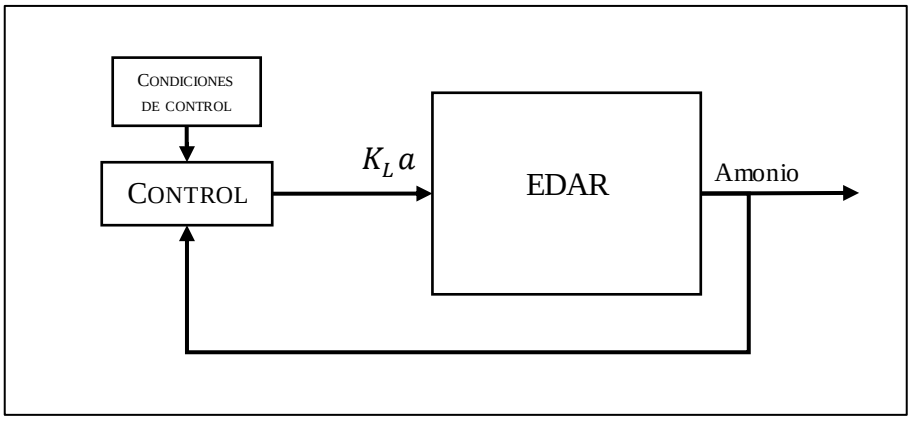

Imagen 11.- Control monovariable $K_L a$ – Amonio

- **Control multivariable del amonio usando $K_L a$ y SRI**. En este caso, el amonio se controla tanto modificando los sopladores de oxígeno ($K_L a$) y la válvula de recirculación (SRI). De esta manera, en el caso de que el amonio entrante aumente, la válvula puede compensar parte de la subida del amonio, sin necesidad de consumir energía con el $K_L a$, cumpliendo por tanto el requisito de minimización de la energía.

No obstante, la válvula presenta un problema: su efecto sobre el amonio tiene una dinámica mucho más lenta que el efecto del $K_L a$ sobre éste. Por tanto, es imprescindible una apropiada predicción del caudal entrante, así como un correcto control predictivo, para que esta opción sea viable. También es imprescindible tener en cuenta la sobreoscilación que sufre el amonio al variar la apertura de la válvula, por lo que es necesario que dicha variación sea lo más pequeña posible, hecho que puede conseguirse aplicando adicionalmente un filtro o un limitador de rampa para reducir dicha sobreoscilación.

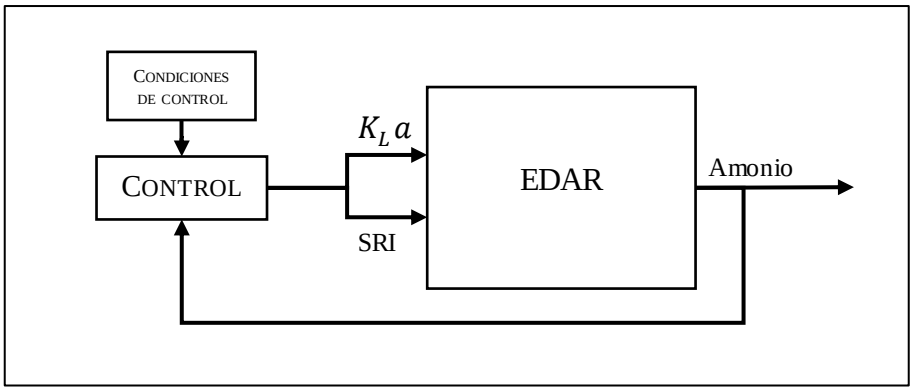

Imagen 12.- Control multivariable del amonio usando $K_L a$ y SRI

- Control multivariable del amonio y del fango usando como entradas $K_L a$ y SRI. Esta opción consiste en controlar tanto el amonio como el fango usando ambas entradas. Aunque ésta podría ser la solución ideal, hay que tener en cuenta las diferencias entre los comportamientos dinámicos de ambas salidas ante ambas entradas. De todas las dinámicas, la única que se estabiliza en un periodo inferior a un día es la del amonio al cambiar $K_L a$. Además, analizando las funciones de transferencia del sistema, se observa que modificar la válvula de recirculación SRI no afecta al valor en régimen permanente de la cantidad de fangos generado a largo plazo, y dificulta notablemente el control de $K_L a$, de manera que esta opción de control no es óptima.

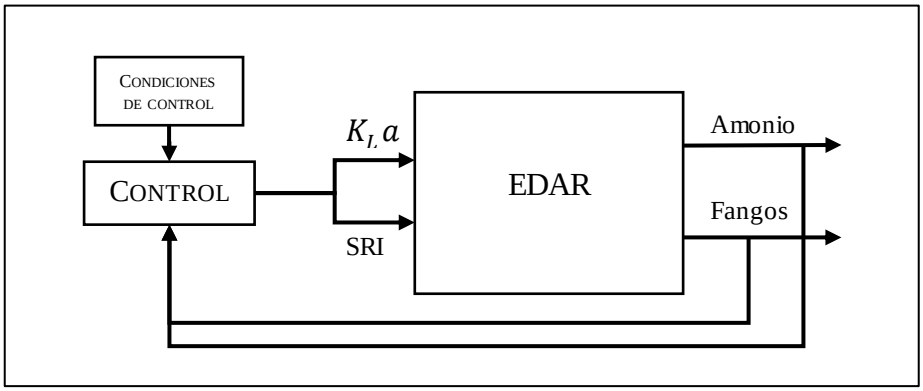

Imagen 13.- Control multivariable del amonio y del fango usando como entradas $K_L a$ y SRI

El control se ha aplicado para controlar el amonio mediante $K_L a$ y SRI. La opción de controlar el fango resulta en un control más complejo debido a las muy distintas dinámicas del amonio y de los fangos. Por otro lado, el control del amonio mediante el $K_L a$ exclusivamente no aporta suficientes grados de libertad como para aplicar una minimización del consumo eléctrico. Controlar la salida del amonio en el efluente mediante el $K_L a$ y el SRI es la única opción que permite encontrar un punto óptimo para aplicar el control predictivo.

1.7.3.- DISEÑO Y APLICACIÓN DEL CONTROL PREDICTIVO

Se ha decidido controlar la EDAR mediante un algoritmo consistente en un control predictivo, controlando la salida de amonio del efluente utilizando la $K_L a$ y la válvula de recirculación interna. La función de coste a minimizar es la siguiente:

$$J_1 = \sum_{j=1}^{Hor_{pred}} precio_{elec}(j) \cdot K_L a(t+j-1) \quad \textbf{(1.2)}$$

En esta función de coste se intenta minimizar el consumo de $K_L a$, multiplicándolo por el coste de la electricidad en ese periodo. Esta función de coste está sometida a restricciones de ratios de cambio de la válvula, limitaciones físicas tanto del $K_L a$ como de la válvula (evitar que se cierre a menos del 0%), y a que la salida del efluente no supere el límite establecido.

En caso de que dicha minimización no sea factible y se vayan a superar los límites de salida, se establece un segundo control que intenta reducir la salida del siguiente instante de tiempo bajo los límites establecidos.

$$J_2 = \left(\left(\text{limitemax}_{S_{NH}} - 1\right) - y_{amon(j+1)}\right)^2 \quad \textbf{(1.3)}$$

Esta minimización está sometida a todas las restricciones anteriores salvo a la de salida.

Sin embargo, para poder poner en funcionamiento el control predictivo, se requiere información adicional, como por ejemplo un modelo fiable del comportamiento de la EDAR, las tarifas eléctricas o una correcta predicción de las perturbaciones del caudal afluente. Todo esto configura el problema de control que se procederá a resolver en este Proyecto.

El procedimiento para resolver el problema de control se muestra en el esquema de la imagen 8. Como puede observarse, el problema puede dividirse en varios subproblemas interconectados, pero suficientemente independientes como para poder ser resueltos de manera individual.

Los subproblemas son los siguientes:

Partiendo de las ecuaciones que ofrece el Benchmark, el primer paso es realizar el modelado matemático de la EDAR. Para ello, se necesita seleccionar un **entorno** de programación adecuado para realizar la **simulación**, y escribir y adaptar las ecuaciones en función de las características de dicho entorno.

Debido a que el sistema de la EDAR es un sistema altamente no lineal, es imprescindible la **obtención de un modelo simplificado** que luego pueda utilizarse para diseñar el sistema de control. Los datos utilizados para calcular el modelo se obtienen tras realizar una simulación usando caudales procedentes de los datos ofrecidos por el Benchmark, o variaciones de éstos, mediante un **generador de caudales** distintos.

Por otro lado, los datos del Benchmark pueden ser **analizados** para obtener los **patrones de comportamiento** de la generación del caudal afluente, y generar un patrón base de predicción, que posteriormente puede **actualizarse** con valores nuevos de caudales de entrada. También es necesario **detectar la presencia de lluvia**, que puede distorsionar la predicción del caudal futuro,

El siguiente paso es la **aplicación del sistema de control**. Partiendo del método de control predictivo que se ha seleccionado, se utiliza el modelo simplificado, la predicción del caudal entrante futuro y la **selección de restricciones y parámetros** de control considerados en los requisitos para **aplicar el algoritmo de control** al sistema de la EDAR.

El último paso es la **ejecución del experimento**, en el que se realiza una simulación del comportamiento de la EDAR con caudales generados aleatoriamente, donde se aplica el método de control seleccionado.

En los siguientes apartados se mostrarán y analizarán las distintas soluciones propuestas para cada subproblema, finalizando con una selección y recapitulación final donde se vincularán todas las opciones elegidas, proponiendo un esquema de propuestas para la resolución del problema de control.

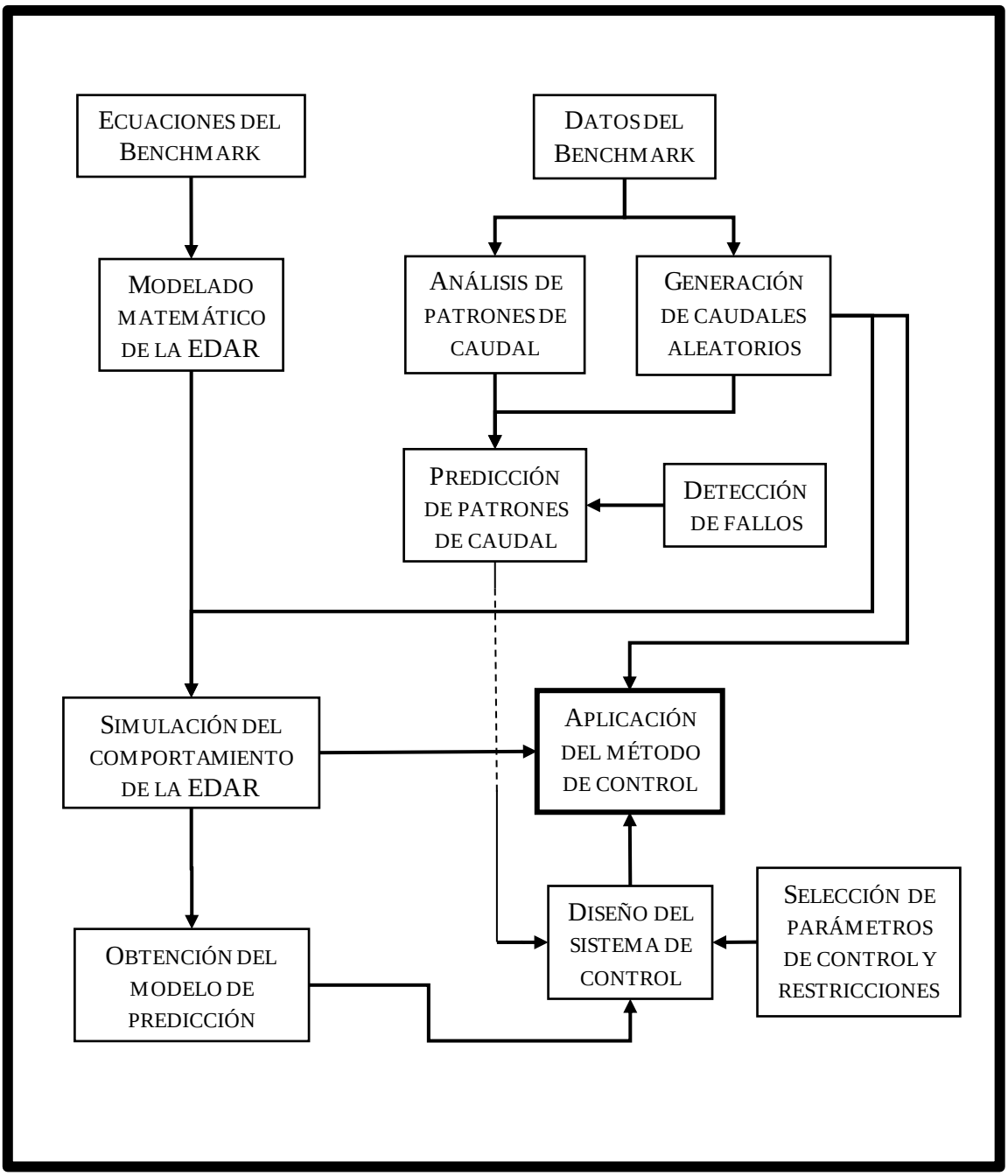

Imagen 14.- Procedimiento para resolver el problema de control

1.7.4.- ENTORNO UTILIZADO PARA REALIZAR EL MODELADO MATEMÁTICO Y LA SIMULACIÓN

El modelo matemático de la EDAR debe realizarse en un entorno o plataforma de programación que permita adaptar e implementar las ecuaciones obtenidas del Benchmark, permitiendo posteriormente la adecuada simulación del modelo no lineal. Los entornos considerados se listan a continuación.

- **WEST**. Este entorno de programación tiene la ventaja de que ya contiene las ecuaciones del ASM1, así como del comportamiento físico del decantador secundario, de manera que la cantidad de líneas de código empleadas se reduce, reduciendo el riesgo de errores durante la transcripción y adaptación y favoreciendo una mayor fidelidad y precisión respecto al modelo estandarizado de comportamiento de las EDAR. Por tanto, con este entorno, el modelado del sistema se reduce a buscar los distintos elementos almacenados en WEST, vincularlos de la manera oportuna, introducir parámetros básicos (tamaños, número de recirculaciones, etc.) y adaptar e introducir los distintos caudales y concentraciones de componentes a la entrada del sistema.

Por otro lado, como se trata de un programa orientado principalmente a la simulación de sistemas de EDAR, las posibilidades de control avanzado son más reducidas que las que podrían realizarse con otros entornos o plataformas. Aun así, WEST puede conectarse con Matlab para intercambiar información, pero requiere una sincronización adicional, pero la conexión entre Matlab y WEST, para hacer simulaciones, es muy lenta.

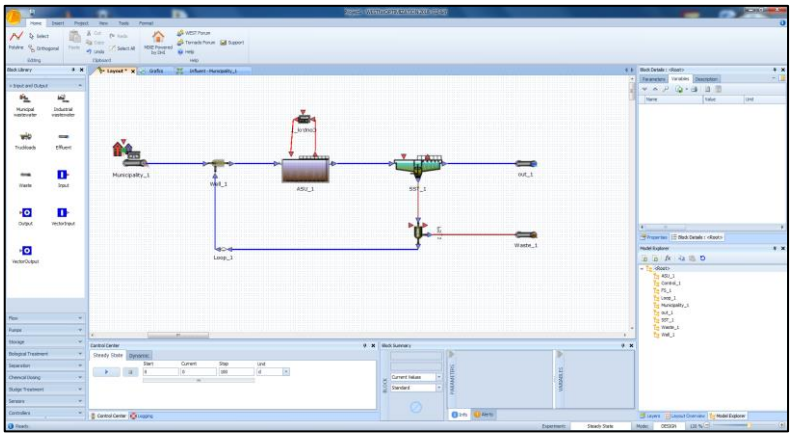

Imagen 15.- Ventana de trabajo de WEST

- **Simulink**. Esta plataforma de programación permite aplicar las ecuaciones del modelo matemático de la EDAR utilizando una simulación en tiempo continuo, por lo que la precisión de los cálculos es elevada. También tiene la ventaja de que, al disponer de una interfaz muy gráfica, permite representar el sistema de manera rápida, y la interpretación de los resultados es más directa. No obstante, la simulación en tiempo continuo comporta unos tiempos de cálculo muy elevados. Además, los bloques de funciones empleados para resolver las ecuaciones en tiempo continuo (denominadas **Level-2 Matlab S-Function**) deben ser escritas en un código muy estructurado y concreto, por lo que la conversión de las ecuaciones del Benchmark al modelo matemático puede ser compleja.

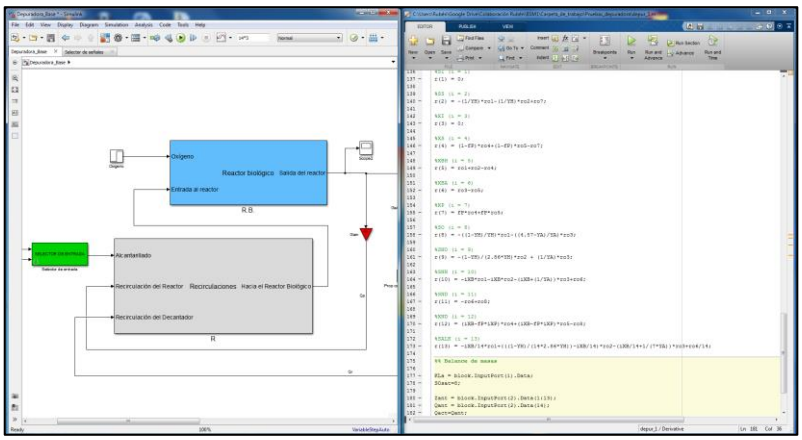

Imagen 16.- Ventana de trabajo de Simulink

- **Matlab**. En este caso, Matlab tiene la ventaja de permitir la escritura directa de las ecuaciones del Benchmark en líneas de código, además de poder clasificar, agrupar y reordenar cada apartado o problema a resolver en distintos ficheros y archivos. Es suficientemente versátil como para realizar métodos de control complejos, así como usar aplicaciones adicionales que le permitan realizar acciones más complejas, tales como optimizaciones, análisis de frecuencias o interpretación de señales. También permite realizar los cálculos de la simulación del modelo de manera mucho más rápida que Simulink.

Sin embargo, tiene la desventaja de que el modelo debe ser escrito de manera discretizada, realizando aproximaciones de Fourier de primer orden, que causan un ligero error de aproximación, por lo que el tiempo entre cálculos debe ser fijado adecuadamente para poder lograr una precisión suficiente.

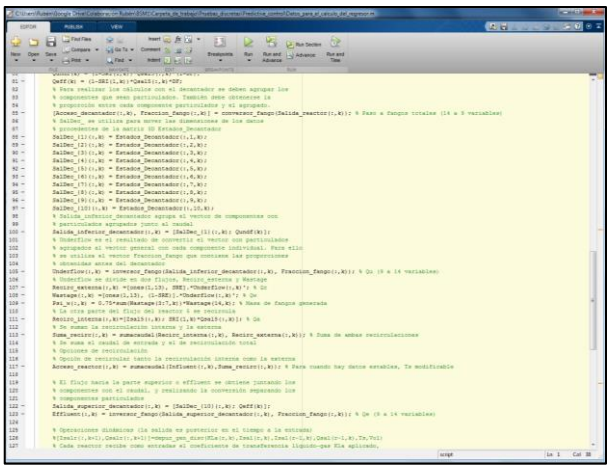

Imagen 17.- Ventana de escritura de código de Matlab

En este caso, el entorno seleccionado para modelar y simular el comportamiento del sistema es Matlab, debido a su versatilidad, su velocidad de cálculo y su capacidad para aplicar métodos de control más complejos. Por tanto, las ecuaciones deberán discretizarse para adaptarse al entorno de programación. No obstante, también se ha utilizado Simulink para verificar el comportamiento del sistema en tiempo continuo.

1.7.5.- OBTENCIÓN DEL MODELO DE PREDICCIÓN

Debido a que la EDAR es un sistema multivariable con una elevada cantidad de no linealidades, es imprescindible obtener una aproximación en forma de modelo para poder diseñar un sistema de control eficaz. Este método se denomina identificación del sistema, y las opciones consideradas son las siguientes:

- **Identificación continua mediante funciones de transferencia**. Este tipo de identificación consiste en determinar las entradas y salidas del sistema y realizar una serie de experimentos. Estos experimentos consisten en llevar al sistema a un punto de funcionamiento estable, y aplicar una variación en forma de escalón a una de las entradas para comprobar el efecto en las salidas, manteniendo el resto de las entradas en un punto estable. El experimento se repite con cada una de las entradas, y posteriormente se analizan con una herramienta de identificación de funciones de transferencia.

Esta opción es relativamente rápida y simple, y el sistema se reduce a una matriz de funciones de transferencia de $m\ salidas\ x\ n\ entradas$. Sin embargo, debido a la complejidad del sistema que se está tratando, el comportamiento transitorio identificado puede llegar a no ser muy preciso, dado que las herramientas de análisis empleadas usan una combinación de comportamientos lineales limitada, por lo que se acaba produciendo una mayor aproximación, y, por tanto, un mayor error del sistema. Además, esta opción utiliza funciones en tiempo continuo (transformadas de Laplace), por lo que es necesaria una conversión posterior a tiempo discreto.

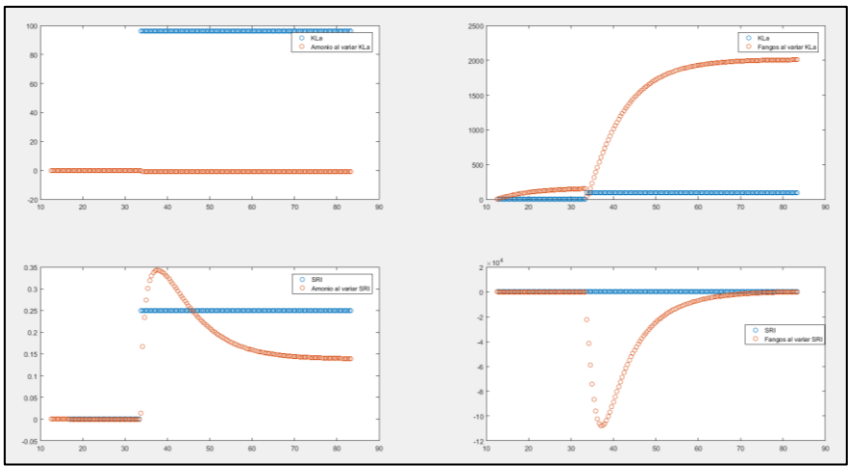

Imagen 18.- Funciones de transferencia del modelo Amonio/Fangos - KLA/SRI

- **Identificación discreta mediante el uso de regresores lineales**. Este método consiste en reducir el comportamiento del sistema analizado a un sistema de ecuaciones lineales, en el que la variación de la salida respecto a un punto estable es función de un determinado número de entradas anteriores (si se trata de un regresor no autorregresivo) o también de un determinado número de salidas anteriores (si es autorregresivo). Estos conceptos aparecen ampliados en el Anexo 2.4.1.- Cálculo de regresores.

Esta opción permite una mayor precisión que la identificación mediante funciones de transferencia, ya que el error cometido puede regularse en función de la cantidad de parámetros utilizados. Sin embargo, esto también puede convertirse en un inconveniente, pues una elevada cantidad de parámetros aumenta los tiempos de cálculo y, en el caso de que las entradas y las salidas tengan órdenes de magnitud muy diferenciados, pueden producirse errores numéricos por redondeos.

El modelo simplificado que se quiere obtener (si se desea autorregresivo) tiene la siguiente estructura (N es el orden del regresor):

$$y_k = \sum_{i=1}^{N} (a_i y_{k-i} + b_i u_{k-i}) \qquad \textbf{(1.4)}$$

Para obtener el vector de parámetros Θ que incluye a los parámetros a y b, se calcula un vector Y que incluye una serie de y_k y un vector X que incluye a todos los y_{k-i} y u_{k-i} que han precedido a cada y_k de manera ordenada. Posteriormente, se aplica $\Theta = X \backslash Y$, o lo que es lo mismo, $min((Y - X \cdot \Theta)^2)$.

- **Identificación discreta mediante el uso de regresores avanzada (con minimización de la norma)**. Este caso, similar al anterior, tiene la ventaja de permitir la modificación de un parámetro que permita regular la disparidad (varianza) entre los distintos valores del vector de parámetros, homogeneizando el resultado. Aunque tiene la desventaja de que matemáticamente es un poco más complejo que el caso anterior, permite un control más robusto al evitar posibles oscilaciones en la predicción debido a que los parámetros sean muy distintos entre sí. Este método también se ve ampliado en el Anexo 2.4.1.- Cálculo de regresores.

La ecuación aplicada, en el caso de ser autorregresivo, es:

$$min\left((Y - X \cdot \Theta)^2 + \frac{\lambda}{M}\sum_{j=1}^{M}(\Theta(j) - \mu_\Theta)^2\right) \quad \textbf{(1.5)}$$

Donde M es la longitud del vector Θ, y λ un factor de peso.

- **Identificación discreta por convolución mediante respuesta ante impulso**. Este método de identificación consiste en llevar al sistema a un estado estable, y aplicar una señal impulsional a una de las entradas. En este caso, se estudian los valores de variación de la salida, que se utilizarán como parámetros para predecir la salida en el futuro. Aunque es un método bastante simple, si la salida tarda mucho en estabilizarse se requerirán muchos parámetros para obtener un correcto modelo del sistema. Debido a que solo utiliza los valores de entrada para predecir el valor de la salida, se trata de un método no autorregresivo.

El modelo resultante tiene esta forma:

$$\hat{y}(t+k|t) = \sum_{i=1}^{N} h_i \cdot u(t+k-i|t) \quad \textbf{(1.6)}$$

Donde las h_i se obtienen directamente de los valores de salida del experimento.

- **Identificación discreta por modelos de Volterra**. Esta opción es una ampliación de los modelos de respuesta ante impulso. En este caso, los valores de salida dependen de una serie de parámetros que multiplican al cuadrado de las entradas anteriores, y, por tanto, se trata de un método no autorregresivo. Este tipo de identificación es más precisa que la anterior, aunque también más compleja, con todas las ventajas y desventajas proporcionales que conlleva.

El modelo tiene esta forma:

$$y(k) = h_0 + \sum_{i=1}^{N_1} h_1(i) \cdot u(k-i) + \sum_{i=1}^{N_2}\sum_{j=i}^{N_2} h_2(i,j) \cdot u(k-i)u(k-j) \quad \textbf{(1.7)}$$

En este caso, se ha decidido utilizar una identificación discreta mediante el uso de regresores autorregresivos con minimización de la norma, debido a que permite un suficiente equilibrio entre precisión y complejidad. De todas formas, el algoritmo está diseñado de manera que los parámetros de varianza del vector de parámetros obtenido puedan editarse, por lo que, si éstos son cero, se puede calcular un vector de parámetros por regresor sin minimización de la norma. En el Anexo 2.4.1.- Cálculo de regresores, se especifican condiciones y términos más técnicos sobre el método seleccionado.

1.7.6.- ANÁLISIS DEL CAUDAL AFLUENTE Y CREACIÓN DE PATRONES

El caudal afluente es la principal perturbación que afecta al sistema, hasta el punto de que su efecto es mucho mayor y mucho más rápido que el efecto de compensación de los actuadores. Por tanto, es imprescindible tener una predicción suficientemente acertada para poder actuar preventivamente ante estos cambios bruscos. Para ello, se procede a analizar una serie de datos con las composiciones del caudal afluente, con el objetivo de obtener algún patrón suficientemente apto que pueda funcionar como base de predicciones. Es necesario encontrar un correcto equilibrio entre el número de parámetros empleados en el patrón y la precisión de dicho patrón aproximado.

Las opciones se muestran a continuación:

- **Tablas**. Esta opción consiste en ordenar los datos de caudal entrante de una semana entera en una tabla que relacione cada dato con la fecha y la hora que le corresponde, de manera que la predicción del caudal en el futuro consista en seleccionar el dato de la tabla situado en el momento temporal indicado. Este método tiene la ventaja de ser muy directo, simple y con la mejor precisión de todas las opciones consideradas, y permite una actualización de los datos muy rápida. Sin embargo, tiene el inconveniente de requerir tantos parámetros como datos temporales almacenados, hecho que puede ralentizar un poco la ejecución de los programas.

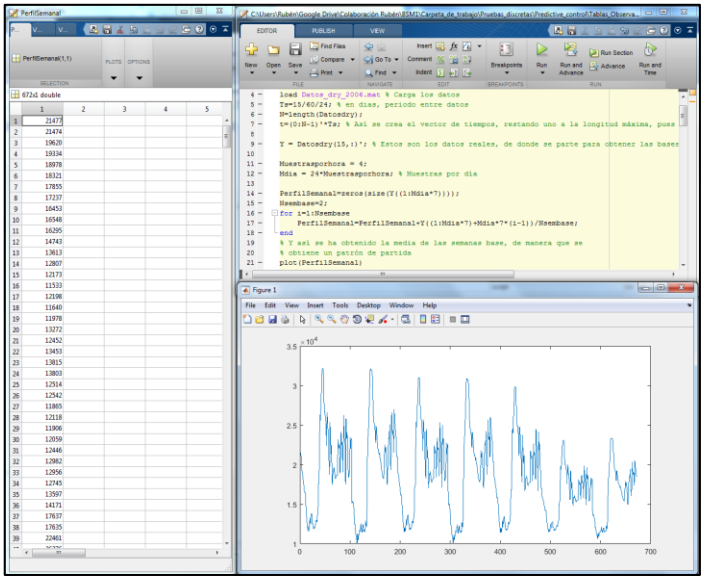

Imagen 19.- Representación del patrón en tablas

- **Series de Fourier**. Este método consiste en aproximar los datos semanales del caudal de entrada a una serie de Fourier, es decir, a una combinación lineal de senos y cosenos. Para obtener los parámetros de dicha combinación lineal, se utiliza un regresor, cuyo orden depende de la cantidad de harmónicos que se quieran considerar. Esta opción requiere una cantidad más reducida de parámetros, pero es imprescindible realizar previamente un análisis del espectro de frecuencias de la señal para poder concretar cuáles son sus frecuencias fundamentales.

Por tanto, el patrón queda aproximado a una ecuación con la siguiente estructura:

$$y(t) = a_0 + \sum_{i=1}^{H} \left(b_i \cdot sin(i \cdot \omega \cdot t) + c_i \cdot cos(i \cdot \omega \cdot t) \right) \qquad \textbf{(1.8)}$$

Donde H es el número de harmónicos de ω considerados.

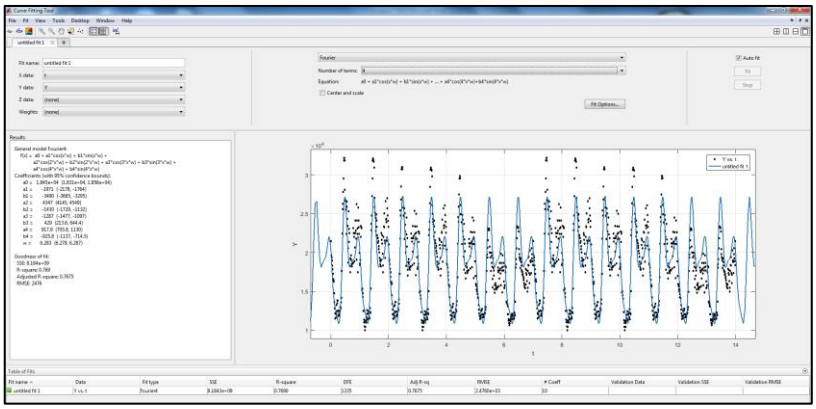

Imagen 20.- Aproximación por Fourier mediante una aplicación de Matlab

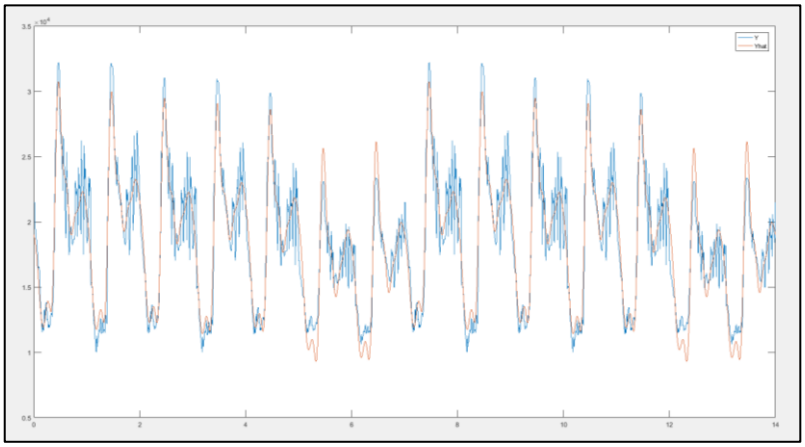

Imagen 21.- Aproximación por series de Fourier mediante regresores

- **Series de Fourier no lineales**. Este método es similar al anterior, solo que en este caso no se obtiene una combinación puramente lineal de senos y cosenos, sino que también aparecen términos consistentes en potencias y productos cruzados de dichas funciones sinusoidales. Requiere más parámetros que el caso de las series de Fourier, pero también es más preciso. De todas maneras, sigue requiriendo el análisis del espectro de frecuencias de la señal.

$$y(t) = a_0 + \sum_{i=1}^{H} (b_i \cdot sin(i \cdot \omega \cdot t) + c_i \cdot cos(i \cdot \omega \cdot t) + d_i \cdot sin(i \cdot \omega \cdot t)^2 + e_i \cdot cos(i \cdot \omega \cdot t)^2) \quad \textbf{(1.9)}$$

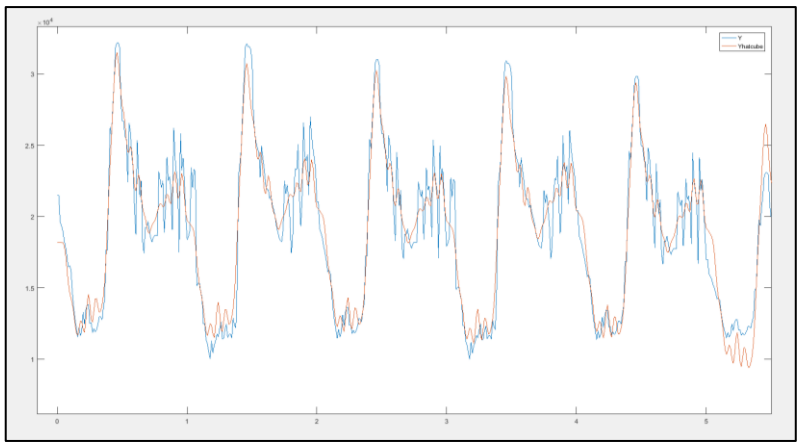

Imagen 22.- Aproximación por series de Fourier no lineales mediante regresores

- **Polinomios**. Este caso consiste en adaptar la forma de la curva semanal o diaria mediante una función polinómica dependiente del tiempo.

Esta opción puede ser útil si el patrón de entrada no es muy complejo ni oscilatorio; de no ser así, por muy elevado que sea el polinomio, el error de aproximación sigue siendo demasiado alto para poder ser una solución útil.

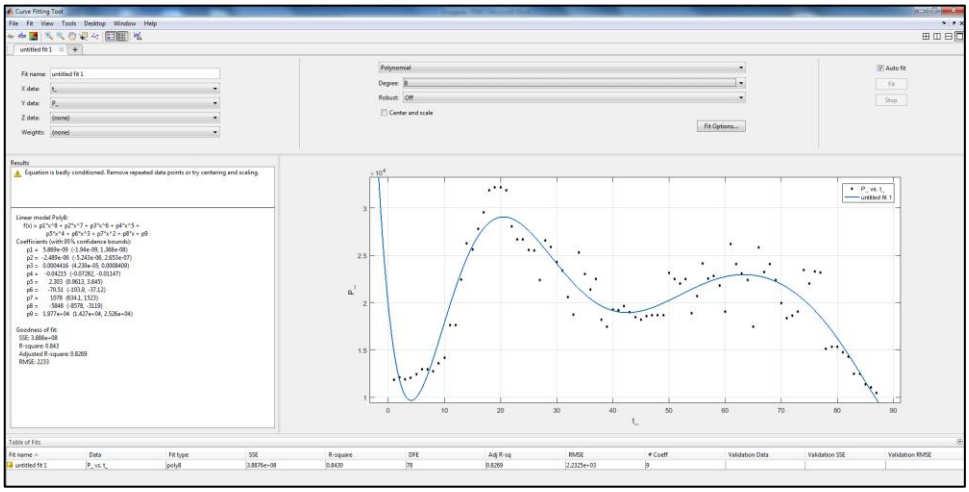

Imagen 23.- Aproximación por polinomios usando una aplicación de Matlab

- **Splines**. Este método es una mejora de la opción anterior, pues en este caso la señal se divide en múltiples *splines* o curvas que son polinomios de grado reducido, generando una función a trozos continua y derivable.

Permite conseguir una precisión casi tan elevada como la de las tablas, pero al precio de un número de parámetros igual o incluso superior, por lo que resulta menos apropiada. Además, su transcripción a ecuaciones en el código dificultaría su posterior tratamiento y actualización.

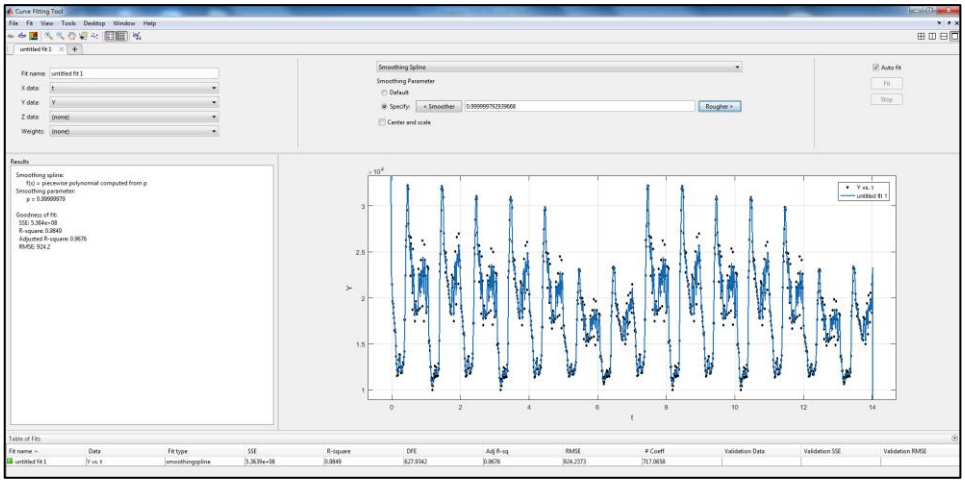

Imagen 24.- Aproximación por splines usando una aplicación de Matlab

- **Redes neuronales**. En este caso, el procedimiento sería entrenar una red neuronal para que aprendiese a detectar los patrones de caudal entrante y realizar predicciones por su cuenta. Sin embargo, se trata de una solución demasiado complicada, que requeriría muchos datos de entrada (de los que no se dispone), y hay listados aquí métodos más simples que producen un resultado suficientemente acertado.

Imagen 25.- Neural Fitting app, de Matlab

En este caso, se ha seleccionado el uso de tablas, porque a pesar del número de variables a almacenar, el proceso de actualización posterior es mucho más sencillo. De todas maneras, el uso de series de Fourier también se ha utilizado para realizar un análisis más exhaustivo de los patrones de comportamiento.

1.7.7.- DETECCIÓN DE LA ENTRADA SÚBITA DE LA LLUVIA Y ACTUALIZACIÓN DEL PATRÓN BASE

Todos los métodos antes comentados se realizan antes del experimento de control, es decir, *off-line*. Sin embargo, durante el experimento entrarán nuevos valores de caudal, y por tanto el patrón base obtenido previamente debería ir actualizándose con esos nuevos valores. No obstante, el patrón obtenido se refiere al comportamiento habitual de generación de aguas residuales, y los cambios bruscos (como, por ejemplo, lluvia o tormentas) pueden provocar un aumento inesperado del caudal entrante, así como una modificación en las concentraciones de sus componentes.

Hay varias opciones a considerar en este caso:

- **Ignorar la presencia de cambios bruscos en la entrada del sistema**. Esta opción consiste en considerar que la actualización del patrón base, modelizada como un filtro IIR, está diseñada para que el efecto de las nuevas entradas tenga un efecto muy reducido en el valor del patrón base. Aunque este método produce una cierta inmunidad al patrón, también provoca que la dinámica del sistema de predicción sea muy lenta, y no sea capaz de percibir cambios debidos a la estacionalidad. Esta opción es más recomendada si se ha considerado utilizar un patrón base diario, en lugar de semanal.

- **Detectar la presencia de lluvia y aplicar un filtrado**. Este método consiste en detectar la existencia de un sobrecaudal de entrada, estimando las diferencias entre el caudal de entrada y el patrón base, y aplicando un coeficiente de fiabilidad diferenciando entre lo que realmente es un sobrecaudal, el error propio del caudal base y el ruido de medida. Toda esta teoría aparece explicada en el Anexo 2.6.4.- Detección de la lluvia. La lluvia estimada se puede restar al valor de caudal de entrada, y entonces se aplica la actualización del patrón. Esta opción permite mejorar la dinámica del sistema de actualizaciones, pues permite que las nuevas entradas tengan un efecto mayor en el sistema. Por otra parte, aparece un nuevo parámetro de regulación, el coeficiente de fiabilidad mencionado anteriormente, que debe ajustarse correctamente para no distorsionar los datos.

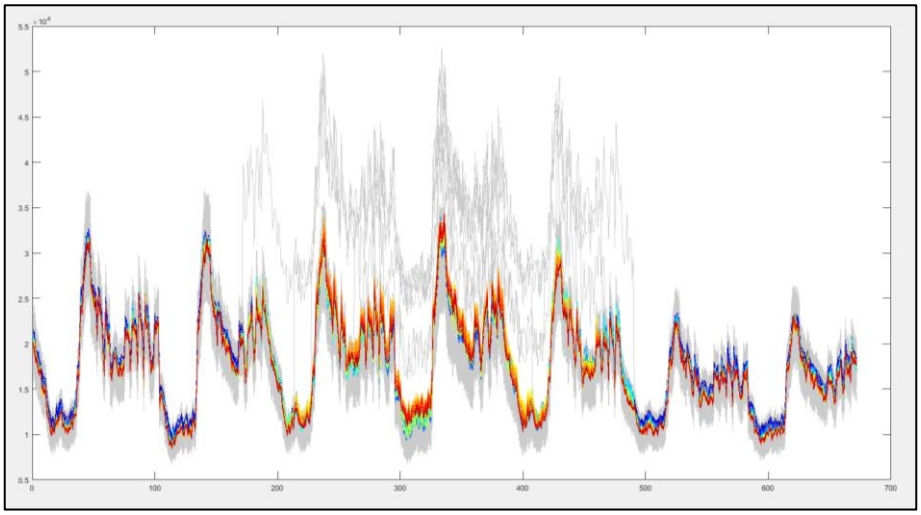

Imagen 26.- Actualización del patrón base usando caudales con lluvia añadida (en gris)

En este proyecto se ha decidido detectar la presencia de lluvia y aplicar un filtrado. A pesar de necesitar un algoritmo propio, no solo es más preciso a largo plazo que ignorar la lluvia, sino que facilita mucho más la regulación de los parámetros de actualización del patrón de predicción.

1.7.8.- MÉTODOS DE GENERACIÓN DE CAUDALES

Para realizar una simulación del modelado de una EDAR suficientemente realista, es imprescindible que los caudales de entrada tengan una cierta variación en su comportamiento semana a semana, bien sea debido al ruido inherente a todo sistema como al comportamiento aleatorio de la generación de aguas residuales por parte de las urbes y las industrias. Por tanto, no es idóneo repetir exclusivamente los datos de dos semanas ofrecidos por el Benchmark una y otra vez durante todo el experimento. Por tanto, para conseguir nuevos caudales puede emplearse alguna de estas opciones:

- **Uso de datos reales**. La opción de disponer de datos reales de una EDAR real parece, en un principio, óptima, pero no tiene por qué ser así; seguramente, esta EDAR estará diseñada con unos volúmenes, capacidad y condiciones distintas a las establecidas por el Benchmark, por lo que es bastante probable que la simulación o bien sea incapaz de asimilar los caudales y concentraciones entrantes, o bien serán tan reducidos que el método de control dará unos resultados incorrectos.

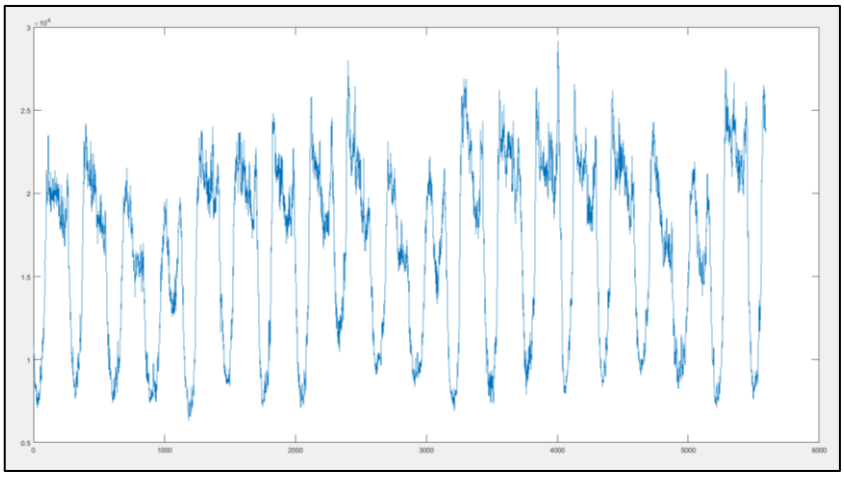

Imagen 27.- Datos procedentes de depósitos de Almazora

- Aplicación de valores aleatorios sobre un patrón base medio procedente del Benchmark. Esta opción consiste en obtener la media entre los valores de la primera semana y los de la segunda, ofrecidos por el Benchmark. Posteriormente, cada vez que se requiera una semana nueva para la simulación, se genera una semana nueva a partir de ese patrón medio, al que se le aplica un valor aleatorio representando el ruido del sistema, otro valor aleatorio representando el comportamiento errático del consumo hidráulico por parte de la gente, y un valor constante representando la estacionalidad. Además, se le añade, también de manera aleatoria, otro vector con un caudal elevado simbolizando periodos de lluvia.

Este método es idóneo para realizarlo con patrones escritos en tablas, de manera que se reduce el número de conversiones. Es muy editable y como puede observarse, tiene en cuenta distintos fenómenos de consumo. Sin embargo, tiene la desventaja de que, al basarse en el mismo patrón medio procedente del Benchmark, al realizar las actualizaciones del patrón base, éste tenderá al patrón medio de manera relativamente rápida.

Hay que tener en cuenta que en la imagen 28 solo se han representado los valores de caudal, pero estas variaciones aleatorias también pueden aplicarse al resto de concentraciones de caudal entrante, si bien con una menor variabilidad.

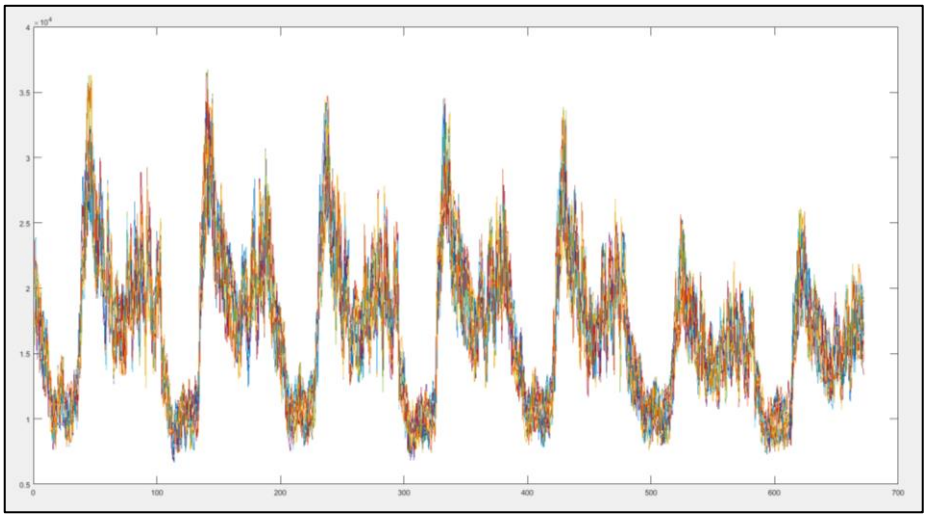

Imagen 28.- Valores aleatorios sobre un patrón base medio procedente del Benchmark

- **Aplicación de valores aleatorios sobre distintos días procedentes del Benchmark**. Esta opción es una variante del anterior, pero no parte de un patrón base medio. En este caso, los datos de dos semanas distintas procedentes del Benchmark se clasifican en los distintos días de la semana, y el patrón de caudal que se genera procede de una media ponderada con pesos aleatorios distinta para cada día de la semana. Posteriormente se aplican los mismos vectores aleatorios que en el caso anterior, y tiene la ventaja de que las actualizaciones tardan más en tender al patrón base medio.

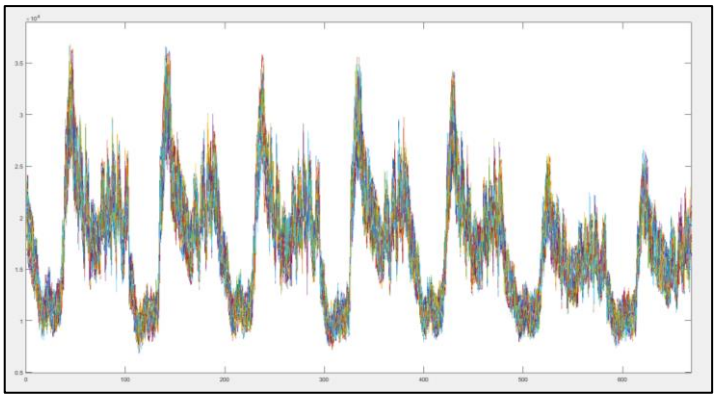

Imagen 29.- Valores aleatorios sobre distintos días procedentes del Benchmark

- **Aplicación de caudal variable con concentraciones constantes**. Este caso consiste en simplificar las entradas en el sistema, considerando que los valores de las concentraciones se mantienen constantes durante todo el tiempo. El tratamiento de datos sería similar al de las opciones anteriores, solo que solo se usarían las variaciones de caudal entrante, mientras que las concentraciones serían las del punto de equilibrio del sistema. Esta simplificación es posible cuando a la entrada del sistema hay un depósito de volumen muy elevado, y en el que la condición es que el caudal entrante es igual al caudal saliente. Este caso es plausible, dado que antes de la entrada al tratamiento secundario (que es la parte que de la EDAR que simulan las ecuaciones del BSM1) está el tratamiento primario, consistente en una serie de embalses de gran tamaño en el que las aguas entrantes se dejan reposar durante elevados periodos de tiempo. Esta simplificación se demuestra en el Anexo 2.5.2.- Simplificación del caudal afluente.

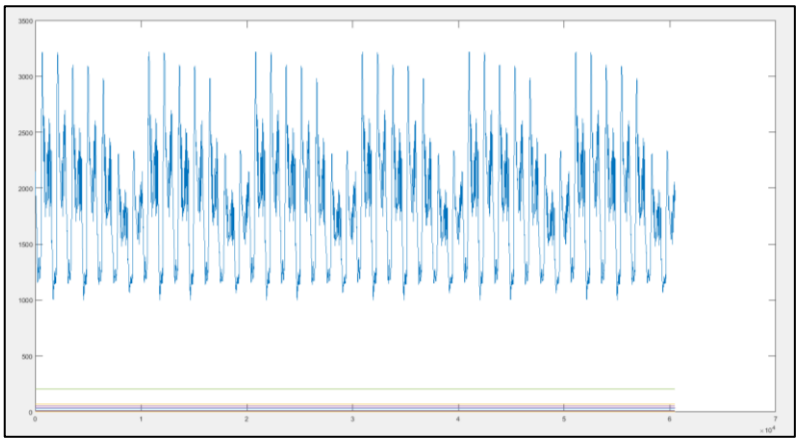

Imagen 30.- Caudal variable con concentraciones constantes

La opción seleccionada para este proyecto es la última, la aproximación de concentraciones constantes. Se ha preferido dicha opción para poder lograr una predicción del caudal entrante menos no lineal, simplificando un poco los cálculos de la obtención del modelo lineal. No obstante, una ampliación del presente Proyecto debería tener en cuenta una cierta variación de las concentraciones entrantes.

1.7.9.- SELECCIÓN FINAL

Una vez consideradas todas las alternativas y seleccionadas las opciones más adecuadas, se procede a realizar una recapitulación de todas ellas:

El programa utilizado para el modelado y simulación de la EDAR es **Matlab**, debido a su mayor velocidad y a la posibilidad de realizar complejos algoritmos de control. Por otro lado, el modelo va a tener que discretizarse.

El modelo se ha identificado de manera discreta mediante el uso de **regresores**. Para mejorar su comportamiento, evitando el efecto de retardos y de oscilaciones indeseadas, se ha aplicado un método de control avanzado con dichos regresores, que utiliza la **minimización de la norma**.

El caudal entrante se ha determinado mediante el uso de **tablas** que almacenan todos los datos, ofreciendo la máxima precisión disponible a la hora de identificar los patrones. Estas tablas van a actualizarse y a realizar la **detección de la lluvia** mediante un observador de fallos. El caudal afluente a la depuradora se ha generado de manera aleatoria teniendo en cuenta la simplificación de la **concentración constante**.

El sistema de control seleccionado es el control **predictivo**, debido a las múltiples ventajas que ofrece respecto al resto de opciones, y se ha controlado la salida de **amonio** mediante las entradas $\boldsymbol{K_L a}$ y la válvula de recirculación interna (**SRI**).

Para aplicar las restricciones y los objetivos del control predictivo, se han determinado una serie de condiciones procedentes de los requisitos: el control se ha diseñado teniendo en cuenta la minimización del coste del **consumo eléctrico**, por lo que se han aplicado valores de **tarifas** eléctricas obtenidas desde webs.

Se ha considerado que únicamente la acción de control $K_L a$ tiene impacto económico, así que el movimiento de la válvula, que se ha intentado limitar al máximo, no consume energía a tener en cuenta. No se ha aplicado ningún control directo a la salida de fangos debido a que su minimización es proporcional a la minimización de $K_L a$. Por último, el efluente amónico se controla para mantenerlo bajo unos **límites** en su vertido.

En la Imagen 31 se puede observar el esquema final con el procedimiento seguido para resolver el sistema de control.

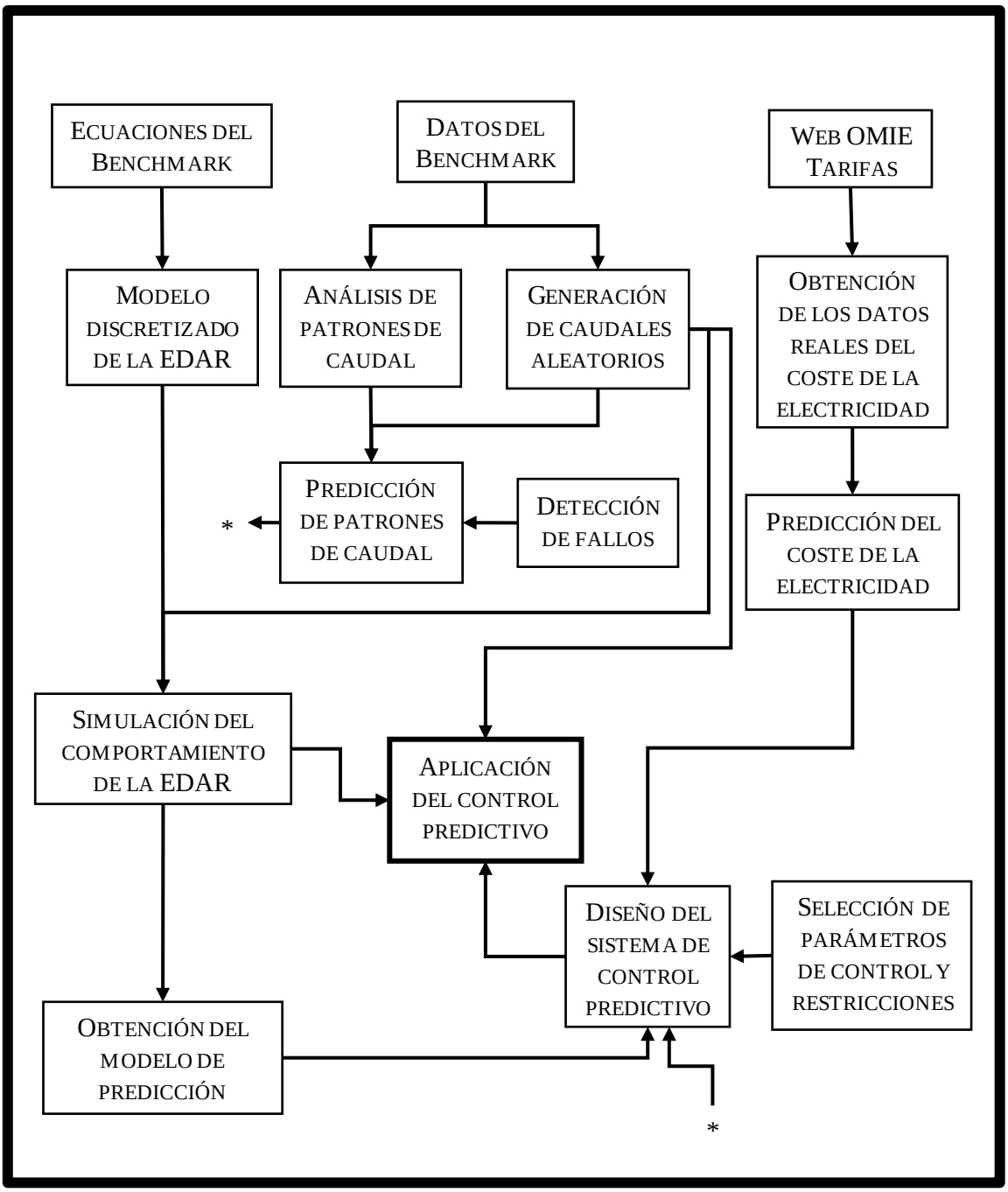

Imagen 31.- Procedimiento seguido para resolver el problema de control

1.8.- RESULTADOS FINALES

En este apartado se procederá a realizar el análisis de los resultados obtenidos en los experimentos de control en los que se ha aplicado el algoritmo de control predictivo, comprobando que cumple los requisitos indicados. Posteriormente, se procederá a realizar una comparación con un experimento realizado con un control tipo relé bajo las mismas condiciones de caudal afluente, para comprobar el ahorro producido con el uso del algoritmo de control predictivo.

1.8.1.- PARÁMETROS DE CONTROL USADOS EN LOS EXPERIMENTOS

El algoritmo de control predictivo utilizado en el presente experimento ha sido diseñado con los siguientes parámetros:

La frecuencia de muestreo utilizada para obtener el modelo simplificado ha sido de 1 hora, manteniéndose ésta como frecuencia de cálculo del optimizador. El orden del regresor, el cual es autorregresivo con minimización de la norma, es de 13 horas, de manera que los cálculos de predicción utilizan los valores de las entradas y salidas de esas 13 h anteriores.

El control predictivo realiza los cálculos de optimización para las próximas 32 horas (horizonte de predicción). Esto significa que toma también esa cantidad de valores del patrón de caudal previsto. La duración de la simulación es de unos 40 días, de manera que el control predictivo realiza los cálculos 952 veces. Se ha considerado una restricción de salida de amonio de 9 $g\ N/m^{-3}$. Los caudales utilizados son los procedentes del tiempo seco (con concentración constante) del Benchmark; no se han utilizado para la comparación los caudales generados aleatoriamente, de forma que se facilite una mejor visualización de los resultados y de las diferencias entre ambos métodos de control. Por último, se ha considerado una tarifa con tres periodos distintos, incluso en fin de semana, para valorar mejor la robustez del sistema.

Por otra parte, el relé se ha configurado de manera que si el amonio supera el límite establecido, se enciende el $K_L a$ al máximo (240 d^{-1}), y si baja de dicho valor, el $K_L a$ se apaga. En este caso, la válvula se mantiene fija.

1.8.2.- VERIFICACIÓN DEL FUNCIONAMIENTO DEL CONTROL PREDICTIVO

1.8.2.1.- COMPORTAMIENTO DEL PATRÓN DE AMONIO EFLUENTE

Una vez ejecutado el experimento utilizando el algoritmo de control predictivo, se procede al análisis de los resultados. La primera gráfica obtenida muestra los valores de caudal entrante (divididos entre 10000, en rojo), y los valores del amonio saliente (en azul). Debido a que se realiza la simplificación del caudal de entrada con concentración constante, un aumento del caudal entrante conlleva necesariamente un aumento de la masa entrante de amonio.

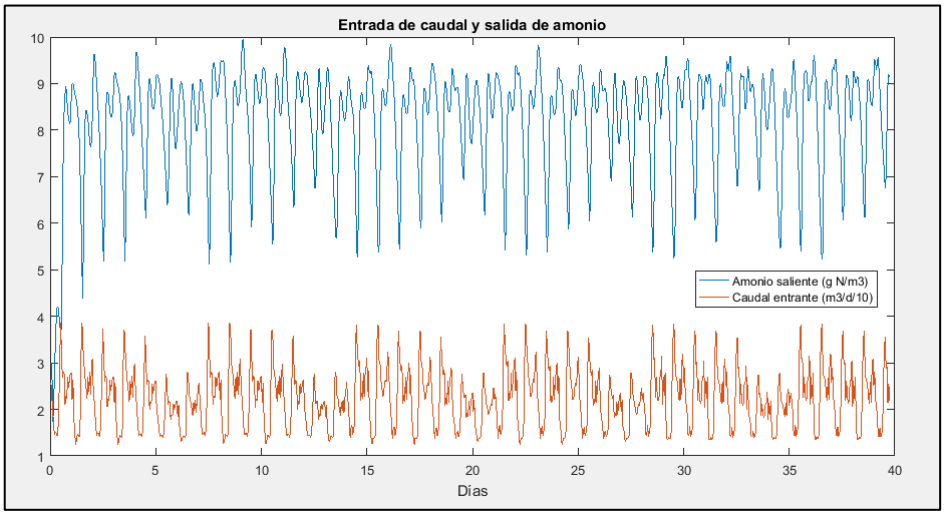

Imagen 32.- Caudal afluente y efluente amónico (Control predictivo)

En la imagen 33, se observa una ampliación de la gráfica anterior. Puede observarse cómo el patrón del caudal de entrada se repite en la salida, debido a que ni la propia EDAR ni el control pueden eliminar por completo el comportamiento de la señal. También puede observarse un ligero retardo, del orden de horas, entre el patrón entrante y el saliente, causado por el tiempo de tratamiento interno de la depuradora.

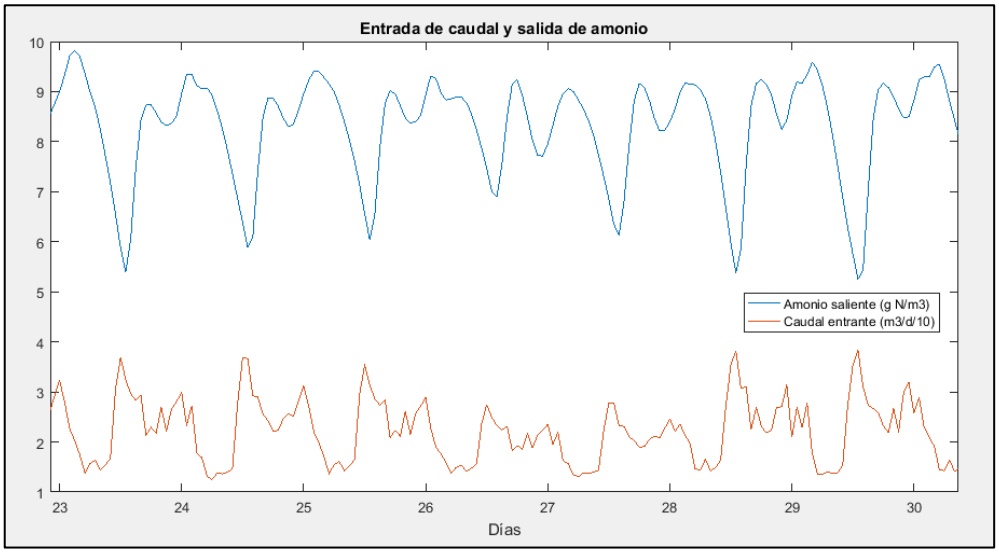

Imagen 33.- Caudal afluente y efluente amónico (Control predictivo) – Ampliación

1.8.2.2.- COMPROBACIÓN DE LOS LÍMITES DEL AMONIO EFLUENTE

El algoritmo de control predictivo se ha diseñado indicando un límite superior permitido de salida efluente de amonio de 9 $g\ N/m^{-3}$. Como puede verse en la imagen 34, la salida de amonio (en azul), supera a veces el valor límite, pero la media diaria (en naranja) se mantiene bastante por debajo, que es el valor relevante.

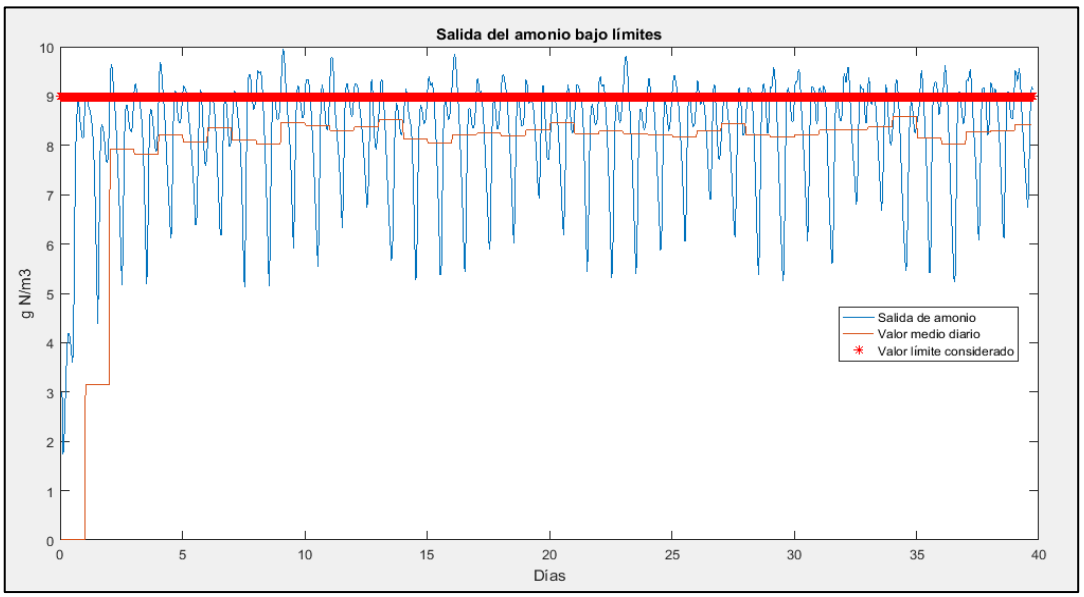

Imagen 34.- Efluente amónico bajo límites (Control predictivo)

1.8.2.3.- COMPROBACIÓN DE LA MINIMIZACIÓN DEL COSTE DEL CONSUMO ELÉCTRICO

El siguiente punto por comprobar es que el control predictivo sea capaz de realizar correctamente sus predicciones, y minimice el consumo eléctrico de manera que en periodos de coste elevado solo aplique la $K_L a$ si es imprescindible, y que la mayor parte del consumo eléctrico se realice en periodos con costes reducidos, incluso consumiendo en exceso para anticiparse a futuras subidas previstas.

En la imagen 35, se puede observar el comportamiento de $K_L a$ (en la parte superior, en azul). Se puede observar cómo intenta evitar el consumo en periodos con costes más elevados (coste en naranja, multiplicado por 100), a no ser que debido a un aumento (o predicción de aumento) del caudal afluente (y, por tanto, de masa amónica), le obligue a activar $K_L a$. En este caso, puede observarse (en el día 31,5) como el pico del caudal entrante (en la gráfica inferior, en azul) le obliga a activar $K_L a$, pero al mismo tiempo, se observa cómo el pico posterior correspondiente no llega a superar el límite establecido, cuando lo normal sería que lo hubiese sobrepasado notablemente.

Por otro lado, se puede observar cómo en los periodos más bajos, el $K_L a$ está prácticamente todo el rato encendido, incluso en momentos donde la salida amónica se ha reducido por debajo del límite. Eso se debe a que predice el comportamiento del caudal entrante, y sabe que posteriormente va a aumentar tanto el precio como el amonio entrante, de manera que actúa preventivamente.

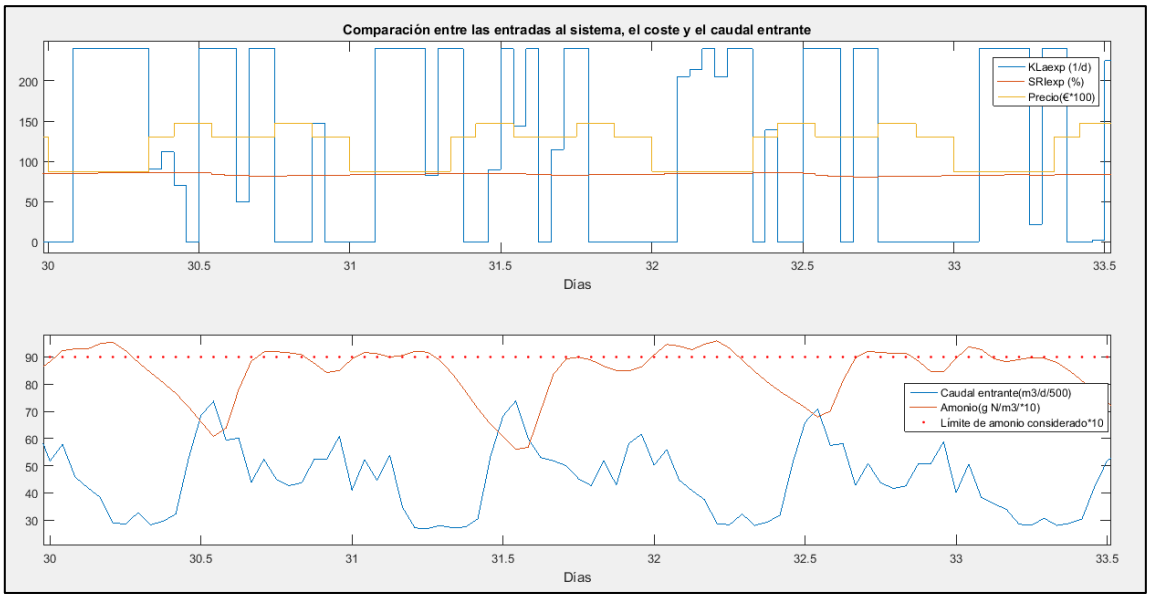

Imagen 35.- Comportamiento de las entradas respecto al coste (Control predictivo)

Por otro lado, se puede comprobar cómo la válvula no cambia en exceso. Esto se debe a que los cambios en la válvula de recirculación tienen un efecto a largo plazo y un impacto relativamente elevado en la salida amónica. Para visualizar mejor los cambios, en la imagen 36, puede observarse el comportamiento normalizado de las entradas y del coste. Puede observarse que, en momentos de coste más bajo, la válvula tiende a abrirse, provocando un aumento en la salida de amonio que se encarga de neutralizar $K_L a$. Más tarde, en periodos más críticos, con coste más elevado y con mayor entrada de amonio desde el afluente, como la válvula ha podido abrirse más, puede cerrarse minimizando la salida de amonio sin necesidad de utilizar el $K_L a$, y manteniendo el efluente bajo los límites exigidos.

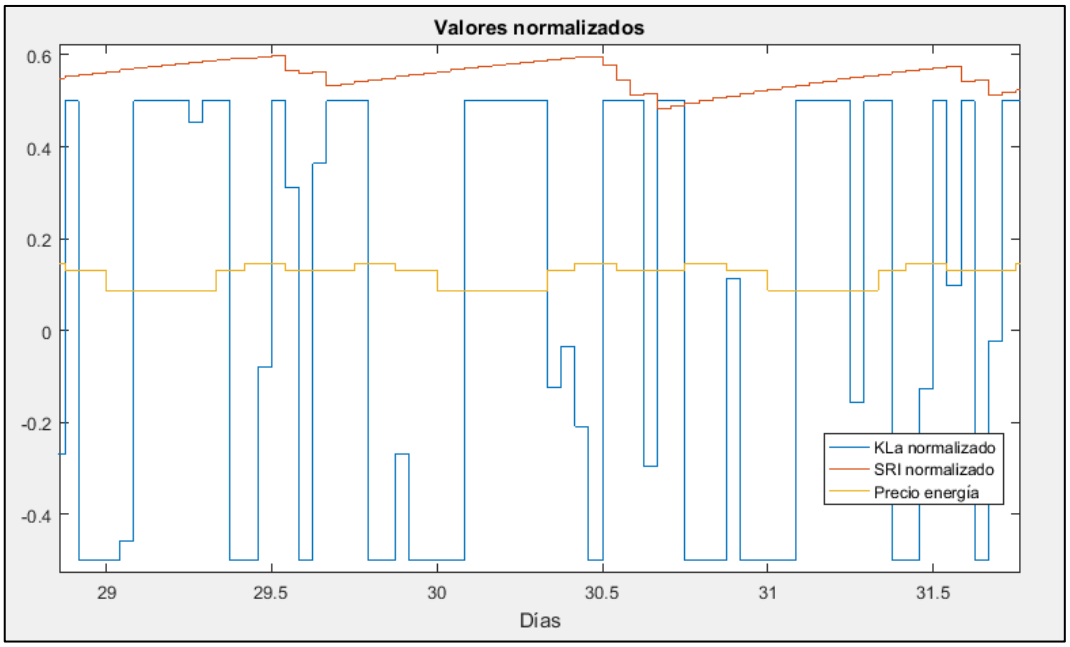

Imagen 36.- Entradas normalizadas respecto al coste (Control predictivo)

1.8.3.- COMPARACIÓN CON EL CONTROL TIPO RELÉ

1.8.3.1.- FUNCIONAMIENTO DEL EXPERIMENTO TIPO RELÉ.

Tras realizar el experimento aplicando el algoritmo de control de tipo relé con las condiciones especificadas en el primer punto de este apartado, se procede a analizar los resultados obtenidos.

Si se observa el comportamiento de limitación del efluente amónico (Imagen 37), se puede comprobar que la salida presenta un comportamiento mucho más oscilatorio que en el experimento con el control predictivo. Además, esta salida supera mucho más frecuentemente el límite superior exigido, de manera que el valor medio diario está mucho más cerca de dicho límite, llegando a cruzarlo contadas veces.

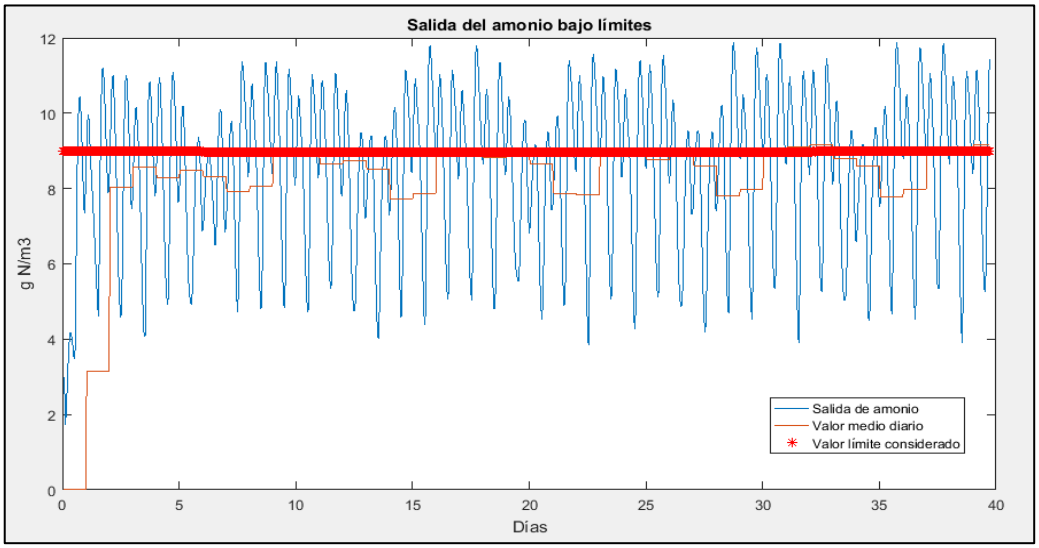

Imagen 37.- Efluente amónico bajo límites (Control relé)

Por tanto, ya se observa que el control tipo relé no cumple el requisito de no superar el límite, por lo que funciona de peor manera que el control predictivo. Viendo la imagen 38, puede analizarse el comportamiento de la activación de las entradas en función del coste. Como la válvula se mantiene fija en un punto estable y la $K_L a$ responde exclusivamente a que la salida de amonio cruce el umbral establecido, ésta se activa independientemente de los costes, dando lugar a que consuma energía indiferentemente en los periodos más caros y en los más baratos. Por tanto, cumple mucho peor el requisito de ahorro energético y económico.

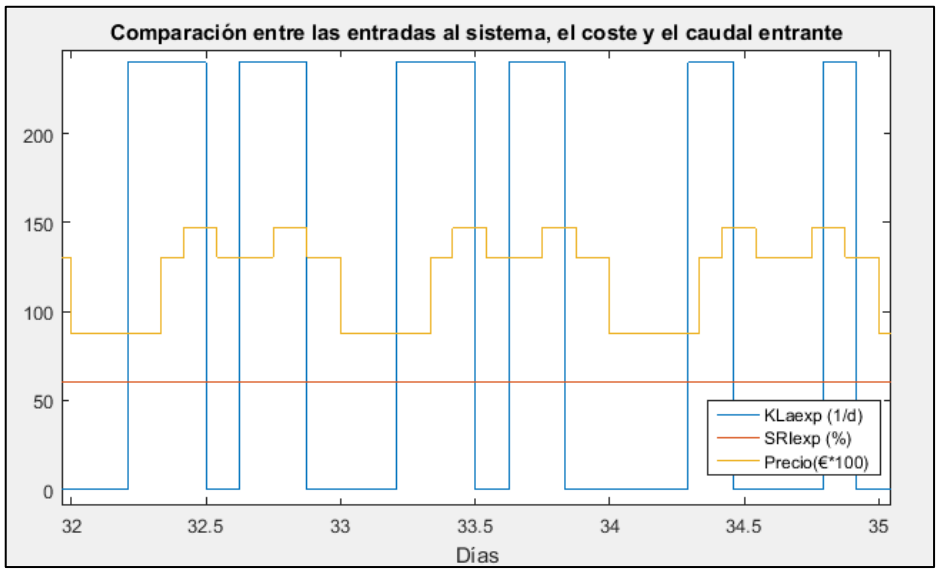

Imagen 38.- Comportamiento de las entradas respecto al coste (Control relé)

1.8.3.2.- COMPARACIÓN DE LOS COSTES TOTALES

Comparando los valores de coste medio diario obtenidos en el experimento con el algoritmo de control predictivo (en naranja) con los obtenidos con el algoritmo de control tipo relé (en azul), puede observarse una clara mejora, especialmente los fines de semana (utilizando la tarifa oportuna, los valores serían aún serían menores). El ahorro total es, realizando los cálculos (ver Apéndice A.6), de aproximadamente un 25%. Este cálculo se concretará en el siguiente apartado.

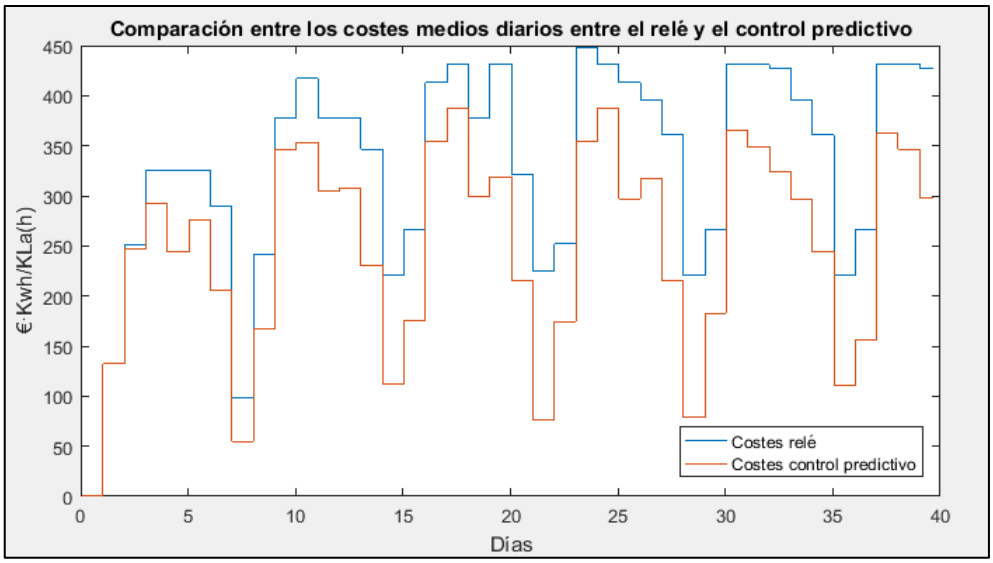

Imagen 39.- Comparación de costes entre métodos de control

1.9.- ESTUDIOS DE VIABILIDAD

Para realizar el estudio de viabilidad, hay que partir de los dos experimentos realizados en el Apartado 1.8. En ellos, se ha calculado el valor del coste total consumido por el $K_L a$, teniendo en cuenta el precio de cada hora. En el caso del experimento con el algoritmo de control predictivo, el resultado de coste total obtenido es de 9963 € $\frac{K_L a(h)}{KWh}$. En el caso del control tipo relé, el coste total es de 13185 € $\frac{K_L a(h)}{KWh}$. El ahorro obtenido es, por tanto, de $\left(\frac{13185-9963}{13185}\right) \cdot 100 = 24{,}43\%$

Siguiendo el Benchmark, se ha considerado la simplificación de que la energía consumida es directamente proporcional al $K_L a$ consumido. El coeficiente de proporcionalidad es función de parámetros fijos de diseño de la EDAR, como es el volumen de los compartimentos.

La ecuación del Benchmark, teniendo en cuenta que $S_O^{sat} = 8\ g/m^3$, es la siguiente:

$$\text{Energía}_{K_L a}[KWh] = \frac{S_O^{sat}}{\Delta t \cdot 1{,}8 \cdot 1000} \int_{t_0}^{t} V_{as,5} \cdot K_L a_5\,(t) \qquad \textbf{(1.10)}$$

Operando con los valores anteriores, se obtiene que la energía ahorrada es el 24,43 % de la consumida por el relé, que es el método típicamente usado en las depuradoras hoy en día. Por tanto, la energía ahorrada de manera diaria (teniendo en cuenta que el experimento duró 40 días y un $V_{as,5}$de 1333 m^3) es de:

$$Ahorro\ \left(\frac{€}{\text{día}}\right) = \frac{8}{40 \cdot 24 \cdot 1{,}8 \cdot 1000} \cdot 1333 \cdot 13185 \cdot 0{,}2443 = 19{,}877\ \frac{€}{\text{día}} \qquad \textbf{(1.11)}$$

No se calcula el gasto en otros compartimentos del reactor porque no se puede controlar su $K_L a$.

Tomando los valores de coste total del proyecto desde el punto 4.2. del presupuesto, el coste total del experimento es de 13983,73 €, por lo que el Periodo de Retorno sería de:

$$PR\ (\text{años}) = \frac{13983{,}73}{365 \cdot 19{,}877} = 1{,}9274\ \text{años} \qquad \textbf{(1.12)}$$

1.10.- PLANIFICACIÓN

Dado que este Proyecto tiene un carácter principalmente investigativo, no es posible determinar una serie concreta de pasos orientados a la compleción de dicho Proyecto. Esto es debido a que a lo largo del desarrollo de los distintos programas no se ha seguido exclusivamente una única línea de investigación, y en muchos casos se han realizado avances y retrocesos durante las distintas selecciones de las técnicas y de los algoritmos.

1.11.- ORDEN DE PRIORIDAD ENTRE LOS DOCUMENTOS

El orden de prioridad entre los documentos del presente proyecto se establece según la norma 157001:2014 "Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico".

1.12.- CONCLUSIONES Y TRABAJO FUTURO

1.12.1.- CONCLUSIONES

En el transcurso del presente Proyecto se ha conseguido resolver una serie de problemas de control avanzado.

Se ha realizado la conversión de un modelo matemático complejo, un Benchmark basado en una EDAR, en un algoritmo discretizado con una fidelidad correcta. También se ha logrado obtener un modelo simplificado lineal y uno no lineal con una fiabilidad aceptable, pero se ha demostrado la dificultad de linealizar un modelo con combinaciones no lineales de saturaciones internas y ganancias dependientes de otras entradas.

Por otro lado, se ha procedido a utilizar técnicas óptimas de identificación de patrones para obtener el caudal base, así como a diseñar un observador de fallos para identificar la presencia de lluvias. También se han usado filtros autorregresivos para actualizar dicho patrón mediante caudales generados de manera aleatoria.

Por último, se ha conseguido diseñar el algoritmo del control predictivo minimizando el coste de la energía y manteniendo la salida amónica bajo límites. Los resultados han sido satisfactorios, y aún pueden mejorarse aumentando la cantidad de datos tratados; para ello, se requeriría un ordenador con mayor potencia de cálculo.

1.12.2.- TRABAJO FUTURO

El presente Proyecto ha asentado la base para el desarrollo de futuros proyectos de investigación sobre control avanzado en EDARs y otras estaciones hídricas. Las posibilidades de ampliación de este trabajo en el futuro son las siguientes:

- Ampliar el algoritmo de control, o reescribir uno en paralelo siguiendo las técnicas diseñadas en este Proyecto, para poder utilizar otros Active Sludge Models (ASM), mediante la modificación de los modelos actuales y manteniendo las mismas premisas utilizadas en el diseño del actual control predictivo. Estos otros ASM incluyen factores como la temperatura o la eliminación de fósforo.

- Aplicar en el modelo actual (o en modificaciones futuras) el efecto de las dinámicas de los sensores. Desarrollar un algoritmo que permita la detección de fallos de funcionamiento de estos sensores. De esta manera, su aplicación industrial mejoraría al implementar técnicas de control preventivo.

- Mejorar el método de control predictivo aplicando varios horizontes de predicción, y varios periodos distintos de muestreo. Para ello, se necesita un desarrollo posterior del algoritmo actual, así como una mayor capacidad y velocidad de cálculo.

- Realizar las actualizaciones de caudal y del modelo lineal de la EDAR en tiempo real, incluyéndolos dentro del bucle cerrado de control. De esta manera, la precisión del control mejoraría notablemente. Mejorar el modelo lineal utilizando concentraciones variables.

- Cambiar el método de simulación a Simulink. El aumento de la velocidad de cálculo necesaria se puede conseguir mediante técnicas de compresión de la Level-2 Matlab S-Function.

2.- ANEXOS

2.1.- INTRODUCCIÓN A LAS EDAR

Una Estación Depuradora de Aguas Residuales, o EDAR, es una instalación industrial cuyo objetivo es tratar las aguas que proceden del alcantarillado, eliminando los contaminantes que contienen, de manera que puedan ser vertidas de nuevo al medio receptor minimizando el impacto ecológico que supone. Esta minimización se regula mediante límites tanto de concentraciones de productos químicos como de temperatura en el momento del vertido.

El tratamiento de aguas residuales puede agruparse en cinco etapas distintas en función de los tipos de proceso que se realizan y del objetivo de cada proceso. Las etapas son las siguientes:

Pretratamiento

El pretratamiento consiste en la realización de una serie de tratamientos físicos destinados a separar los contaminantes suficientemente grandes cuya simple presencia en el sistema de depuración pueda afectar o dañar a los componentes que realizan los tratamientos (tuberías, bombas, etc.).

Los pretratamientos que se realizan son los siguientes:

- **Desbaste o cribado**. Este es el primer tratamiento que se realiza en la EDAR. El agua residual pasa por una reja de gruesos. Esta reja consiste en una serie de barras con una separación entre las mismas que impide el paso a sólidos grandes que no son solubles, como por ejemplo latas, bolsas, ropa, ramas, etc. Todos estos residuos son retirados manual o automáticamente, y son posteriormente tratados como residuos sólidos urbanos (RSU).

- **Desarenado**. Este tratamiento consiste en reducir la velocidad de las aguas residuales de forma que las arenas, así como otros sólidos lo suficientemente grandes (como semillas, café o cáscaras) que hayan podido atravesar la reja de gruesos acaben precipitando y sedimentándose. Esta reducción de la velocidad de las aguas se consigue mediante el uso de embalses de gran tamaño. La eliminación de las arenas y componentes similares es imprescindible, debido a que la acumulación de estos materiales podría acabar erosionando las bombas y acumulándose en las tuberías.

- **Desengrasado**. De manera similar al desarenado, al reducirse la velocidad de las aguas residuales, todos aquellos componentes más ligeros que el agua, como aceites, grasas, y otros lípidos se elevan a la superficie y pueden ser retirados para su posterior tratamiento y saponificación. Estas grasas suelen eliminarse automáticamente mediante máquinas separadoras de aceites.

Tratamiento primario

Una vez pretratadas las aguas residuales, éstas pasan al tratamiento primario, consistente también en una serie de procesos físicos. En este caso, las aguas se introducen en el decantador primario, que es un embalse de grandes dimensiones donde la velocidad de las aguas residuales se reduce mucho. De esta manera, las partículas flotantes de reducido tamaño que no habían podido ser eliminadas en el desarenado precipitan en este tratamiento, y las burbujas de grasa suficientemente pequeñas que no habían sido eliminadas en el desengrasado flotan a la superficie.

Como los procesos de eliminación de contaminantes del pretratamiento y del tratamiento primario son tan similares, a veces se consideran parte del mismo conjunto de tratamientos.

Tratamiento secundario

En la siguiente etapa de la EDAR, las aguas residuales pasan a una serie de reactores que realizan tratamientos biológicos, donde unos microorganismos eliminan la materia orgánica que contienen, y generan una serie de fangos que son retirados en el decantador secundario. Hay diferentes tipos de reactores, pero los que se tratarán en el presente trabajo serán los reactores de fangos activados.

Los cálculos realizados en el presente trabajo se centran exclusivamente en el tratamiento secundario, por lo que más adelante se proporcionará su modelo matemático.

Tratamiento terciario

Aunque tras el tratamiento secundario el agua ya puede ser apta para ser vertida al medio receptor, normalmente se realizan unos tratamientos posteriores para mejorar la calidad del agua. Algunos de estos tratamientos son la cloración o la aplicación de rayos UV para eliminar posibles gérmenes que permanezcan en el agua, la adición o eliminación de ciertos nutrientes, o la adición de oxígeno disuelto (para evitar efectos contraproducentes en el medio donde se vierte el agua), etc.

Todos estos tratamientos posteriores suponen un coste adicional que puede ser reducido si el tratamiento secundario se realiza de manera correcta.

Línea de fangos

Durante el proceso de depuración se genera una cantidad de fangos que deben ser tratados. Estos fangos pueden considerarse como residuos sólidos urbanos y llevarse a vertederos o ser incinerados. También pueden ser utilizados como abono para usos agrícolas o la generación de biogás. En cualquier caso, necesitan ser tratados mediante un espesamiento, digestión y deshidratación previos, además ser transportados a su lugar de destino.

2.2.- DATOS PROCEDENTES DEL BSM1

En este anexo se incluye un resumen de la información que proporciona el Benchmark BSM1, es decir, todas las ecuaciones, esquemas y relaciones que conforman dicho modelo.

***Nota:** Se han traducido los elementos principales con los que se trabajará en el control. El resto se mantienen en inglés, para mayor fidelidad con respecto al documento original.

2.2.1.- COMPORTAMIENTO DINÁMICO Y CONTROL DE EDAR: PROBLEMAS

Los procesos químicos y biológicos que conforman el tratamiento secundario de una EDAR son muy diversos y complejos, y muchos de estos procesos están muy interrelacionados, de manera que un pequeño cambio en una variable altera prácticamente todo el sistema. Por tanto, es factible afirmar que las EDAR son sistemas muy no lineales, hecho que dificulta su control.

Además, las EDAR están sometidas a elevadas perturbaciones, debido a que sus condiciones dependen del caudal de entrada de las aguas residuales procedentes del alcantarillado, así como de la concentración de los componentes de entrada. Ninguno de estos componentes es constante

a lo largo del tiempo, y aunque es posible realizar una aceptable predicción del flujo de entrada, debido al carácter periódico de los patrones de generación de aguas residuales, la variabilidad de la entrada es lo suficientemente elevada como para que no se pueda dar un modelo de la perturbación muy preciso, por lo que el control debe ser más robusto.

También es importante tener en cuenta las restricciones en la salida. La normativa medioambiental va haciéndose cada vez más estricta en términos de la calidad de los vertidos de las depuradoras, por lo que el control de los procesos cada vez debe ser más preciso. Teniendo también en cuenta los factores de no-linealidad y variabilidad de las perturbaciones, puede observarse que el control de las EDAR no es simple.

Otro punto que dificulta el control de una EDAR es la variedad de las constantes de tiempo de cada uno de los procesos, ya que algunos se estabilizan en cuestión de minutos, y otros pueden llegar a tardar días en estabilizarse.

Por último, se puede observar un último problema a la hora de obtener modelos y cálculos que sean lo suficientemente genéricos para cualquier EDAR, ya que cada lugar tiene sus propias restricciones de vertidos, distintos patrones de generación de aguas residuales, cada EDAR tiene unas dimensiones o componentes distintos, …

2.2.2.- SOBRE EL MODELO DEL BSM1

Para intentar solucionar el último problema, se hace uso de un benchmark. Un benchmark es un caso de estudio, o de referencia, que presenta un modelo simplificado de un problema, y una propuesta de resolución de dicho problema. Este trabajo parte del BSM1 (Benchmark Simulation Model no. 1), que presenta un modelo matemático de una depuradora para su simulación, y una propuesta de control simple de la misma. Este modelo es lo suficientemente genérico como para poder realizar posteriormente una adaptación a una depuradora real.

El BSM1 fue desarrollado por el *Dept. of Industrial Electrical Engineering and Automation* de la Universidad de Lund, en Suecia. La versión de la que se ha partido en el presente trabajo es la de 2008, actualizada posteriormente en 2018.

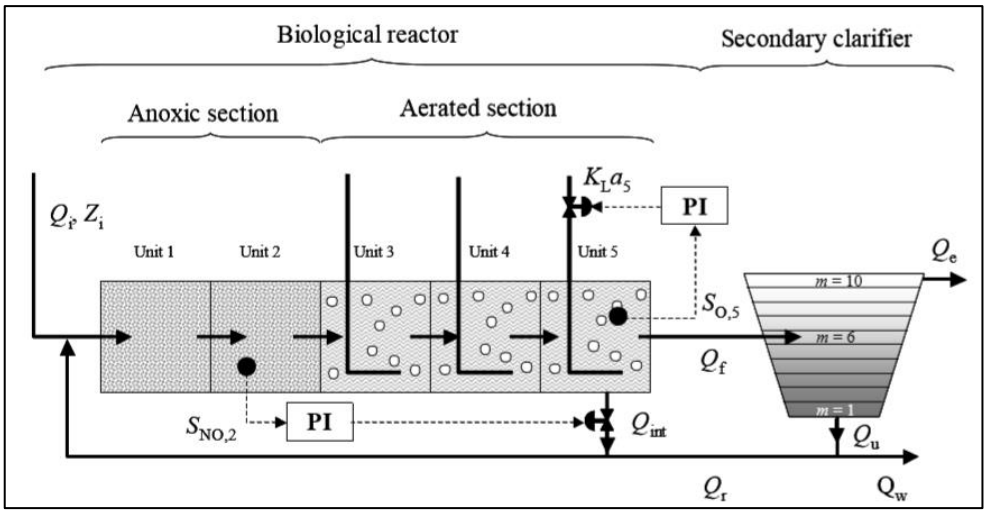

Imagen 40.- Componentes de una EDAR modelados por el BSM1

2.2.2.1.- CARACTERÍSTICAS DEL MODELO DEL BENCHMARK

El sistema de depuración detallado en el BSM1 comprende cinco compartimentos de un reactor biológico interconectados en serie y un decantador secundario de diez capas (*clarifier*), además de dos recirculaciones del agua residual. Los distintos caudales (Q) y concentraciones (Z) que interconectan los elementos del sistema se detallan en la siguiente tabla.

Tabla 1.- Denominación de los caudales del sistema

SIGLAS	NOMBRE	EXPLICACIÓN
Q_0, Z_0	FLUJO DE ENTRADA	El flujo de entrada procede del alcantarillado, y junto a las recirculaciones entra en el compartimento 1.
Q_f, Z_f	FLUJO DE ALIMENTACIÓN AL DECANTADOR (FEED FLOW)	Parte del flujo del compartimento 5 pasa a alimentar el decantador.
Q_a, Z_a	RECIRCULACIÓN INTERNA	La otra parte del flujo que sale del compartimento 5 se recircula al compartimento 1.
Q_e, Z_e	EFLUENTE	El agua que sale de la parte superior del decantador se vierte al río.
Q_u, Z_u	FLUJO INFERIOR (UNDERFLOW)	Flujo que sale por la parte inferior del decantador.
Q_r, Z_r	RECIRCULACIÓN EXTERNA	Parte del flujo inferior se recircula de vuelta al compartimento 1.
Q_w, Z_w	FLUJO DE PURGA (WASTAGE)	Fango que se retira de la recirculación para evitar que se acumule en exceso al recircular el flujo inferior.

Según el modelo básico del BSM1, las variables de control de la depuradora son el $K_L a$ del compartimento 5 (parámetro que considera tanto la cantidad de oxígeno como la facilidad de que el agua residual lo absorba) y la válvula de recirculación interna.

Por tanto, el modelo matemático del BSM1 incluye ecuaciones tanto para el reactor biológico (los cinco compartimentos comparten un modelo interno muy similar) y para el decantador secundario, así como unas ecuaciones para determinar la edad del fango.

2.2.2.2.- REACTOR BIOLÓGICO

El reactor biológico se modeliza utilizando las ecuaciones del ASM1 (Activated Sludge Model no.1, Modelo de Fangos Activos número 1), que describen los procesos biológicos que se producen dentro del reactor biológico.

Las relaciones entre los distintos componentes se pueden observar en la imagen 41.

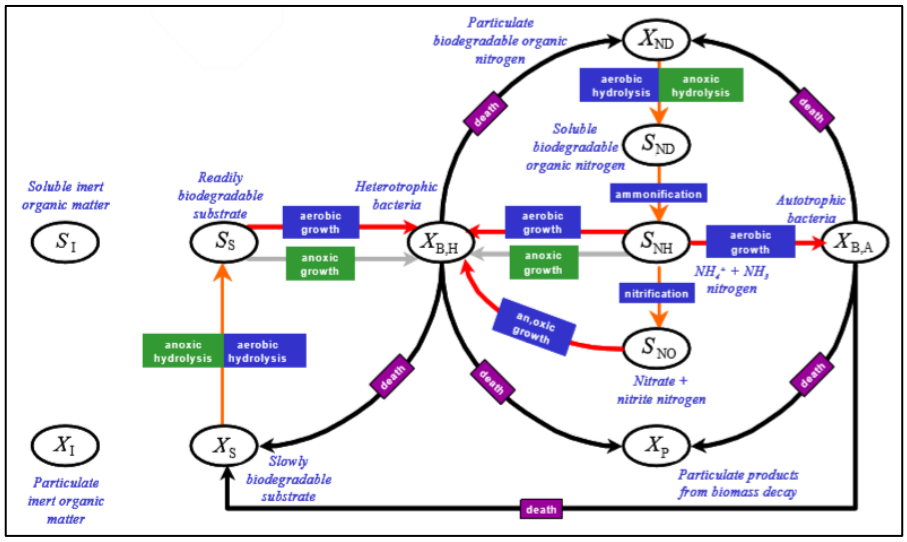

Imagen 41.- Relación entre las distintas variables en el ASM1

Todas estas relaciones entre variables se traducen en una serie de ecuaciones, que se explicarán en los siguientes apartados.

Listado de variables

En el bloque del reactor biológico se utilizan las 13 variables de estado que caracterizan al modelo ASM1 correspondientes a la composición del agua residual. Representan la concentración de distintos componentes en el agua residual que se está tratando. Además de estas variables de estado, también se utiliza el valor del caudal circulante.

Tabla 2.- Variables del ASM1

NOMBRE	NOTACIÓN	UNIDADES
SOLUBLE INERT ORGANIC MATTER	S_I	$g\ COD \cdot m^{-3}$
READILY BIODEGRADABLE SUBSTRATE	S_S	$g\ COD \cdot m^{-3}$
PARTICULATE INERT ORGANIC MATTER	X_I	$g\ COD \cdot m^{-3}$
SLOWLY BIODEGRADABLE SUBSTRATE	X_S	$g\ COD \cdot m^{-3}$
ACTIVE HETEROTROPHIC BIOMASS	$X_{B,H}$	$g\ COD \cdot m^{-3}$
ACTIVE AUTOTROPHIC BIOMASS	$X_{B,A}$	$g\ COD \cdot m^{-3}$
PARTICULATE PRODUCTS ARISING FROM BIOMASS DECAY	X_P	$g\ COD \cdot m^{-3}$
OXYGEN	S_O	$g\ (-COD) \cdot m^{-3}$
NITRATE AND NITRITE NITROGEN	S_{NO}	$g\ N \cdot m^{-3}$
$NH_4^+ + NH_3$ NITROGEN	S_{NH}	$g\ N \cdot m^{-3}$
SOLUBLE BIODEGRADABLE ORGANIC NITROGEN	S_{ND}	$g\ N \cdot m^{-3}$
PARTICULATE BIODEGRADABLE ORGANIC NITROGEN	X_{ND}	$g\ N \cdot m^{-3}$
ALKALINITY	S_{ALK}	$mol \cdot m^{-3}$
FLOW RATE	Q	$m^3 \cdot d^{-1}$

Puede observarse que las variables que representan componentes solubles comienzan por la letra S, y las variables que representan componentes particulados comienzan por la letra X. Esta diferenciación es muy importante, especialmente a la hora de realizar los cálculos con el decantador.

Nota: S_{NH} es la variable que se denomina *amonio* en la memoria, controlada en el efluente.

Hay ocho procesos básicos que definen el comportamiento biológico de cada reactor. Estos se muestran a continuación:

j = 1: Aerobic growth of heterotrophs

$$\rho_1 = \mu_H \left(\frac{S_S}{K_S + S_S}\right)\left(\frac{S_O}{K_{O,H} + S_O}\right) X_{B,H} \quad \textbf{(2.1)}$$

j = 2: Anoxic growth of heterotrophs

$$\rho_2 = \mu_H \left(\frac{S_S}{K_S + S_S}\right)\left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\eta_g X_{B,H} \quad \textbf{(2.2)}$$

j = 3: Aerobic growth of autotrophs

$$\rho_3 = \mu_A \left(\frac{S_{NH}}{K_{NH} + S_{NH}}\right)\left(\frac{S_O}{K_{O,A} + S_O}\right) X_{B,A} \quad \textbf{(2.3)}$$

j = 4: Decay of heterotrophs

$$\rho_4 = b_H X_{B,H} \quad \textbf{(2.4)}$$

j = 5: Decay of autotrophs

$$\rho_5 = b_A X_{B,A} \quad \textbf{(2.5)}$$

j = 6: Ammonification or soluble organic nitrogen

$$\rho_6 = k_a S_{ND} X_{B,H} \quad \textbf{(2.6)}$$

j = 7: Hydrolysis of entrapped organics

$$\rho_7 = \frac{X_S/X_{B,H}}{K_X + (X_S/X_{B,H})}\left[\left(\frac{S_O}{K_{O,H} + S_O}\right) + \eta_H \left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\right] X_{B,H} \quad \textbf{(2.7)}$$

j = 8: Hydrolysis of entrapped organic nitrogen

$$\rho_8 = \frac{X_S/X_{B,H}}{K_X + (X_S/X_{B,H})}\left[\left(\frac{S_O}{K_{O,H} + S_O}\right) + \eta_H \left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\right] X_{B,H}\left(\frac{X_{ND}}{X_S}\right) \quad \textbf{(2.8)}$$

Las ecuaciones que definen los procesos 1, 2, 3, 7 y 8 se denominan funciones *switch*. Esto se debe a que contienen fracciones cuyos términos son una constante (denominada como K) y una variable del sistema, y uno de los dos términos se repite tanto en el numerador como en el denominador. Por tanto, en el caso de que la variable sea el término que se repite, si es mucho más elevada que la constante, la fracción queda acotada a 1, y si se reduce hasta quedar muy por debajo de la constante, la fracción queda acotada a 0. En el caso contrario, en que la constante sea el término que se repite, si la variable crece mucho la fracción queda acotada a 0, y si se reduce hasta ser muy inferior a la constante, la fracción queda acotada a 1. Estas funciones son fuente de no linealidades que dificultan el control de la EDAR.

El resultado de las funciones que definen los procesos son unos valores denominados ρ. El siguiente paso es calcular las ratios de variación de los componentes debido a los procesos biológicos. Las ratios son combinaciones lineales de los valores ρ, y se listan a continuación.

$\boldsymbol{S_I\ (i\ =\ 1)}$

$$r_1 = 0 \tag{2.9}$$

$\boldsymbol{S_S\ (i\ =\ 2)}$

$$r_2 = -\frac{1}{Y_H}\rho_1 - \frac{1}{Y_H}\rho_2 + \rho_7 \tag{2.10}$$

$\boldsymbol{X_I\ (i\ =\ 3)}$

$$r_3 = 0 \tag{2.11}$$

$\boldsymbol{X_S\ (i = 4)}$

$$r_4 = (1 - f_P)\rho_4 + (1 - f_P)\rho_5 - \rho_7 \tag{2.12}$$

$\boldsymbol{X_{B,H}\ (i\ =\ 5)}$

$$r_5 = \rho_1 + \rho_2 - \rho_4 \tag{2.13}$$

$\boldsymbol{X_{B,A}\ (i\ =\ 6)}$

$$r_6 = \rho_3 - \rho_5 \tag{2.14}$$

$\boldsymbol{X_P\ (i\ =\ 7)}$

$$r_7 = f_P\rho_4 + f_P\rho_5 \tag{2.15}$$

$\boldsymbol{S_O\ (i\ =\ 8)}$

$$r_8 = -\frac{1 - Y_H}{Y_H}\rho_1 - \frac{4.57 - Y_A}{Y_A}\rho_3 \tag{2.16}$$

$\boldsymbol{S_{NO}\ (i\ =\ 9)}$

$$r_9 = -\frac{1 - Y_H}{2.86\ Y_H}\rho_2 + \frac{1}{Y_A}\rho_3 \tag{2.17}$$

$\boldsymbol{S_{NH}\ (i\ =10)}$

$$r_{10} = -i_{XB}\rho_1 - i_{XB}\rho_2 - \left(i_{XB} + \frac{1}{Y_A}\right)\rho_3 + \rho_6 \tag{2.18}$$

$\boldsymbol{S_{ND}\ (i\ =\ 11)}$

$$r_{11} = -\rho_6 + \rho_8 \tag{2.19}$$

$\boldsymbol{X_{ND}\ (i\ =\ 12)}$

$$r_{12} = (i_{XB} - f_P i_{XP})\rho_4 + (i_{XB} - f_P i_{XP})\rho_5 - \rho_8 \tag{2.20}$$

$\boldsymbol{S_{ALK}\ (i\ =\ 13)}$

$$r_{13} = -\frac{i_{XB}}{14}\rho_1 + \left(\frac{1 - Y_H}{14 \cdot 2.86\ Y_H} - \frac{i_{XB}}{14}\right)\rho_2 - \left(\frac{i_{XB}}{14} + \frac{1}{7Y_A}\right)\rho_3 + \frac{1}{14}\rho_6 \tag{2.21}$$

Las constantes utilizadas en las ecuaciones del reactor biológico se muestran a continuación. Para el cálculo de las funciones de los procesos biológicos se utilizan los parámetros cinéticos, y para el cálculo de las ratios, los parámetros estequiométricos.

Tabla 3.- Parámetros estequiométricos

PARÁMETRO	UNIDAD	VALOR
Y_A	$g\ cell\ COD\ formed.(g\ N\ oxidized)^{-1}$	0.24
Y_H	$g\ cell\ COD\ formed.(g\ COD\ oxidized)^{-1}$	0.67
f_P	$dimensionless$	0.08
i_{XB}	$g\ N.(g\ COD)^{-1}\ in\ biomass$	0.08
i_{XP}	$g\ N.(g\ COD)^{-1}\ in particulate\ products$	0.06

Tabla 4.- Parámetros cinéticos

PARÁMETRO	UNIDAD	VALOR
μ_H	d^{-1}	4.0
K_S	$g\ COD.m^{-3}$	10.0
$K_{O,H}$	$g\ (-COD).m^{-3}$	0.2
K_{NO}	$gNO_3 - N.m^{-3}$	0.5
b_H	d^{-1}	0.3
η_g	$Sin\ dimensiones$	0.8
η_h	$Sin\ dimensiones$	0.8
k_h	$g\ slowly\ biodegradable\ COD.(g\ cell\ COD.d)^{-1}$	3.0
K_X	$g\ slowly\ biodegradable\ COD.(g\ cell\ COD)^{-1}$	0.1
μ_A	d^{-1}	0.5
K_{NH}	$g\ NH_3 - N.m^{-3}$	1.0
b_A	d^{-1}	0.05
$K_{O,A}$	$g\ (-COD).m^{-3}$	0.4
k_a	$m^3.(g\ COD.d)^{-1}$	0.05

Características del reactor biológico

El modelo del BSM1 utiliza la siguiente configuración:

N.º de compartimentos del reactor: 5

Compartimentos anóxicos: compartimentos 1 y 2

Compartimentos aerobios:

- *Compartimentos 3 y 4*. Se les aplica un coeficiente de transferencia de oxígeno constante ($K_La = 10\ h^{-1} = 240\ d^{-1}$)

- *Compartimento* 5: El K_La puede manipularse libremente. Su punto estable es $K_La = 84\ d^{-1}$.

El volumen de cada compartimento está relacionado con el tipo de proceso que ocurre en su interior:

- **Compartimentos anóxicos**: $V_{as,1} = V_{as,2} = 1000\ m^3$

- **Compartimentos aeróbicos**: $V_{as,3} = V_{as,4} = V_{as,5} = 1333\ m^3$

Cada compartimento se denomina con el subíndice k en las ecuaciones, por lo que el caudal es Q_k, el vector de concentraciones es Z_k y el vector de ratios es r_k.

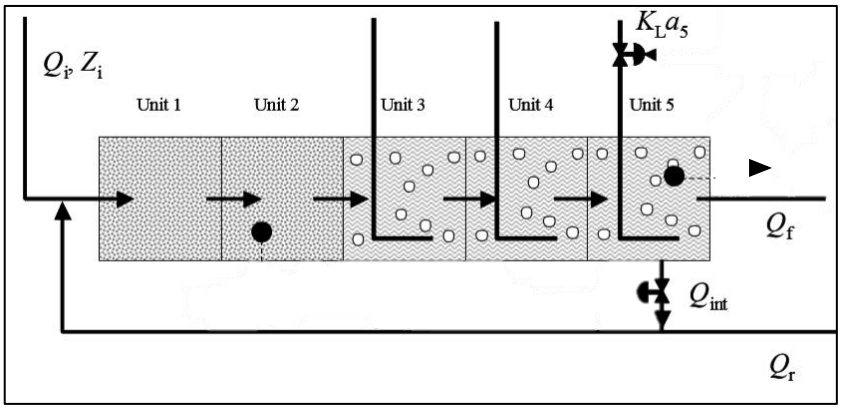

Imagen 42.- Reactor biológico, según el BSM1

Balances de masa

Las ecuaciones para los balances de masa se muestran a continuación:

Para el primer compartimento ($k = 1$):

$$\frac{dZ_{as,1}}{dt} = \frac{1}{V_{as,1}}\left(Q_{int}Z_{int} + Q_r Z_r + Q_i Z_i + r_{Z,1}V_{as,1} - Q_1 Z_{as,1}\right) \quad \textbf{(2.22)}$$

$$Q_1 = Q_{int} + Q_r + Q_i \quad \textbf{(2.23)}$$

Para el resto (k del 2 al 5):

$$\frac{dZ_{as,k}}{dt} = \frac{1}{V_{as,k}}\left(Q_{k-1}Z_{as,k-1} + r_{Z,k}V_{as,k} - Q_k Z_{as,k}\right) \quad \textbf{(2.24)}$$

$$Q_k = Q_{k-1} \quad \textbf{(2.25)}$$

Excepto para el **oxígeno**, que en todos los casos se calcula así:

$$\frac{dS_{O,as,k}}{dt} = \frac{1}{V_{as,k}}\left(Q_{k-1}S_{O,as,k-1} + r_{Z,k}V_{as,k} + (K_L a)_k V_{as,k}(S_O^* - S_{O,as,k}) + Q_k S_{O,as,k}\right) \quad \textbf{(2.26)}$$

Donde la concentración de saturación para el oxígeno es $S_0^* = 8\, g \cdot m^{-3}$

Con estas ecuaciones se realiza el cálculo de la derivada del vector Z, que contiene todas las variables de estado del compartimento (concentraciones), en función del tiempo. Por tanto, la derivada del vector de estados Z de un compartimento depende del caudal y del vector Z de entrada, del caudal y del vector Z de salida (cuyos valores de concentración son los mismos que los del compartimento), de la ratio r de variación de los componentes, y del volumen del compartimento. La excepción es el caso de k = 1, porque el caudal y las concentraciones de entrada son las de las recirculaciones y las procedentes del alcantarillado.

Este cálculo se aplica de la misma manera para todos los componentes salvo para el oxígeno, que utiliza una ecuación de variación propia que depende de un coeficiente de transferencia líquido-gas procedente de la adición de oxígeno (en función de si se trata de un compartimento aerobio), y de un parámetro de saturación de la concentración de oxígeno.

Relación entre los caudales y concentraciones

$$Z_{int} = Z_{as,5} \quad \textbf{(2.27)}$$

$$Z_f = Z_{as,5} \quad \textbf{(2.28)}$$

$$Z_w = Z_r \quad \textbf{(2.29)}$$

$$Q_f = Q_e + Q_r + Q_w = Q_e + Q_u \quad \textbf{(2.30)}$$

La concentración de la recirculación interna y de la alimentación al decantador es la misma que la del compartimento 5, pues es de donde proceden ambos caudales de agua residual, y no hay ninguna reacción intermedia. La concentración de la recirculación externa es la misma que la de la corriente de purga, debido a que proceden de una división del flujo inferior (*underflow*). En la última ecuación se muestran las divisiones de caudal: el caudal de salida del compartimento 5 se divide en el de recirculación interna y el de alimentación del decantador, y éste último se separa en el caudal efluente, de recirculación externa y de purga.

2.2.2.3.- DECANTADOR SECUNDARIO

Características

El modelo que ofrece el BSM1 es de un decantador de 10 capas, en el cual no se produce ningún tipo de reacción biológica. El caudal de alimentación entra al decantador en la capa número 6, el efluente sale de la capa superior, que es la nº10, y el caudal inferior sale de la capa nº1.

Las dimensiones del decantador son:

- **Área**: 1500 m^2

- **Altura de las capas**: 0'4 m

- **Volumen total del decantador**: 6000 m^3

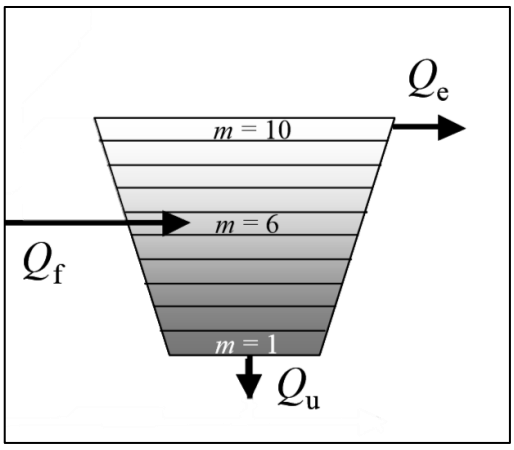

Imagen 43.- Decantador secundario, según el BSM1

En el caso que ofrece el BSM1, se modeliza el comportamiento de cada una de las 10 capas (denominadas m) conectadas en serie que conforman el decantador. Los procesos que suceden en el interior son físicos (no se producen reacciones químicas), y las ratios de variación de las concentraciones dependen de si se trata de un componente soluble (S) o particulado (X).

Componentes solubles

Los componentes solubles se calculan de la siguiente manera:

- Para la capa de alimentación (m = 6):

$$\frac{dZ_{sc,m}}{dt} = \frac{\frac{Q_f Z_f}{A} - (v_{dn} + v_{up})Z_{sc,m}}{z_m} \quad \textbf{(2.31)}$$

- Para las capas inferiores (1 a 5):

$$\frac{dZ_{sc,m}}{dt} = \frac{v_{dn}(Z_{sc,m+1} - Z_{sc,m})}{z_m} \quad \textbf{(2.32)}$$

- Para las capas superiores (7 a 10):

$$\frac{dZ_{sc,m}}{dt} = \frac{v_{up}(Z_{sc,m-1} - Z_{sc,m})}{z_m} \quad \textbf{(2.33)}$$

Las velocidades de circulación del caudal se calculan de la siguiente manera:

$$v_{dn} = \frac{Q_u}{A} = \frac{Q_r + Q_w}{A} \quad \textbf{(2.34)}$$

$$v_{up} = \frac{Q_e}{A} \quad \textbf{(2.35)}$$

Se puede entender que v_{dn} es proporcional al caudal inferior, y v_{up} al caudal del efluente. Ambos caudales son variables libres, pues no dependen de ninguna otra ecuación, y deben regularse de manera externa.

De esta forma, la ratio de variación de las concentraciones de los componentes solubles en las capas inferiores depende de la altura de la capa, las concentraciones en la propia capa, las concentraciones en la capa superior y de la velocidad de descenso del agua. Esta ecuación representa que las concentraciones varían simplemente con el movimiento de descenso del agua.

De manera similar, en las capas superiores las ratios dependen de la altura de la capa, de la velocidad de subida del agua, así como de la concentración de la propia capa y de la inferior, por lo que las variaciones dependen del movimiento de ascenso del agua.

Por último, en la capa de alimentación, las ratios de variación de las concentraciones dependen del flujo de alimentación, de la altura de la capa, de la concentración actual de la capa, y de tanto el caudal de subida como el de bajada, porque ésta es la única capa en la que el agua acaba yendo en dos direcciones (subida y bajada).

Componentes particulados

En el decantador secundario, el comportamiento de los componentes particulados se calcula agrupándolos en una sola variable, pues se considera que van a reaccionar de la misma manera al tratamiento físico, al flujo del agua residual y a la gravedad.

Por tanto, se realiza una conversión a un estado particulado total, y todos los cálculos posteriores dentro del decantador se realizan en función de este estado. La conversión realizada consiste en la suma de las concentraciones de los componentes realizados, con un posterior cambio de unidades de COD (demanda química de oxígeno) a SS (sólidos suspendidos). El factor de conversión fr_{COD-SS} es de 4/3.

$$X_f = \frac{1}{fr_{COD-SS}}\left(X_{S,as,5} + X_{P,as,5} + X_{I,as,5} + X_{B,H,as,5} + X_{B,A,as,5}\right) \quad \textbf{(2.36)}$$
$$= 0.75\left(X_{S,as,5} + X_{P,as,5} + X_{I,as,5} + X_{B,H,as,5} + X_{B,A,as,5}\right)$$

Las concentraciones utilizadas para el cálculo de este nuevo estado son las de la salida del último compartimento del reactor biológico (compartimento 5).

Además, se calculan las proporciones de cada concentración de componente particulado respecto a la concentración del total de particulados, porque se realiza la simplificación de que esta proporción se mantiene a lo largo de todo el decantador, y en ambas salidas de este (flujo inferior y efluente). Por tanto, para obtener los valores de todos los componentes en las salidas del decantador, debe realizarse el proceso inverso.

$$\frac{X_{S,as,5}}{X_f} = \frac{X_{S,sc,1}}{X_u} \quad \textbf{(2.37)}$$

Sin embargo, hay que tener en cuenta que esta simplificación comporta que los componentes particulados no presenten un comportamiento dinámico dentro del decantador, y que una variación súbita de uno de los componentes a la entrada del decantador se transmitirá de manera inmediata a las salidas, a diferencia de los componentes solubles y el particulado total, que sí presentan un comportamiento dinámico en el interior del decantador.

Para calcular las dinámicas del flujo de sólidos es necesario calcular el flujo de las partículas debido al efecto de la gravedad:

$$J_S = v_s(X_{sc})X_{sc} \quad \textbf{(2.38)}$$

Donde X_{sc} es la concentración de particulados. La v_s se calcula mediante la ecuación siguiente:

$$v_s(X_{sc}) = max\left[0, min\left\{v_0', v_0\left(e^{-r_h(X_{sc}-X_{min})} - e^{-r_p(X_{sc}-X_{min})}\right)\right\}\right] \quad \textbf{(2.39)}$$

En el que $X_{min} = f_{ns}X_f$. La ecuación de la v_s representa que la velocidad del flujo de partículas depende de unos parámetros (r_p y r_h) que determinan las zonas en un decantador en función de la densidad de las partículas.

Tabla 5.- Parámetros de sedimentación

NOMBRE	PARÁM.	UNIDADES	VALOR
MAXIMUM SETTLING VELOCITY	v'_0	$m.d^{-1}$	250
MAXIMUM VESILIND SETTLING VELOCITY	v_0	$m.d^{-1}$	474
HINDERED ZONE SETTLING PARAMETER	r_h	$m^3.(gSS)^{-1}$	0.000576
FLOCCULANT ZONE SETTLING PARAMETER	r_p	$m^3.(g\,SS)^{-1}$	0.00286
NON-SETTLEABLE FRACTION	f_{ns}	$Sin\ dimensiones$	0.00228

Por otro lado, se encuentra el flujo J_{sc}, que se calcula como:

$$J_{sc,j} = \begin{cases} \min\left(v_{s,j}X_{sc,j}, v_{s,j-1}X_{sc,j-1}\right) \ \ if\ X_{sc,j-1} > X_t \\ v_{s,j}X_{sc,j}\ if\ X_{sc,j-1} \leq X_t \end{cases} \quad \textbf{(2.40)}$$

Este flujo está sujeto a los límites marcados por los umbrales de concentración. Para la capa 10 se tiene que:

$$J_{sc,10} = \begin{cases} \min\left(v_{s,10}X_{sc,10}, v_{s,9}X_{sc,9}\right) \ \ if\ X_{sc,9} > X_t \\ v_{s,10}X_{sc,10}\ if\ X_{sc,9} \leq X_t \end{cases} \quad \textbf{(2.41)}$$

Donde $X_t = 3000\ g \cdot m^{-3}$

Las ecuaciones que rigen el comportamiento de la variable de partículas en cada capa son, por tanto:

- Para la capa de alimentación ($m = 6$):

$$\frac{dX_{sc,m}}{dt} = \frac{\frac{Q_f X_f}{A} + J_{sc,m+1} - \left(v_{up} + v_{dn}\right)X_{sc,m} - min\left(J_{s,m}, J_{s,m-1}\right)}{z_m} \quad \textbf{(2.42)}$$

- Para las capas bajo la de alimentación (de $m = 5$ hasta $m = 2$):

$$\frac{dX_{sc,m}}{dt} = \frac{v_{dn}\left(X_{sc,m+1} - X_{sc,m}\right) + \min\left(J_{s,m}, J_{s,m+1}\right) - \min\left(J_{s,m}, J_{s,m-1}\right)}{z_m} \quad \textbf{(2.43)}$$

- Para la capa inferior ($m = 1$):

$$\frac{dX_{sc,1}}{dt} = \frac{v_{dn}(X_{sc,2} - X_{sc,1}) + \min\left(J_{s,2}, J_{s,1}\right)}{z_1} \quad \textbf{(2.44)}$$

- Para las capas sobre la de alimentación (de $m = 7$ hasta $m = 9$):

$$\frac{dX_{sc,m}}{dt} = \frac{v_{up}\left(X_{sc,m-1} - X_{sc,m}\right) + J_{sc,m+1} - J_{sc,m}}{z_m} \quad \textbf{(2.45)}$$

- Para la capa superior ($m = 10$):

$$\frac{dX_{sc,10}}{dt} = \frac{v_{up}\left(X_{sc,9} - X_{sc,10}\right) - J_{sc,10}}{z_{10}} \quad \textbf{(2.46)}$$

En el caso del componente particulado total, deben calcularse las distintas velocidades del flujo de sólidos causadas por la gravedad. Estas velocidades dependen principalmente de las concentraciones particuladas totales de cada capa. En función de si la capa donde se realizan los cálculos se encuentra sobre o bajo la capa de alimentación, las velocidades también dependen de la concentración de la capa inferior.

La variación de la concentración del componente particulado total en cada unidad de tiempo depende tanto de la diferencia de concentraciones entre la capa actual y la anterior en el sentido del flujo de aguas residuales, del flujo de caída por gravedad de las partículas en la capa, así como del de flujo de caída por gravedad desde las capas superiores, así como de los valores del caudal de entrada en la capa de alimentación.

2.2.2.4.- GENERACIÓN DE FANGOS

La cantidad de fangos que se purga a la salida del decantador se calcula de la siguiente manera:

$$\Psi_w(g/día) = TSS_{sc,1} \cdot Q_w \quad \textbf{(2.47)}$$

Donde la cantidad de componentes particulados en la capa 1 del decantador es:

$$TSS_{sc,1}\left(\frac{g\ SS}{m^3}\right) = \frac{1}{fr_{COD-SS}}\left(X_{S,sc,1} + X_{P,sc,1} + X_{I,sc,1} + X_{B,H,sc,1} + X_{B,A,sc,1}\right) \quad \textbf{(2.48)}$$

2.2.3.- FICHEROS DEL CAUDAL AFLUENTE A LA EDAR

El caudal afluente a la EDAR que utiliza el BSM1 se puede descargar desde la web del IWA (ver Apartado 1.4.4.). Los datos del afluente se encuentran en tres ficheros separados, denominados respectivamente Inf_dry_2006.txt, Inf_rain_2006.txt y Inf_strm_2006.txt.

2.2.3.1.- ESTRUCTURA DE LOS FICHEROS

Los ficheros son documentos de texto que contienen los datos del afluente ordenados en una matriz de 14 columnas. Los datos de cada columna son, en orden, el tiempo, las trece variables del modelo ASM1 para el reactor biológico y el caudal entrante, todo en las unidades especificadas en el BSM1. Los datos de cada columna están separados por tabuladores.

Cada fila es una muestra distinta del caudal afluente. El periodo de tiempo entre muestras es constante, de 0,10417 días, o 15 minutos. En total hay 1345 muestras, que equivalen a 14 días.

2.2.3.2.- COMPARACIÓN ENTRE LOS FICHEROS

El fichero Inf_dry_2006.txt muestra el comportamiento del caudal afluente a la depuradora durante el transcurso de dos semanas sin precipitaciones. Sin embargo, si se analizan los datos, se puede observar que los datos de la segunda semana son prácticamente idénticos a los de la primera semana.

El fichero Inf_rain_2006.txt representa el comportamiento del caudal afluente durante dos semanas en las que ha habido un par de días en los que se han registrado precipitaciones continuas. No obstante, si se analizan los datos, se puede observar que este fichero es idéntico al anterior, solo que entre la muestra 804 y la 1002 (entre el día 8 a las 8:30 y el 10 a las 10:30) se le ha sumado un valor constante de 20000 m^3 al caudal, y las concentraciones de los componentes cuyo valor anterior era distinto de cero disminuyen su valor durante ese periodo.

El fichero Inf_strm_2006.txt contiene los datos del comportamiento del caudal afluente durante dos semanas en los que ha habido precipitaciones muy intensas durante periodos de tiempo muy reducidos. Analizando los datos, se descubre que es también idéntico al primer fichero, pero en este caso se le ha añadido un valor de 40000-45000 m^3 de caudal durante dos periodos de tiempo: uno entre la muestra 846 y la 858 (el día 8 entre las 19:15 y las 22:30), y otro entre las muestras 1050 y 1110 (entre el día 10 a las 22:00 y el día 11 a las 13:15). Las concentraciones, si previamente eran distintas de cero, durante ese periodo varían su valor.

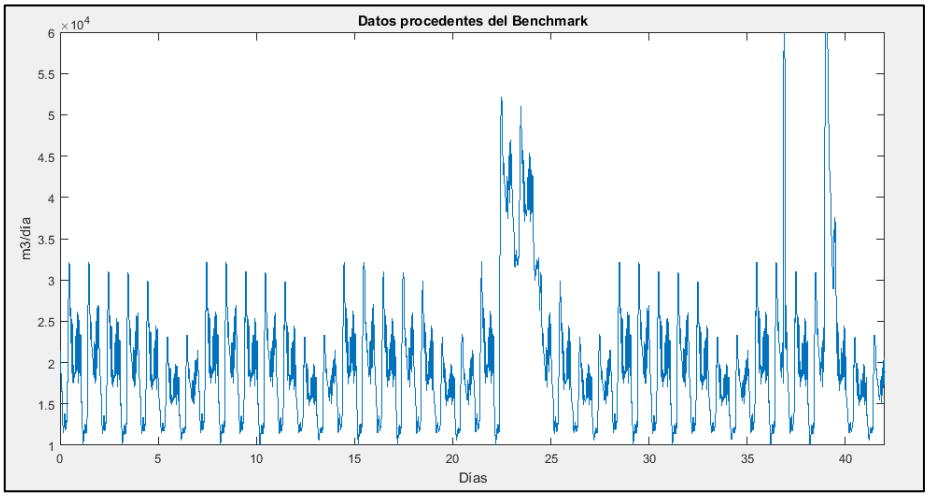

Imagen 44.- Caudales afluentes, según el BSM1

2.3.- CONTROL PREDICTIVO BASADO EN MODELO (MPC)

2.3.1.- ¿EN QUÉ CONSISTE EL MPC?

El control predictivo basado en modelo es un algoritmo de control que utiliza una serie de estimaciones y aproximaciones para predecir el comportamiento futuro del sistema que se está controlando, y actuar al respecto. Para ello, utiliza un modelo simplificado del sistema que controla, así como una estimación de las perturbaciones que le van a afectar. También requiere los valores de referencias que se aplicarán en el futuro.

A diferencia del control por PID, en el MPC las acciones de control no se obtienen mediante la aplicación de una serie de ecuaciones, sino que son el resultado de la optimización de una función de coste. Esta optimización suele incorporar restricciones, tanto en las entradas como en las salidas, o incluso en las ratios de variación de las entradas.

La estrategia que se utiliza en el control predictivo se conoce como estrategia deslizante, en la que, aunque durante la optimización se calculan todos los potenciales valores de la acción de control durante todo el horizonte previsto, solo se utiliza el primer valor, y se repite toda la optimización en el siguiente periodo.

El MPC tiene la ventaja de poder ser multivariable, permite la minimización de costes, tiene en cuenta las limitaciones de los actuadores y permite el ahorro de técnicas adicionales para tener en cuenta retardos grandes. Sin embargo, requiere un modelo apropiado del sistema a controlar, y requiere una elevada carga computacional.

La función objetivo a minimizar más genérica tiene la siguiente forma:

$$J = \sum_{j=N_1}^{N_2} \delta(j)\,[\hat{y}(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{N_u} \lambda(j)\,[\Delta u(t+j-1)]^2 \qquad \textbf{(2.49)}$$

Donde N_1 y N_2 conforman la ventana de predicción (el periodo de tiempo en el que se controla la diferencia entre la salida y la referencia a seguir para minimizar el error cometido); $\delta(j)$ y $\lambda(j)$ son factores de ponderación, y $N_u$es el horizonte de control (el periodo de tiempo en el que se minimiza la variación de la salida Δu.

Por supuesto, hay muchas variaciones de esta función de coste. En el presente Proyecto, se da peso a minimizar la acción de control (en valor absoluto, no a minimizar la variación del actuador).

2.3.2.- ¿QUÉ INFORMACIÓN NECESITA EL ALGORITMO DEL MPC?

Para poder diseñar y aplicar un correcto control predictivo, es imprescindible disponer de la siguiente información:

- **Un modelo simplificado del sistema.** De esta forma, el control puede simular el comportamiento del sistema de manera interna, y predecir las salidas en función de las entradas propuestas.

- **Una aceptable previsión de las perturbaciones.** Con esta previsión, el control conoce las entradas al sistema que no puede controlar e intenta anticiparse a ellas.

- **Los valores de las referencias a seguir en el futuro**. Si dispone de esta información, el control puede prever los cambios requeridos desde el exterior, y aplicar las acciones de control oportunas con antelación. Es especialmente importante si el modelo tiene retardos importantes. En este Proyecto, debido al diseño de la función de coste del optimizador, esta información no se ha requerido.

En los siguientes apartados se procederá a explicar la metodología empleada para resolver el problema de la obtención de dicha información en este Proyecto.

2.4.- OBTENCIÓN DEL MODELO SIMPLIFICADO DEL SISTEMA.

El objetivo principal de obtener un modelo simplificado de un sistema complejo es conseguir una aproximación lineal del comportamiento no lineal de dicho sistema. Hay muchas formas de conseguir esta aproximación; en estos anexos se explicarán los distintos métodos utilizados en este Proyecto.

2.4.1.- CÁLCULO POR REGRESORES

2.4.1.1.- DEFINICIÓN MATEMÁTICA

El objetivo de calcular un modelo por regresores es obtener la serie de parámetros que multiplican a la ecuación lineal simplificada que modela el comportamiento de un sistema. Esta ecuación puede depender de un número limitado de entradas y salidas anteriores al sistema, o exclusivamente de un número limitado de entradas anteriores al sistema.

En el primer caso, se está buscando calcular una ecuación autorregresiva:

$$y_k^1 = a_1 \cdot y_{k-1}^1 + a_2 \cdot y_{k-2}^1 + \cdots + a_N \cdot y_{k-N}^1 + b_1 \cdot u_{k-1}^1 + \cdots + b_N \cdot u_{k-N}^1 + c_1 \cdot u_{k-1}^2 + \cdots + c_N \cdot u_{k-N}^2 \quad \textbf{(2.50)}$$

En el segundo caso, la ecuación es no autorregresiva:

$$y_k^1 = b_1 \cdot u_{k-1}^1 + \cdots + b_N \cdot u_{k-N}^1 + c_1 \cdot u_{k-1}^2 + c_2 \cdot u_{k-N}^2 + \cdots \quad \textbf{(2.51)}$$

En este caso, los superíndices indican el número de salida o entrada del sistema (u_{k-1}^2 indica que es la entrada 2 en el instante $k-1$). N es el orden del regresor, es decir, la cantidad de instantes de tiempo del pasado que se toman para predecir el siguiente instante.

Por tanto, el objetivo es obtener es un vector que contiene los parámetros $[a_1, \dots a_N, b_1, \dots b_N, c_1, \dots c_N, \dots]$. En algunos casos, también es posible añadir un término independiente a_0 sumado a la ecuación lineal.

2.4.1.2.- OBTENCIÓN DE LOS DATOS EXPERIMENTALES

Para poder calcular el vector mediante un regresor, primero es necesario realizar un experimento inicial consistente en la simulación del sistema, de manera que se puedan obtener datos aptos. Para ello, se debe llevar al sistema a un punto de equilibrio (en este Proyecto, este punto viene proporcionado por el Benchmark), y aplicar variaciones a las entradas, de manera que la salida varíe consecuentemente. Todos estos valores deben almacenarse apropiadamente, de manera ordenada según los instantes de tiempo en los que se hayan producido.

En el caso de sistemas no lineales como éste, es recomendable aplicar un vector de entradas aleatorias que excite el sistema de manera muy frecuente, consiguiendo así que la salida no llegue a estabilizarse en ningún momento. Con estos datos puede llegar a obtenerse un vector de parámetros con alta precisión. El código para la generación de este vector se encuentra en el Apéndice A.

2.4.1.3.- ESTRUCTURA PARA EL CÁLCULO MEDIANTE REGRESORES

Para obtener el vector de parámetros, hay que montar una serie de matrices en orden correcto, de forma que consigan los parámetros en el orden apropiado.

Para este ejemplo, se ha considerado una ecuación lineal autorregresiva con dos entradas. Dados los vectores:

$$\begin{aligned} u^1 &= \{u_0^1, u_1^1, u_2^1, \dots u_N^1\} \\ u^2 &= \{u_0^2, u_1^2, u_2^2, \dots u_N^2\} \\ y^1 &= \{y_0, y_1, y_2, \dots y_N\} \end{aligned} \quad \textbf{(2.52)}$$

Donde N es el total de elementos calculados y n el orden del regresor, la matriz de salidas se calcula como:

$$Y = \begin{bmatrix} y_n^1 \\ y_{n+1}^1 \\ y_{n+2}^1 \\ \vdots \\ y_N^1 \end{bmatrix} \quad \textbf{(2.53)}$$

Por otro lado, la matriz de regresores X, que contiene todas las entradas en instantes anteriores, tiene la siguiente forma:

$$X = \begin{bmatrix} y_{n-1}^1 & \cdots & y_0^1 & u_{n-1}^1 & \cdots & u_0^1 & u_{n-1}^2 & \cdots & u_0^2 \\ y_n^1 & \cdots & y_1^1 & u_n^1 & \cdots & u_1^1 & u_n^2 & \cdots & u_1^2 \\ y_{n+1}^1 & \cdots & y_2^1 & u_{n+1}^1 & \cdots & u_2^1 & u_{n+1}^2 & \cdots & u_2^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{N-1}^1 & \cdots & y_{N-1-(n-1)}^1 & u_{N-1}^1 & \cdots & u_{N-1-(n-1)}^1 & u_{N-1}^2 & \cdots & u_{N-1-(n-1)}^2 \end{bmatrix} \quad \textbf{(2.54)}$$

El objetivo de este cálculo es encontrar cuál es el vector de parámetros que minimice el error entre el vector Y y su estimación $\hat{Y} = X \cdot \theta$.

Para ello se puede usar la función de Matlab `mldivide`, o bien realizar una minimización de la resta $\hat{Y} - Y$, de manera que los parámetros a calcular sean los del vector θ. Ambas opciones realizan los mismos cálculos, solo que el segundo método es más editable, ya que pueden añadirse restricciones adicionales.

2.4.1.4.- VARIACIONES DEL CÁLCULO

Hasta ahora solo se han mostrado dos maneras de rellenar el regresor: con las entradas y las salidas del sistema a modelar (autorregresivo) o con solo las entradas al sistema (no autorregresivo). No obstante, hay más opciones para calcular el modelo, combinables con los métodos anteriores. Sin embargo, el modelo pasa a ser una simplificación no-lineal:

- Utilizar los valores cuadráticos de las entradas. Este método, basado en modelos de Volterra, consiste en añadir términos cuadráticos de las entradas a la ecuación modelo, que queda de la siguiente forma:

$$y_k^1 = a_1 \cdot y_{k-1}^1 + \cdots + a_N \cdot y_{k-N}^1 + b_1 \cdot u_{k-1}^1 + \cdots + b_N \cdot u_{k-N}^1 + c_1 \cdot u_{k-1}^2 + \cdots + c_N \cdot u_{k-N}^2 + \\ \ldots + d_1 \cdot (u_{k-1}^1)^2 + \cdots + d_N \cdot (u_{k-N}^1)^2 + e_1 \cdot (u_{k-1}^2)^2 + \cdots + e_N \cdot (u_{k-N}^2)^2 + \cdots \quad \textbf{(2.55)}$$

También existe la opción de incluir el producto cruzado de entradas en el cálculo con regresores. Estos métodos son muy útiles cuando el efecto de una entrada depende del estado de otra entrada.

- Usar la raíz de las salidas. El cálculo se realiza con la raíz cuadrada de los valores de la salida del sistema. Esta opción es útil cuando las salidas presentan un comportamiento con picos bruscos, de manera que el cálculo de su raíz cuadrada modera esa dinámica.

La ecuación que se quiere obtener es, por tanto:

$$\sqrt{y_k^1} = a_1 \cdot \sqrt{y_{k-1}^1} + a_2 \cdot \sqrt{y_{k-2}^1} + \cdots + a_N \cdot \sqrt{y_{k-N}^1} + b_1 \cdot u_{k-1}^1 + \cdots + b_N \\ \cdot u_{k-N}^1 + c_1 \cdot u_{k-1}^2 + \cdots + c_N \cdot u_{k-N}^2 \quad \textbf{(2.56)}$$

Esta opción parece funcionar bastante bien para identificar el comportamiento de la salida de amonio S_{NH} frente al $K_L a$ y el SRI debido a que presenta el comportamiento mencionado.

2.4.1.5.- CONSEJOS PARA EL CÁLCULO

- Minimización de la norma. El resultado del cálculo mediante regresores no siempre es capaz de determinar correctamente todas las dinámicas del sistema, especialmente si éste ha sido excitado con el vector de entradas aleatorias comentado en el 2.4.1.2. A veces es incapaz de identificar apropiadamente un retardo, y los parámetros obtenidos provocan que el modelo simplificado sea algo oscilatorio. Para evitar estos problemas, se ha propuesto minimizar, además de la diferencia $\hat{Y} - Y$, la varianza entre los valores del vector de parámetros θ, ya que la dispersión entre los parámetros provoca dicho comportamiento aleatorio no deseado. La minimización queda, por tanto:

$$min\left(P_1 \cdot (Y - X \cdot \Theta)^2 + P_2 \cdot \frac{1}{M} \sum_{j=1}^{M} (\Theta(j) - \mu_\Theta)^2 \right) \quad \textbf{(2.57)}$$

Donde se añaden pesos P_1, P_2delante de cada opción a minimizar para regular su impacto en el cálculo. Si $P_2 = 0$ y $P_1 \neq 0$, se dispone del regresor base.

- Frecuencias de muestreo. La simulación de las ecuaciones se realiza con una frecuencia de cálculo de 1 minuto, de manera que el resultado de las ecuaciones sea lo más preciso posible. No obstante, el comportamiento dinámico del sistema funciona con tiempos mucho más elevados, del orden de horas o incluso días. Debido a que el orden del regresor debe ser capaz de abarcar esos periodos de tiempo para poder obtener un modelo simplificado fiable, utilizar directamente las muestras obtenidas de la simulación implicaría calcular regresores con órdenes superiores a 1000, requiriendo potencias y tiempos de cálculo totalmente inasumibles. Por tanto, la recomendación es aplicar un submuestreo (*decimation*, en inglés), de manera que se tome una de cada 15, 30 o 60 muestras (cada cuarto, media y hora entera).

- Normalización e incrementalización de las variables entrantes. Una opción válida para el cálculo mediante regresores es normalizar (dividir entre el valor máximo) e incrementar (restar el valor de equilibrio) los vectores de entrada y salida. Con esta opción, no se añade el valor adicional a_0 al regresor.

2.4.1.6.- APLICACIÓN EN EL ALGORITMO

Para el modelado de la dinámica del comportamiento de la salida de amonio S_{NH}, se ha decidido utilizar un método autorregresivo, dado que proporcionaba mejores resultados que el no autorregresivo. También se ha decidido aplicar la minimización de la norma, debido a que la dinámica presenta ciertos retardos. Se ha decidido utilizar valores absolutos, por lo que no se ha aplicado ninguna normalización ni incrementalización de las variables, y se ha requerido la inclusión del parámetro a_0. La frecuencia de muestreo seleccionada ha sido de 60 minutos, que, aunque no es tan precisa como la de 15, permite órdenes del regresor mucho mayores (mejorando el funcionamiento del control predictivo). Por último, se ha probado la opción de realizar los cálculos con la entrada normal y con la raíz de la entrada.

El algoritmo está diseñado para su posterior edición, de forma que modificando unas ciertas variables se pueda recalcular el modelo con otra frecuencia de submuestreo, con normalización e incrementalización, y de manera no autorregresiva.

Debido a que el código para el cálculo mediante regresores abarca una gran cantidad de líneas de código, no se presenta ninguna muestra aquí. Todo el código se encuentra en el Apéndice A.3.

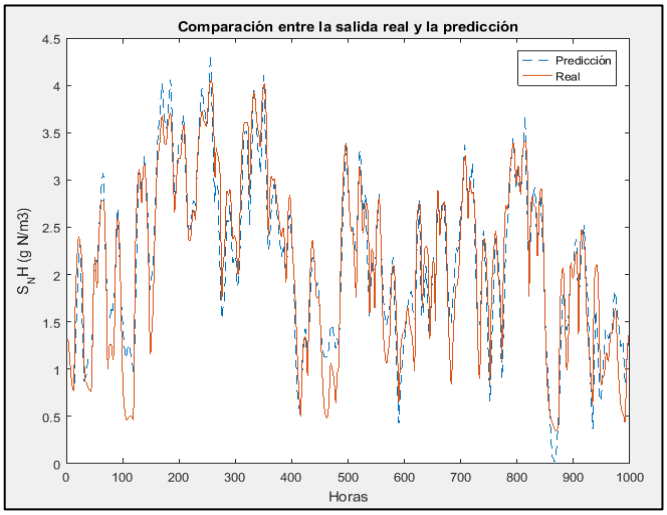

Imagen 45.- Aplicación del modelo de regresores autorregresivos

2.4.2.- IDENTIFICACIÓN POR FUNCIONES DE TRANSFERENCIA

2.4.2.1.- DEFINICIÓN MATEMÁTICA.

El uso de funciones de transferencia es el método más utilizado para identificar el comportamiento dinámico de un sistema. Consiste en obtener un modelo lineal que relaciona una entrada con una salida, expresado como un cociente entre ambas señales, utilizando transformadas de Laplace.

La función de transferencia típica tiene la siguiente forma:

$$G(s) = \frac{k \cdot (1 \pm \beta s) \cdot e^{-Ts}}{s \cdot (1 + \tau_1 s) \cdot (1 + \tau_2 s) \cdot (1 + 2\xi\omega_n^{-1} s + \omega_n^{-2} s^2)} \quad \textbf{(2.58)}$$

Donde k es la ganancia del sistema, T el tiempo de retardo, τ_n son función del tiempo de establecimiento de la señal, ξ y ω_n determinan el comportamiento oscilatorio del sistema, β indica la presencia de sobreoscilaciones, y la existencia de s en el denominador representa que la dinámica es integral.

El proceso de identificación por funciones de transferencia requiere tres partes: la obtención experimental de datos, la identificación del comportamiento mediante el uso de herramientas informáticas y la conversión a tiempo discreto.

2.4.2.2.- REALIZACIÓN DEL EXPERIMENTO

Para conseguir los datos para la identificación, debe efectuarse un experimento consistente en la simulación del sistema procedente de las ecuaciones del Benchmark. A este sistema se le deben aplicar entradas escalón para posteriormente comprobar el comportamiento de las salidas.

El primer paso consiste en encontrar un punto estable del sistema. Este punto viene proporcionado por el BSM1, de forma que se aplican dichas entradas al sistema simulado. Una vez esté estable, se aplica una entrada escalón a una sola entrada, manteniendo el resto en su punto estable. Se repiten los pasos de estabilización y de efectuar un escalón con cada una de las entradas que se quieren calcular. Por último, se almacenan los datos para su posterior análisis.

2.4.2.3.- IDENTIFICACIÓN DEL SISTEMA

El siguiente paso es analizar los datos para obtener las funciones de transferencia. Para ello pueden utilizarse herramientas como identescalon, de las Freepidtools (ver Anexo 2.9.1.- Identificación mediante respuesta ante escalón, donde se encuentra un manual de uso). Matlab también dispone de herramientas para la identificación de sistemas, como la System Identification Tool (comando `ident`).

2.4.2.4.- DISCRETIZACIÓN DEL SISTEMA

Como el entorno de programación seleccionado es Matlab y todo el control está discretizado, las funciones de transferencia continuas deben convertirse al tiempo discreto. Matlab dispone de la función `c2d`, que permite dicha conversión. Solo es necesario especificar el tiempo de muestreo en las unidades apropiadas.

Esta nueva función de transferencia tendrá la siguiente forma:

$$H(z) = \frac{\beta_0 + z^{-1}\beta_1 + z^{-2}\beta_2 + \cdots + z^{-M}\beta_M}{\alpha_0 + z^{-1}\alpha_1 + z^{-2}\alpha_2 + \cdots + z^{-N}\alpha_N} \quad \textbf{(2.59)}$$

Si se desea convertir la función de transferencia en un vector de parámetros, utilizados en modelos IIR autorregresivos (como los comentados en el apartado de regresores), simplemente deben ordenarse los parámetros α_i (que son los parámetros que multiplicarán a $y_{k-1}, y_{k-2}, \ldots$) y β_i (que son los parámetros que multiplicarán a $u_{k-1}, u_{k-2}, \ldots$).

2.5.- PREPARACIÓN DEL CAUDAL AFLUENTE.

2.5.1.- TRATAMIENTO DE LOS DATOS DEL CAUDAL AFLUENTE

La cantidad de días simulados en el experimento durante las pruebas puede llegar a ser de varios meses. Por tanto, existe la posibilidad de que los datos proporcionados por el Benchmark, o los caudales generados, no sean suficientes. Por tanto, se deben encadenar distintos vectores de caudales, asignando correctamente los nuevos vectores temporales.

Además, para poder utilizar los datos en una simulación más precisa, es imprescindible realizar una interpolación de los datos, de manera que se obtengan los valores del caudal cada minuto en vez de cada 15 minutos. Hay dos maneras posibles de interpolar:

- Interpolación de primer orden. Esta opción consiste en calcular los valores intermedios trazando una línea recta entre los dos valores de los que parte la interpolación. La función de Matlab `interp1` permite realizar esta interpolación.

- Interpolación por splines. Esta opción consiste en calcular los valores intermedios trazando curvas de manera que el total de valores interpolados genere una curva continua y derivable en todos sus puntos. La función de Matlab `spline` permite realizar esta interpolación.

El código para realizar tanto el encadenamiento de datos de caudales, así como su interpolación, se encuentra en el Apéndice A.7.4.

2.5.2.- SIMPLIFICACIÓN DEL CAUDAL AFLUENTE (CASO DEL DEPÓSITO INICIAL)

Una de las simplificaciones consideradas en la memoria es la de que las concentraciones de los componentes del caudal afluente no varían con el tiempo, a diferencia del caudal entrante, que sí lo hace. Esto se podría conseguir con un depósito a la entrada del sistema a controlar, el cual contase con una capacidad volumétrica muy elevada.

En este apartado se demostrará la validez de dicha simplificación.

La ecuación que modeliza el comportamiento de una concentración dentro de un depósito, teniendo en cuenta que en el mismo no se produce ninguna reacción física, ni de decantación, es la siguiente:

$$V \cdot \dot{c}_{dep} = q_i \cdot c_i - q_o \cdot c_o \tag{2.60}$$

Es decir, que la variación de la masa del componente concentrado dentro del depósito depende de cuánta masa entre de dicho componente menos la cantidad de masa de dicho componente que sale.

Considerando que se trata de un depósito constantemente lleno, en el que el caudal volumétrico entrante es idéntico al saliente, se considera que

$$q_i = q_o = q \tag{2.61}$$

Además, la concentración de salida será la misma que la del depósito, considerando que el caudal de entrada se ha mezclado de manera homogénea:

$$c_{dep} = c_o = c \tag{2.62}$$

Por tanto, la ecuación queda de la siguiente manera:

$$V \cdot \dot{c} = q \cdot c_i - q \cdot c$$

Agrupando c en un lado:

$$V \cdot \dot{c} + q \cdot c = q \cdot c_i \tag{2.63}$$

Aplicando Laplace:

$$V \cdot C(s) \cdot s + Q(s) \cdot C(s) = Q(s) \cdot C_i(s) \tag{2.64}$$

Reorganizando los términos queda que:

$$C(s) \cdot \big(V \cdot s + Q(s)\big) = Q(s) \cdot C_i(s) \tag{2.65}$$

$$C(s) = \frac{Q(s)}{V \cdot s + Q(s)} \cdot C_i(s) \tag{2.66}$$

Dividiendo ambas partes entre $Q(s)$, queda que:

$$C(s) = \frac{1}{1 + \frac{V}{Q(s)} s} \cdot C_i(s) \tag{2.67}$$

Por lo que el depósito queda simulado como una función de primer orden, en el que el tiempo de establecimiento es proporcional a la constante de tiempo τ.

$$\tau = \frac{V}{Q(s)} \tag{2.68}$$

De manera que, si el volumen del depósito es muy superior al valor del caudal entrante en todo momento, la constante de tiempo es muy elevada, y por tanto, la concentración de salida del depósito se mantiene prácticamente constante a lo largo del tiempo, ya que las variaciones en la entrada no afectan al sistema.

Esta consideración es asumible, dado que en los procesos de pretratamiento y del tratamiento primario las aguas residuales se almacenan en diversos embalses de grandes dimensiones para eliminar grasas y arenas en suspensión.

2.6.- PREDICCIÓN DE LAS PERTURBACIONES. ANÁLISIS DEL COMPORTAMIENTO DEL CAUDAL AFLUENTE

Para poder aplicar un control predictivo de manera correcta se debe tener de antemano todos los valores futuros de referencias y de perturbaciones. En el caso de este sistema, la principal perturbación que se muestra es la variación del caudal y de las concentraciones de entrada a la depuradora, procedente del alcantarillado.

La gráfica de la imagen 46 muestra el comportamiento del caudal y las concentraciones del afluente a la EDAR, proporcionados por el Benchmark. Analizando la gráfica, pueden observarse distintos patrones y peculiaridades.

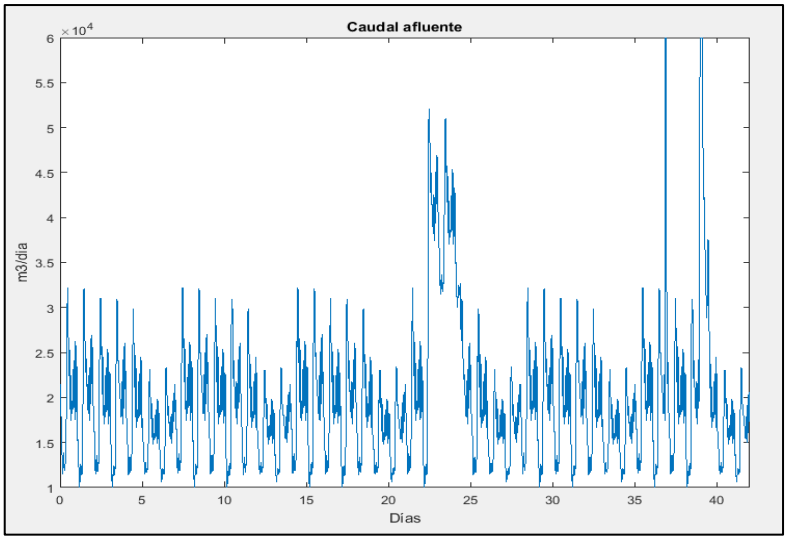

Imagen 46.- Caudal afluente a la EDAR

Como puede observarse, presenta un comportamiento bastante periódico. Esto se debe a que la generación de agua residual sigue patrones similares día tras día, dado que la población y la industria suele seguir la misma rutina de consumo hídrico.

También puede observarse otro **patrón semanal**, debido a que en fin de semana los consumos son inferiores debido a que hay menos industrias generando aguas residuales.

En la gráfica se puede observar que hay momentos en los que el caudal de entrada es notablemente más elevado que otros. Esto se debe a las **precipitaciones**, pues la lluvia es arrastrada a las alcantarillas. Pueden identificarse dos tipos distintos de precipitaciones. En el caso de que se observen los mismos comportamientos de la llegada de aguas residuales, pero en valores superiores a los habituales, se trata de unas precipitaciones **moderadas**, repartidas durante un amplio periodo de tiempo. Sin embargo, si lo que se observa es un pico muy elevado que disrumpe el patrón de caudal típico, se trata de precipitaciones causadas por una **tormenta**, es decir, elevado caudal en un intervalo de tiempo reducido. En ambos casos, además del caudal también se ven afectadas las concentraciones, pues pueden aumentar (si la lluvia ha arrastrado una gran cantidad de materia orgánica a su paso) o disminuir (debido a la dilución de estos componentes en una mayor cantidad de agua).

2.6.1.- DETECCIÓN Y PREDICCIÓN DE COMPORTAMIENTOS ATÍPICOS DEL AFLUENTE

A la hora de predecir el comportamiento del caudal en el futuro, hay que tener en cuenta además la estacionalidad. En función de la época del año los consumos pueden acabar subiendo o bajando en función de distintos factores, tales como el clima (sequía, cortes de agua), o la cantidad de residentes en el municipio (este caso es más destacado en lugares turísticos).

Tanto los factores de precipitaciones como de estacionalidad son importantes tenerlos en cuenta para lograr una predicción adecuada de las oscilaciones del caudal hídrico. Como son patrones que no ocurren de manera regular (precipitaciones) o cuyo periodo de repetición es muy elevado y algo irregular (estacionalidad), se requieren cálculos adicionales que permitan la adaptación de los patrones base a estos cambios.

Por tanto, se puede considerar que la señal medida correspondiente al comportamiento del caudal (y de las concentraciones) de entrada se puede descomponer en las siguientes partes:

$$y_{señal}(t) = f_{t.base}(t) + \Delta y_{inc}(t) + fallo_{lluvia}(t) + v_{ruido}(t) \quad \textbf{(2.69)}$$

- **Patrón base ($f_{t.base}(t)$).** Se trata de una señal periódica que representa las variaciones diarias (y/o semanales) de la llegada de las aguas residuales a la depuradora. Debido a su carácter periódico, es la única parte de la señal que puede predecirse. Normalmente se le aplica una corrección utilizando los valores medidos, de manera que pueda actualizarse para adaptarse a los cambios producidos por la estacionalidad, o por si el patrón base no se ha inicializado correctamente.

- **Incertidumbre del patrón ($\Delta y_{inc}(t)$).** Esta señal representa la inexactitud entre el patrón base considerado y el comportamiento real del sistema (cuando no hay precipitaciones). Aunque la actualización del patrón base tiende a reducir el valor absoluto de esta señal, el valor de esta nunca llega a ser cero debido a la intrínseca aleatoriedad de la señal real.

- **Lluvias ($fallo_{lluvia}(t)$).** Esta señal representa las variaciones sobre el patrón base producidas por las precipitaciones. Para poder predecir los efectos de estas variaciones y que el control predictivo reaccione de manera eficaz, esta señal se considera como un *fallo* en el transcurso periódico de la señal principal, por lo que para su detección se utilizan métodos de análisis de fallos.

- **Ruido de medida ($v_{ruido}(t)$).** Esta señal aleatoria está implícita en cualquier instrumento de medida, y debe asumirse como parte del problema de control.

Para conseguir una buena predicción de las perturbaciones hace falta determinar las siguientes acciones:

- Obtener un patrón base adecuado. Debido a que es la única señal predecible, debe ser lo más fiable posible, para reducir el valor inicial de la incertidumbre del patrón.

- Actualizar el patrón base. La actualización del patrón con nuevos caudales produce que la incertidumbre del patrón adquiera un carácter más aleatorio, pero reducirá su valor absoluto medio causado por una incorrecta inicialización del patrón base.

- Detectar las lluvias. Expresadas como *fallos* de la señal, la precisión de su detección es inversamente proporcional al valor absoluto de la incertidumbre del patrón, así que la correcta resolución de los dos problemas anteriores es clave a la hora de resolver éste.

2.6.2.- OBTENCIÓN DEL PATRÓN BASE

Para predecir el comportamiento de una señal, el primer paso debe ser la determinación de los comportamientos cíclicos que presenta. Como hay distintos métodos disponibles para lograrlo, la selección del más adecuado dependerá tanto de la precisión y adecuación del método a los valores reales como de la cantidad de parámetros necesarios para el cálculo. La condición de precisión es la más relevante, debido a que las imprecisiones acabarán repercutiendo en la incertidumbre del patrón.

Los métodos considerados son los siguientes:

- Series de Fourier. Una serie de Fourier es una manera de componer una función periódica mediante un sumatorio infinito de funciones senoidales. La ecuación teórica es la siguiente:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cdot cos\left(\frac{2n\pi}{T} t\right) + b_n \cdot sin\left(\frac{2n\pi}{T} t\right) \right] \quad \textbf{(2.70)}$$

Donde T es el periodo de las funciones senoidales, y a_0, a_n y b_n son los coeficientes de Fourier

No obstante, para la obtención del patrón base se realiza una aproximación de la serie de Fourier, limitando el número de elementos del sumatorio para evitar un aumento innecesario de los parámetros requeridos (ver Apartado 1.7.6.- Análisis del caudal afluente y creación de patrones).

Para la aproximación también se valora la precisión de la aproximación respecto a los datos reales, teniendo en cuenta también el efecto del ruido. Es muy importante diferenciar el ruido de las oscilaciones de los patrones, y evitar órdenes muy elevados de la serie de Fourier con senoidales de alta frecuencia que intenten seguir dicho ruido. Este error suele ser muy común si el cálculo de la serie de Fourier es realizado mediante un programa de cálculo, especialmente si no se le ha indicado ninguna restricción de ruido.

Para calcular los valores de los parámetros de la serie de Fourier, debe realizarse un paso previo:

Transformada de Fourier

El método de la transformada de Fourier consiste en descomponer una señal en las distintas frecuencias que la constituyen. De esta forma, al aplicar dicha transformada a una señal, se obtienen tanto los valores de las frecuencias como la proporción de cada frecuencia.

Se ha aplicado la transformada de Fourier a la señal del caudal entrante a la depuradora. El espectro de frecuencias obtenido es el siguiente (unidad de tiempo *días*):

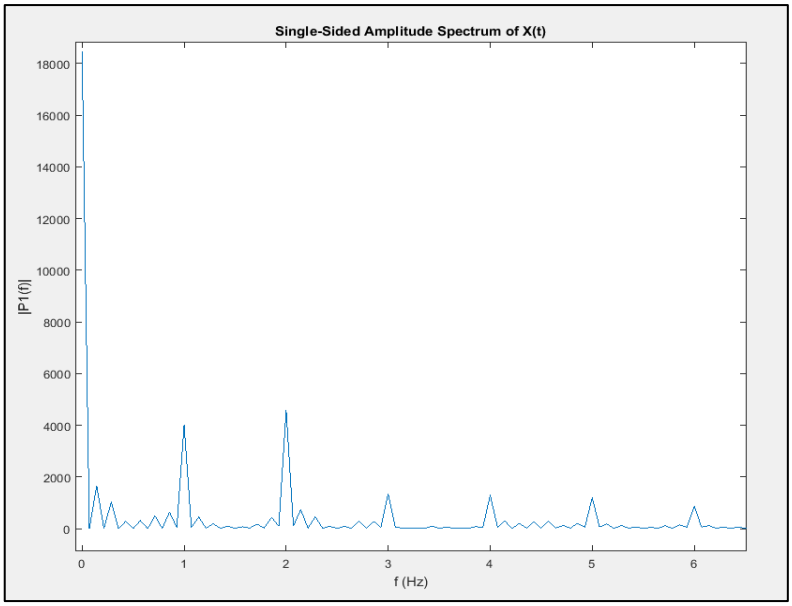

Imagen 47.- Espectro de frecuencias del patrón base del caudal afluente

Analizando la imagen 47 se pueden realizar distintas observaciones:

- El pico en 0 Hz indica el valor medio del caudal de entrada. Por tanto, se considera que el patrón de entrada consiste en distintas oscilaciones a distintas frecuencias a partir de este valor medio.

- La altura de los picos en las frecuencias de 1 a 6 Hz representan la forma y amplitud de las oscilaciones que se repiten formando un patrón. El pico en 1 Hz indica que el patrón principal es diario. El resto de los picos en números naturales son el resultado de la aproximación de la forma del patrón diario mediante series de Fourier. Por tanto, 1 Hz es la frecuencia fundamental de la señal periódica, y el resto de los picos son los armónicos superiores.

- También puede observarse un ligero pico en $0.1429 \approx \frac{1}{7}$. Esto se debe a que la señal presenta una cierta oscilación semanal (cinco días con más picos, dos días con menos). La aparición de ligeros picos en múltiplos de 1/7 indican la presencia de armónicos superiores.

- El resto de los valores obtenidos, cuyos máximos son muy inferiores a los picos mencionados anteriormente, representan el efecto del ruido. Puede observarse como la cantidad de ruido se extiende hasta valores de frecuencias bastante elevadas, características del ruido de medida.

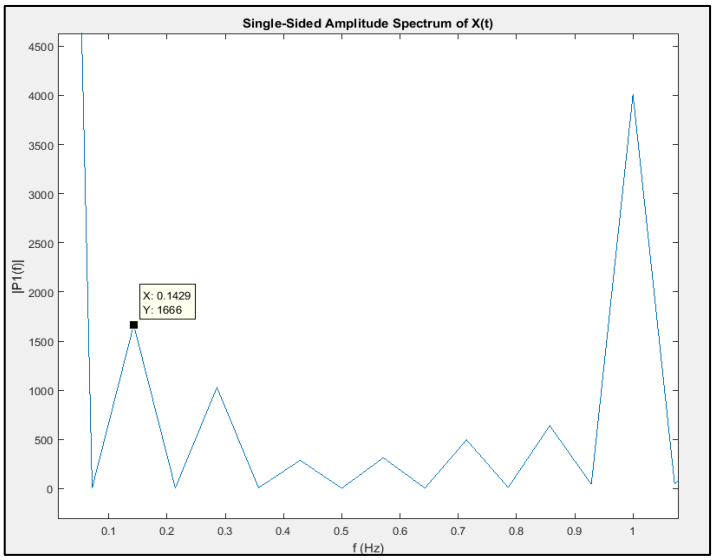

Imagen 48.- Espectro de frecuencias del patrón base del caudal afluente - Ampliación

Para obtener el espectro de frecuencias se ha utilizado el presente código, obtenido desde la web de Matlab:

```
load Datos_dry_2006.mat % Carga los datos
Ts=15/60/24; % en dias, periodo entre datos
N=length(Datosdry);
t=(0:N-1)'*Ts; % Así se crea el vector de tiempos
Y = fft(Datosdry(15,:)');
L=N-1; P2 = abs(Y/L); P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
Fs=1/(t(2)-t(1)); f = Fs*(0:(L/2))/L;
figure('Name','fft')
plot(f,P1); title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)'); ylabel('|P1(f)|')
```

Una vez obtenido el espectro de frecuencias de la señal e identificadas las frecuencias más relevantes, se procede a obtener una versión simplificada mediante una serie de Fourier, para posteriormente predecir el comportamiento del caudal afluente en el futuro.

Regresor de la serie de Fourier

Este método consiste en calcular directamente los parámetros que multiplican a las funciones sinusoidales que conforman la aproximación deseada de la serie de Fourier. Estos parámetros son equivalentes a los picos obtenidos mediante la transformada de Fourier (teniendo en cuenta las aproximaciones realizadas en el regresor).

El código utilizado es el siguiente:

```
load Datos_dry_2006.mat % Carga los datos
Ts=15/60/24; % en días, periodo entre datos
N=length(Datosdry);
t=(0:N-1)'*Ts;
Y = Datosdry(15,:)'; % Estos son los datos reales
w=2*pi/1; % Frecuencia fundamental de la oscilación diaria
wsem=2*pi/7; % Frec. fund. de la oscilación semanal (7 veces más lenta)
nf = 7; % Número de harmónicos considerados para la oscilación diaria
X = ones(N,1); % Valor medio(frecuencia 0)
% Obtención de la matriz X (tiempo x serie de Fourier)
```

```
for i=1:nf,  X=[X,sin(i*w*t), cos(i*w*t)]; end
nfsem = 6; % Número de harmónicos considerados para la oscilación semanal
% Ídem pero con la oscilación semanal
for i=1:nfsem,  X=[X,sin(i*wsem*t), cos(i*wsem*t)]; end
Theta=X\Y;
Yhat=X*Theta; % Yhat es distinto de Y por ser una aproximación.
```

El primer paso para realizar el cálculo con el regresor es montar una matriz con todas las funciones sinusoidales, resueltas para todos los instantes de tiempo del intervalo que se desea calcular. Esta matriz se denominará X (dimensiones: $nº\ de\ muestras\ x\ nº\ de\ parámetros$). Por otro lado, los valores reales de la función patrón se listan en un vector Y (dimensiones: $nº\ de\ muestras\ x\ 1$). En el código se ha considerado tener en cuenta hasta 7 harmónicos de la frecuencia diaria y 6 harmónicos de la frecuencia semanal, por lo que, teniendo en cuenta el parámetro de frecuencia cero, el número de parámetros es de 14.

El siguiente paso es usar `mldivide` (también sirve usar \) para obtener el vector de parámetros Θ, tal que $Y = X \cdot \Theta$. Aquí, Θ tiene dimensiones $(nº\ de\ parámetros\ x\ 1)$.

Hay que tener en cuenta que al aplicar `mldivide`, si el sistema está determinado, $\Theta = X \backslash Y$, y por tanto, $Y = X \cdot \Theta$. Sin embargo, si el sistema está sobredeterminado, en este caso lo que se obtiene con la operación `mldivide` es una Θ tal que al realizar la operación $\hat{Y} = X \cdot \Theta$, el error de $\hat{Y}$ respecto a Y sea mínimo. El valor de $\hat{Y}$ es la predicción del patrón, obtenida por la aproximación por series de Fourier.

- Series de Fourier no lineales. Esta alternativa consiste en utilizar series de Fourier que también incluyan combinaciones no lineales de las funciones sinusoidales, como, por ejemplo, con senos y cosenos elevados a naturales superiores a 1. El procedimiento de cálculo es idéntico al de las series de Fourier tradicionales, pues se utiliza también un regresor.

Este método permite seguir mejor los picos y extremos de las oscilaciones que la serie de Fourier clásica.

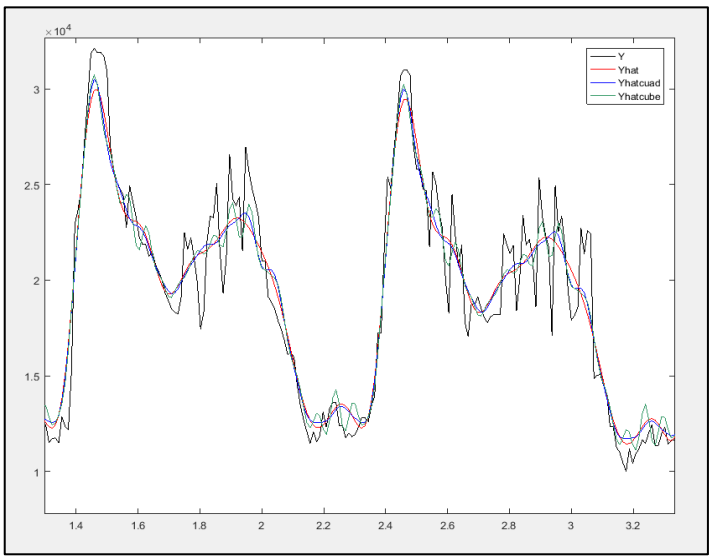

Imagen 49.- Comparación entre la aproximación por series de Fourier lineales y no lineales

Uso de la aplicación Curve Fitting Tool

La aplicación Curve Fitting Tool pertenece al conjunto de aplicaciones del programa de cálculo Matlab. Esta aplicación permite una secuencia de valores en un vector de distintas maneras, ajustando los parámetros de manera que se minimicen los errores. En el anexo 2.9.2. se encuentra un sencillo manual para su uso.

Las opciones que permite son las siguientes:

- Smoothing Splines. Esta opción consiste en ajustar la función dividiendo el vector en múltiples fragmentos, en los que en cada uno se define una curva o *spline* con una forma que intenta ajustarse a los valores obtenidos. Los *splines* también se definen de manera que la función a trozos obtenida sea continua y derivable. El inconveniente que presenta es que requiere una gran cantidad de parámetros para lograr una fiabilidad suficientemente alta, y la función se vuelve compleja.

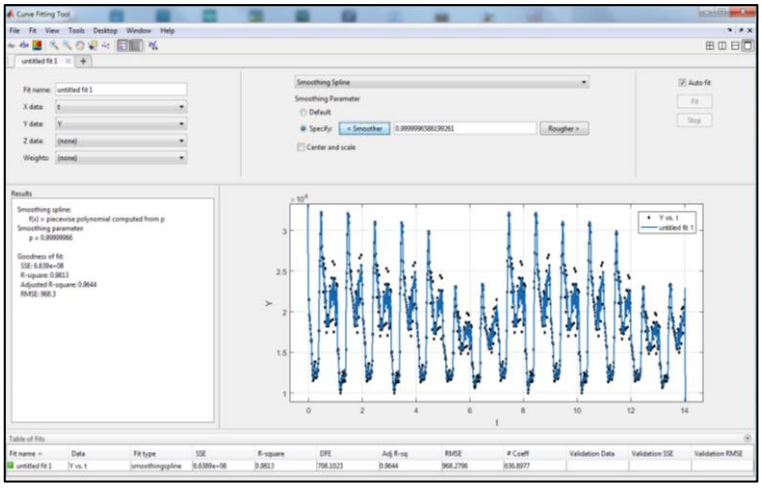

Imagen 50.- Aproximación por splines usando Curve Fitting Tool de Matlab

- Series de Fourier. Esta aplicación permite también ajustar los parámetros de una serie de Fourier a los datos seleccionados. No obstante, el código comentado anteriormente permite una mayor flexibilidad de edición y modificación.

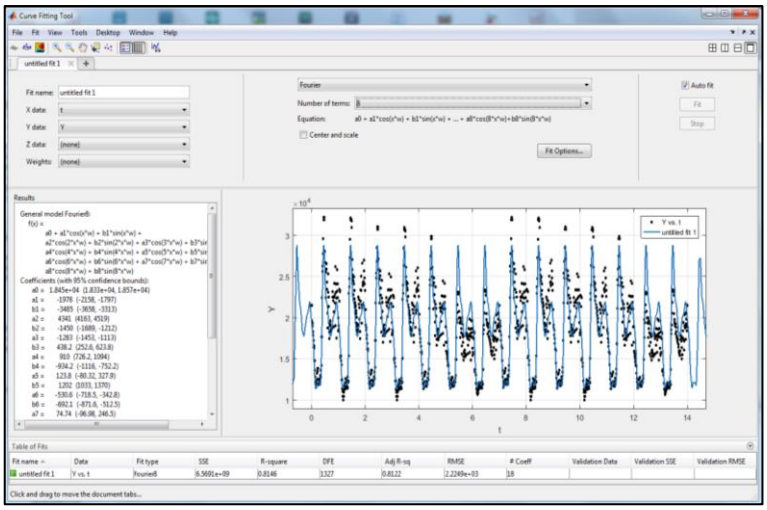

Imagen 51.- Aproximación por series de Fourier usando Curve Fitting Tool de Matlab

- **Polinomios.** Esta opción permite ajustar los parámetros a una función polinómica del tipo:

$$y = \sum_{i=1}^{P} a_i \cdot x^i \tag{2.71}$$

Donde P es el grado del polinomio. Sin embargo, el patrón que se está intentando obtener requiere muchos más grados de los que permite la aplicación, así que esta opción se descarta. Se prefiere utilizar la opción de Custom Equation.

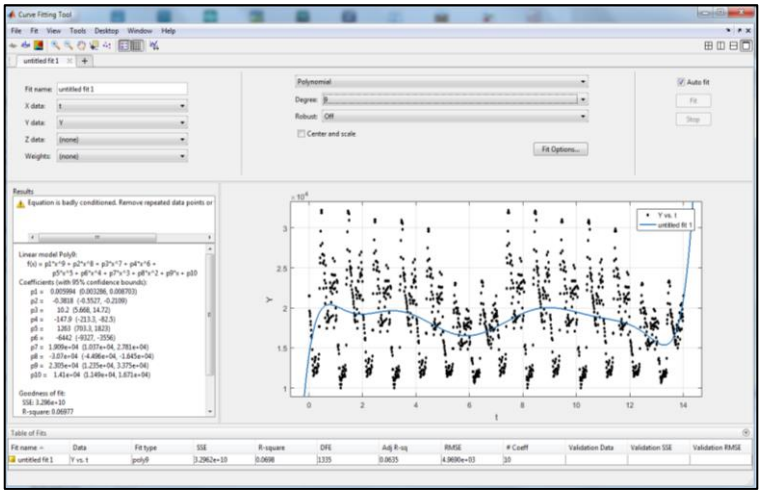

Imagen 52.- Aproximación por polinomios usando Curve Fitting Tool de Matlab

- **Custom Equation**. La aplicación también puede aproximar los parámetros de una ecuación personalizada. De esta manera se pueden probar polinomios de mayor grado que los que permite la aplicación por defecto, o series de Fourier no lineales. También pueden probarse otras opciones, como aproximadores universales, que utilizan combinaciones lineales de variaciones de funciones sigmoides.

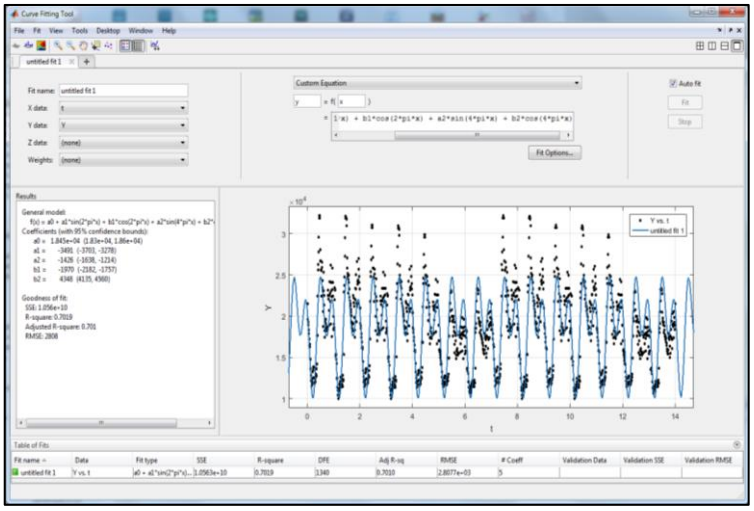

Imagen 53.- Aproximación por ecuaciones personalizadas utilizando Curve Fitting Tool

- **Redes neuronales.** Matlab permite también el entrenamiento y uso de redes neuronales para identificar los patrones de una señal mediante el uso de varias aplicaciones específicas. Aunque esta opción es potencialmente válida, se descarta debido a que hay otras opciones más simples e igual de válidas.

- Tablas. Esta opción consiste en obtener, mediante un programa realizado con código en Matlab, una tabla que incluya todos los valores que toma el patrón base durante una semana. Esta opción es la más precisa para identificar una señal, dado que se trata literalmente de una transcripción de dicha señal. Por otro lado, esta opción requiere de tantos parámetros como muestras tenga la señal. Aun así, sigue manteniendo una relación precisión/parámetros mayor que el spline (de las anteriores, la opción más precisa).

Para el diseño del algoritmo se ha utilizado esta opción, y para obtener la inicialización del patrón base se ha utilizado la media entre los valores de dos semanas en las que no ha habido lluvia.

```
load Datos_dry_2006.mat % Carga los datos
Y = Datosdry(15,:)';
Mdia = 24*4; % Muestras por día
PerfilSemanal=zeros(size(Y((1:Mdia*7))));
Nsembase=2;
for i=1:Nsembase
    PerfilSemanal=PerfilSemanal+Y((1:Mdia*7)+Mdia*7*(i-1))/Nsembase;
end
```

2.6.3.- ACTUALIZACIÓN DEL PATRÓN BASE

Para minimizar la incertidumbre del patrón base y eliminar cualquier error de inicialización del patrón calculado en el apartado anterior, se debe realizar una actualización con los nuevos valores de entrada. Esta actualización también es necesaria para el seguimiento de los patrones de estacionalidad.

La actualización se realiza, muestra a muestra de la tabla, aplicando un filtro IIR:

$$y_k = a \cdot y_{k-1} + (1-a) \cdot u_k \tag{2.72}$$

En este caso, se ha realizado una simple modificación, donde $\alpha = 1 - a$.

$$y_k = (1-\alpha) \cdot y_{k-1} + \alpha \cdot u_k \tag{2.73}$$

$$y_k = y_{k-1} - \alpha \cdot y_{k-1} + \alpha \cdot u_k \tag{2.74}$$

$$y_k = y_{k-1} + \alpha \cdot (u_k - y_{k-1}) \tag{2.75}$$

Por lo que para la actualización de la tabla se utiliza:

$$f_{t.base}(t) = f_{t.base}(t-1) + \alpha \cdot (y_{medida}(t) - f_{t.base}(t-1)) \tag{2.76}$$

La clave es encontrar un parámetro α con el valor oportuno. Si es muy pequeño, la actualización será muy resistente a perturbaciones y fallos no detectados, pero será incapaz de seguir de manera adecuada variaciones del consumo a largo plazo (estacionalidad). Si es muy elevado, la propia aleatoriedad del consumo distorsionará constantemente el patrón, y no será muy fiable (aumentará la incertidumbre del patrón).

El código utilizado en el diseño del algoritmo es así:

```
%% Actualización del patrón %Realidad es el vector entrante
NewPerfSem = zeros(length(PerfilSemanal), Nsemanas);
alfa = 0.01;
for i = 1:length(PerfilSemanal)
    NewPerfSem(i,1) = PerfilSemanal(i) + alfa*(Realidad(i,1)-
PerfilSemanal(i));
end
for i = 1:length(PerfilSemanal)
    NewPerfSem(i,j) = NewPerfSem(i,j-1) + alfa*(Realidad(i,j)-
NewPerfSem(i,j-1));
end
```

2.6.4.- DETECCIÓN DE LA LLUVIA

El último paso para conseguir una buena predicción del caudal entrante es obtener un algoritmo que permita la detección de fallos, que en este caso es la presencia de lluvia. Para ello se procede a reescribir la ecuación de la señal analizada:

$$y_{medida}(t) = f_{t.base}(t) + \Delta y_{inc}(t) + f_{fallo}(t) + v_{ruido}(t) \quad \textbf{(2.77)}$$

La lluvia es $f_{fallo}(t)$. Debido a que es una señal que cambia súbitamente, se mantiene constante durante un periodo de tiempo y luego vuelve a variar (considerando que las precipitaciones comienzan y acaban de golpe, con dinámicas lo suficientemente rápidas como para realizar esta simplificación), esta señal puede modelizarse de la siguiente manera:

$$\dot{f}_{fallo}(t) = \begin{cases} 0 \ en\ general \\ \Delta f_{fallo}\ en\ los\ momentos\ de\ aparición \end{cases} \quad \textbf{(2.78)}$$

Partiendo de la premisa anterior, se puede afirmar que:

$$f_{fallo}(t) = f_{fallo}(t-1) + \Delta f_{fallo}(t) \quad \textbf{(2.79)}$$

Sin embargo, es imposible conocer el valor real del fallo, dado que existe la incertidumbre del patrón y el ruido, que somos incapaces de predecir ni medir, por lo que hay que utilizar una estimación del fallo:

$$\hat{f}_{fallo}(t) = \hat{f}_{fallo}(t-1) + \beta \cdot \widehat{\Delta f}_{fallo}(t) \quad \textbf{(2.80)}$$

Donde $\widehat{\Delta f}_{fallo}(t)$ se calcula de la siguiente manera:

$$\widehat{\Delta f}_{fallo}(t) = y_{medida}(t) - f_{t.base}(t) - \hat{f}_{fallo}(t-1) \quad \textbf{(2.81)}$$

Juntando las dos ecuaciones anteriores se obtiene un observador del fallo:

$$\hat{f}_{fallo}(t) = \hat{f}_{fallo}(t-1) + \beta \cdot \left[y_{medida}(t) - f_{t.base}(t) - \hat{f}_{fallo}(t-1)\right] \quad \textbf{(2.82)}$$

Por lo que de β dependerá la velocidad de detección del fallo. No obstante, hay más características determinadas por ese parámetro. Para ello, se calcula el error cometido en la estimación ($\tilde{f}_{fallo}(t)$). Combinando las ecuaciones anteriores, se obtiene que:

$$\tilde{f}_{fallo}(t) = f_{fallo}(t) - \hat{f}_{fallo}(t) \quad \textbf{(2.83)}$$

$$\tilde{f}_{fallo}(t) = (1-\beta) \cdot \left[\tilde{f}_{fallo}(t-1) + \Delta f_{fallo}(t)\right] - \beta \cdot [\Delta y_{inc}(t) + v_{ruido}(t)] \quad \textbf{(2.84)}$$

Analizando la ecuación obtenida, se observa que si β se acerca a 1, los fallos se detectan muy rápidamente (el error producido por $[\tilde{f}_{fallo}(t-1) + \Delta f_{fallo}(t)]$ desaparece velozmente); sin embargo, al detector le afecta mucho tanto el ruido de medida como la incertidumbre del patrón (de ahí la necesidad de minimizarlo tanto en apartados anteriores), y los considera como lluvia. Por otro lado, una β cercana a 0 minimiza el efecto del ruido y la incertidumbre, pero ralentiza mucho la detección de fallos, dado que los errores anteriores tardan en desaparecer.

La implementación en código es similar a la de la actualización de las tablas base:

```
%% Detección de fallos
NewPerfSem = zeros(length(PerfilSemanal), Nsemanas);
fallohat = zeros(length(PerfilSemanal)*Nsemanas+1,1);
beta = 0.5;  f = 1;
for i = 1:length(PerfilSemanal)
    NewPerfSem(i,1) = PerfilSemanal(i) + alfa*(Realidad(i,1)-
PerfilSemanal(i));
    fallohat(f+1) = fallohat(f) + beta*(Realidad(i,1)-NewPerfSem(i,1)-
fallohat(f));
    f = f+1;
end
```

2.7.- APLICACIÓN DEL CONTROL PREDICTIVO EN EL PROYECTO

Una vez obtenidos toda la información necesaria, se procede a diseñar y aplicar el control predictivo. Es imprescindible utilizar un optimizador para poder minimizar la función de coste, por lo que se utilizará YALMIP.

2.7.1.- MONTAJE DE MATRICES

Dicho *parser* requiere que las matrices del modelo sean introducidas de manera concreta, separando las entradas que tiene que optimizar de las entradas anteriores ya calculadas. Como en el modelo por regresores que se ha calculado, en el vector de parámetros éstos están todos juntos, debe crearse una nueva matriz que redistribuya los parámetros de θ. En la siguiente matriz se presenta la manera de montarla cuando se calcula la salida del amonio a partir del $K_L a$, del SRI y del caudal entrante Q.

Partiendo del vector de parámetros:

$$\theta_1 = [a_0, a_1, a_2, \dots, b_1, b_2, \dots, c_1, c_2, \dots d_1, d_2, \dots] \quad \textbf{(2.85)}$$

Que multiplican a la salida anterior de amonio, al $K_L a$, al SRI, y al caudal, en ese orden, respectivamente (más el término inicial independiente), se redistribuyen los parámetros para que el modelo simplificado tenga la siguiente forma:

$$x_{k+1} = A \cdot x_k + B \cdot u_k + C \cdot a_0 \quad \textbf{(2.86)}$$

Donde:

$$x_{k+1}^T = [NH4_{k+1}, NH4_k, NH4_{k-1}, \dots, K_L a_k, K_L a_{k-1}, \dots, SRI_k, SRI_{k-1}, \dots, Q_k, Q_{k-1}, \dots] \quad \textbf{(2.87)}$$

$$x_k^T = [NH4_k, NH4_{k-1}, NH4_{k-2}, \dots, K_L a_{k-1}, K_L a_{k-2}, \dots, SRI_{k-1}, SRI_{k-2}, \dots, Q_{k-1}, Q_{k-2}] \quad \textbf{(2.88)}$$

$$u_k^T = [K_L a_k, SRI_k, Q_k] \quad \textbf{(2.89)}$$

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & b_2 & b_3 & \cdots & c_2 & c_3 & \cdots & d_2 & d_3 & \cdots \\ 1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \quad \textbf{(2.90)}$$

$$B^T = \begin{bmatrix} b_1 & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots \\ c_1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots \\ d_1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots \end{bmatrix} \quad \textbf{(2.91)}$$

$$C^T = [1 \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad \cdots] \quad \textbf{(2.92)}$$

Esta conversión se encuentra en el Apéndice A.4.2. – Montaje de matrices.

2.7.2.- FUNCIONES DE COSTE Y RESTRICCIONES

Tras tener toda la información de entrada al optimizador correctamente organizada, se procede a diseñar la función de coste. En este caso, el horizonte de predicción y el de control son idénticos, facilitando la implementación del modelo por regresores.

La función de coste que se quiere minimizar es la siguiente:

$$J_1 = \sum_{j=1}^{Hor_{pred}} precio_{elec}(j) \cdot K_L a(t+j-1) \quad \textbf{(2.93)}$$

Se somete a las siguientes restricciones:

La variación de la válvula se limita a un 0.5% a la hora. Esto se debe a que cambios bruscos de esta variable provocan sobreoscilaciones indeseadas. También se limita la salida del $K_L a$ de 0 a 240, y la válvula de 0 a 100%, para evitar soluciones físicamente imposibles.

La última restricción aplicada es que los valores de la salida de amonio no superen el límite indicado (9 $g\ N/m^3$, en este experimento).

Sin embargo, cuando a la entrada del sistema llega un aumento súbito de amonio y el optimizador cree que va a ser incapaz de mantener la salida bajo el límite de ninguna de las maneras, emite un mensaje de error que puede dar problemas a la hora de realizar el control. Por eso, se aplica un segundo control, de emergencia, con el objetivo único de reducir ese pico. La función de coste de emergencia que se quiere minimizar queda, por tanto:

$$J_2 = \left(\left(\text{limitemax}_{S_{NH}} - 1\right) - y_{amon(j+1)}\right)^2 \quad \textbf{(2.94)}$$

De esta manera, se intenta minimizar exclusivamente la siguiente salida bajo el límite establecido, a cualquier coste. Si el control anterior se ha diseñado correctamente, esta función aparece solo como emergencia, en casos muy puntuales de entradas masivas de amonio.

Las restricciones de este optimizador son las mismas que en el principal, excepto la limitación de salida inferior al umbral (que causaba la aparición del error).

2.7.3.- OTRAS CONSIDERACIONES

2.7.3.1.- SOLVERS POSIBLES

Los solvers permiten la optimización de la función. Cada uno utiliza sus métodos propios de cálculo, y pueden ser más o menos útiles dependiendo del caso a optimizar. No obstante, el análisis de las características de los solvers queda fuera del alcance de este Proyecto.

Los solvers que se han probado son `mosek`, `sedumi`, `cutsdp` y `fmincon`.

En este proyecto, el único punto importante que hay que tener en cuenta sobre los solvers numéricos es que no trabajan con restricciones estrictas de desigualdad, debido a que los cálculos se realizan con una tolerancia numérica. Por tanto, no responderán ante una restricción con un signo $<$ o $>$, deberán ser necesariamente $\leq$ o $\geq$.

2.7.3.2.- APLICACIÓN DE TARIFAS

Los datos utilizados para los costes de la electricidad se almacenan en la función `calcula_precio` (Apéndice A.5.4.). Estos datos pueden modificarse en cualquier momento, e incluir los precios de cualquier distribuidora o procedentes de la OMIE.

Pueden modificarse de manera que se tenga cualquier distribución horaria, y en un futuro se prevé la obtención directa de los datos y su ampliación a un año entero con todas las variaciones de tarifas.

2.8.- MODELADO DE LA EDAR MEDIANTE SIMULINK

2.8.1.- ESTRUCTURA DE LOS BLOQUES DE SIMULINK

El entorno de Simulink permite observar el comportamiento de un sistema de manera muy visual, de manera que una primera implementación del modelo de la EDAR en este entorno permite comprobar de manera rápida si la escritura y conversión de las ecuaciones desde el Benchmark se ha realizado correctamente.

Simulink funciona con bloques que realizan distintas funciones. Algunos de dichos bloques pueden englobar a un conjunto de subbloques interrelacionados que realizan funciones con un mismo objetivo. En el caso de este modelo, se han diferenciado tres bloques distintos: el reactor biológico, el decantador de fangos y la interconexión entre tuberías.

En la imagen 54 se identifican los tres bloques, así como la entrada del $K_L a$ (bloque escalón oxigeno), la entrada de caudal externo (el bloque *From File*) y la válvula de recirculación (los dos triángulos rojos *Gain* del centro), así como otros valores de recirculación no controlables, un par de selectores de señales y varios bloques *Scope* que dibujan gráficas.

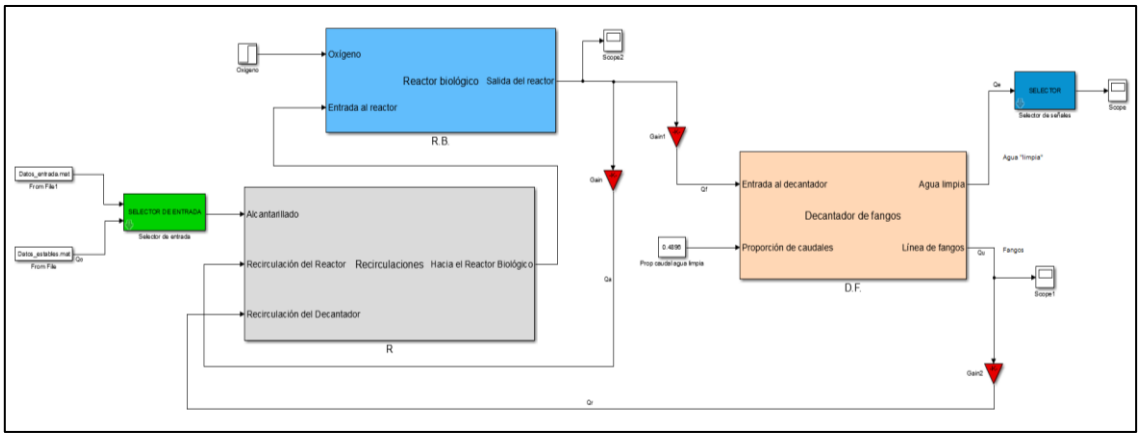

Imagen 54.- Modelo del BSM1 en Simulink

- **Reactor biológico**. Se modeliza como una concatenación de cinco bloques Level-2 Matlab S-Function (ver Anexo 2.8.2.- Cómo programar una Level-2 Matlab S-Function), que representan cada compartimento del reactor. Cada bloque tiene dos entradas: el vector de caudal con las concentraciones del reactor anterior (o en el primer caso, desde el exterior del reactor) y la entrada de oxígeno ($K_L a$, concretamente) al compartimento; también tiene una salida, que es el vector de caudal con las concentraciones de ese compartimento.

El $K_L a$ que entra a cada compartimento se modeliza con una constante igual a cero para los dos primeros compartimentos, dado que son anóxicos; un acceso desde fuera del bloque para la $K_L a$ del quinto compartimento, que es el único $K_L a$ controlable; y dos escalones (que a términos eficaces funcionan como constantes a 240) para el tercer y cuarto compartimento.

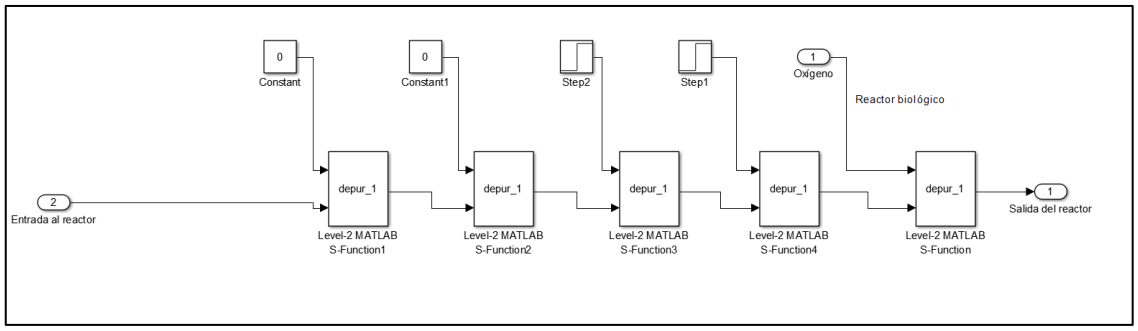

Imagen 55.- Modelo del reactor biológico en Simulink

- **Decantador**. Debido a que en el Benchmark se realiza una conversión de las concentraciones de componentes particulados en una única variable (fangos particulados), la estructura de bloques interna del decantador debe reflejar las mismas operaciones. Por tanto, el primer bloque es una función de Matlab que realiza dicha conversión, además de almacenar en un vector las proporciones entre la nueva variable y los componentes originales, que son utilizados más adelante.

Para modelizar el comportamiento dinámico del decantador, se utiliza una Level-2 Matlab S-Function, que recibe los datos del caudal de entrada con conversión y los de un divisor de caudal (no modificable). La salida de este bloque es un vector con todos los estados (concentraciones) de las 10 capas del decantador, y posteriormente se seleccionan los estados de la capa superior y la inferior mediante bloques de funciones. A estos vectores se les aplica una conversión inversa a la anterior para volver a tener todos los estados particulados.

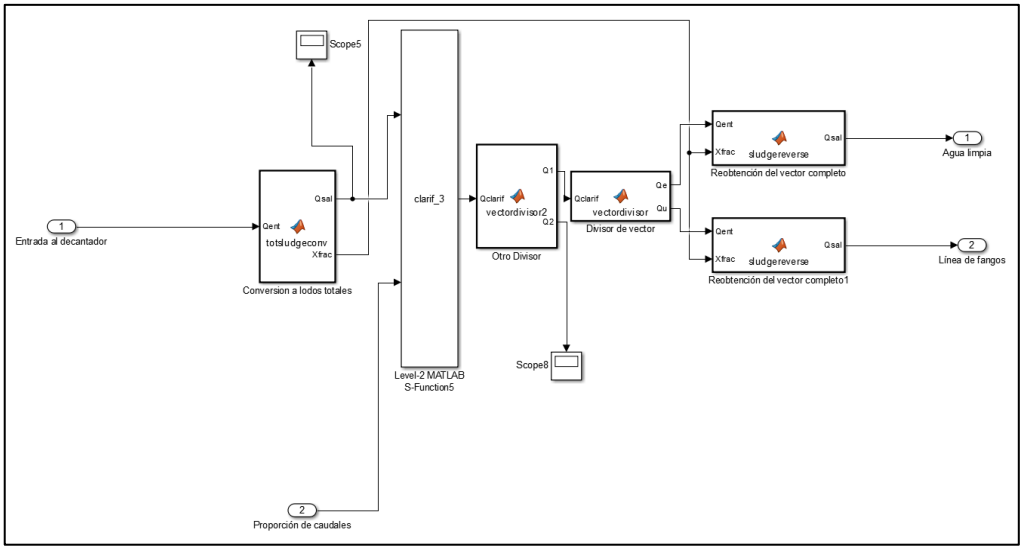

Imagen 56.- Modelo del decantador secundario en Simulink

- **Interconexión entre tuberías**. El último grupo recibe los valores de recirculación del reactor (la recirculación interna) y la procedente del decantador (la externa), así como los valores de caudal afluente desde un fichero externo. El bloque suma los caudales y las concentraciones de manera ponderada, utilizando las proporciones másicas, y se obtiene el caudal que llega al primer compartimento del reactor.

También pueden observarse unos bloques denominados *Memory*. Estos bloques ayudan a la inicialización del sistema, para evitar que ocurra una suma de caudal con valores cero durante las primeras iteraciones de la simulación.

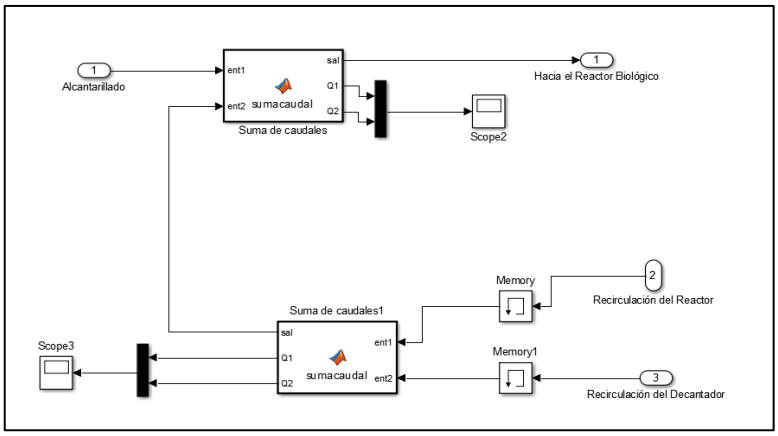

Imagen 57.- Modelo de la recirculación de caudales en Simulink

Una vez adaptado el modelo, se puede proceder a una simulación, para descubrir los puntos de funcionamiento de la depuradora. Para ello se utiliza el *Scope* que se ha visto en la imagen 54.

2.8.2.- CÓMO PROGRAMAR UNA LEVEL-2 MATLAB S-FUNCTION

Una Level-2 Matlab S-Function (S-Function a partir de ahora) es un bloque de funciones para Simulink. Este tipo de bloques permite la resolución y simulación de sistemas de ecuaciones diferenciales en tiempo continuo, por lo que son realmente útiles. En este Proyecto se han utilizado para modelizar el comportamiento de los cinco compartimentos del reactor biológico y el del decantador.

No obstante, tienen la desventaja de que su código está escrito de manera muy estructurada, así que para la correcta implementación de un modelo dinámico en este tipo de bloques se debe realizar una adecuada conversión de las ecuaciones diferenciales.

Para entender el funcionamiento de las S-Function, hay que tener en cuenta que se basan en la representación del sistema en **espacio de estados**.

$$\begin{cases} \dot{x} = A \cdot x + B \cdot u \\ y = C \cdot x + D \cdot u \end{cases} \quad \textbf{(2.95)}$$

Donde x son los estados internos (o variables internas) del sistema, la u es la entrada al sistema, y la y la salida del sistema.

De esta manera, el espacio de estados indica que la ratio de cambio (derivada) de los estados internos depende del producto de los estados internos por una matriz A a la que se añade el producto de una matriz B por las entradas al sistema. Por otro lado, la salida del sistema depende del producto de una matriz C por el estado actual de las variables a la que se le añade el producto de una matriz D por las entradas al sistema. Sin embargo, la matriz D suele ser cero frecuentemente (aunque no en este Proyecto), debido a que, de no ser así, simboliza que el sistema tiene ganancia directa, y que los cambios en las entradas afectan directamente a las salidas sin ninguna dinámica.

Teniendo en cuenta esta consideración, se procede a la exposición de los distintos apartados que hay que completar al implementar una S-Function.

```
function NombredelaSfunction(block)
% Level-2 MATLAB file S-Function for limited integrator demo.
% Copyright 1990-2009 The MathWorks, Inc.
  setup(block);

function setup(block)
  %% Número de parámetros (fijos) que proceden del exterior.
  block.NumDialogPrms = X;

  %% Número de entradas y salidas. Cada entrada o salida puede consistir
en un array.
  block.NumInputPorts  = X;
  block.NumOutputPorts = X;

  %% Setup functional port properties to dynamically inherited. (Dejar
así)
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

%% Establecer las dimensiones del array para cada entrada y salida, y
si alguno de esos valores va a necesitar la matriz D.
  block.InputPort(2).Dimensions        = X;
  block.InputPort(2).DirectFeedthrough = true/false;
  block.InputPort(1).Dimensions        = X;
  block.InputPort(1).DirectFeedthrough = true/false;
  block.OutputPort(1).Dimensions       = X;

  %% Set block sample time to continuous (dejar así para que simule en
tiempo continuo)
  block.SampleTimes = [0 0];

  %% Determinar el número de estados x dentro del sistema
  block.NumContStates = X;

  %% Set the block simStateCompliance to default (i.e., same as a built-
in block) (Dejar así)
  block.SimStateCompliance = 'DefaultSimState';
  %% Register methods (Dejar así)
  block.RegBlockMethod('InitializeConditions',    @InitConditions);
  block.RegBlockMethod('Outputs',                 @Output);
  block.RegBlockMethod('Derivatives',             @Derivative);

function InitConditions(block)
% Inicialización de los estados x. Los estados pueden inicializarse con
los DialogPrms.
  block.ContStates.Data = X;

function Output(block)
  %% Asignación de los valores de salida.
  %% En este punto se aplica y = C·x + D·u.
  %% Aquí, y se almacena en block.OutputPort(numsalida).Data(posicion)

function Derivative(block)
  %% En este punto es donde se aplica ẋ = A·x + B·u, utilizando las
ecuaciones del modelo.
  %% Los valores de x se almacenan en block.ContStates.Data(), y los de
ẋ en block.Derivatives.Data().
  %% Estos almacenes funcionan como arrays, así que en los paréntesis
se debe indicar qué valores se quieren usar.
```

2.9.- MANUALES DE INSTRUCCIONES

2.9.1.- IDENTIFICACIÓN MEDIANTE RESPUESTA ANTE ESCALÓN (IDENTESCALON).

Esta herramienta del conjunto de programas Freepidtools permite identificar un sistema analizando su respuesta ante una entrada escalón. Para lograrlo, hay que seguir los siguientes pasos:

1.- Realizar el experimento para obtener los datos. Hay que tener en cuenta que, en un sistema multivariable, todas las entradas deben empezar en su valor de equilibrio, así como las salidas, y el experimento no debe acabar hasta que las salidas no tengan un comportamiento estable.

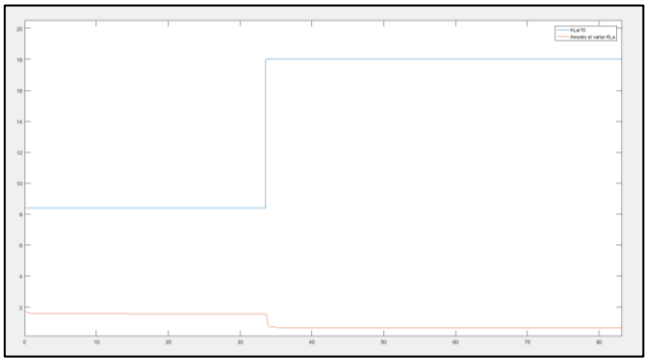

Imagen 58.- Entrada escalón al sistema

2.- Reordenar los datos en una variable, de manera que los datos estén en columnas en este orden: [tiempo, entrada, salida]. Guardarlos en un fichero txt usando sabe en formato ascii y sin tabuladores, utilizando este código:

```
save nombredelfichero.txt nombredelavariable -ascii -tabs
```

Copia el fichero en la misma carpeta donde se encuentran las Freepidtools.

3.- Abrir la herramienta ejs_model_identescalon26. Pulsar "*Reiniciar antes de cargar nuevos datos*". Comprobar que la opción [t, u, y] está marcada. Pulsar "*Cargar datos experimentales*" y seleccionar el fichero con los datos seleccionados.

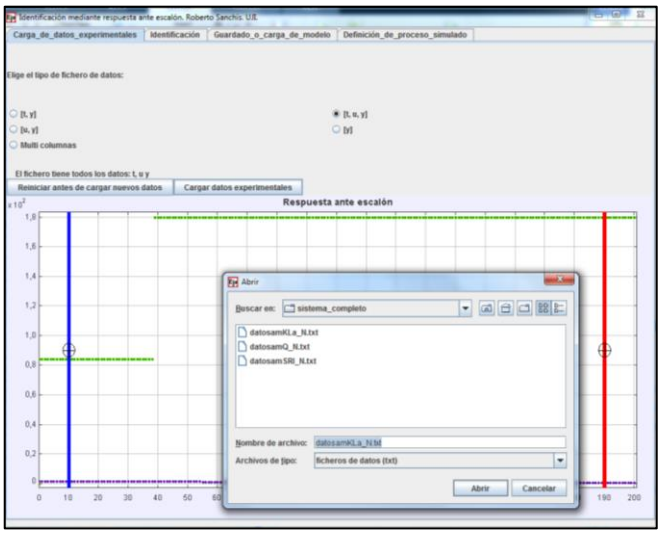

Imagen 59.- Introducción de datos en identescalon

4.- Mover las barras azul y roja para delimitar el número de datos a analizar. Es recomendable situar la barra azul justo antes del escalón, y la roja cuando la salida se haya estabilizado.

5.- Pasar a la pestaña *Identificación*. Ajustar los ejes ymax, ymin para visualizar de manera correcta la salida del sistema.

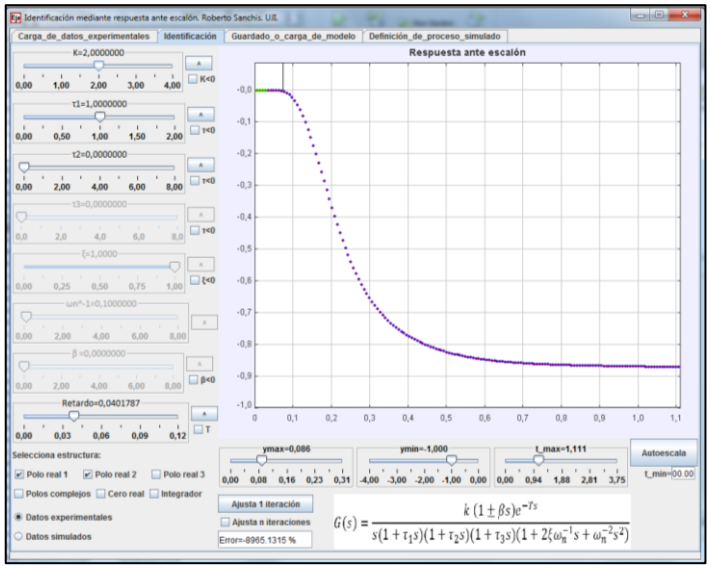

Imagen 60.- Ajuste de ejes en identescalon

6.- Ajustar los valores de la ganancia K el retardo, y los polos τ_1 en adelante (para añadir más polos, ceros o comportamiento oscilatorio, se debe pulsar en las casillas de la parte inferior izquierda).

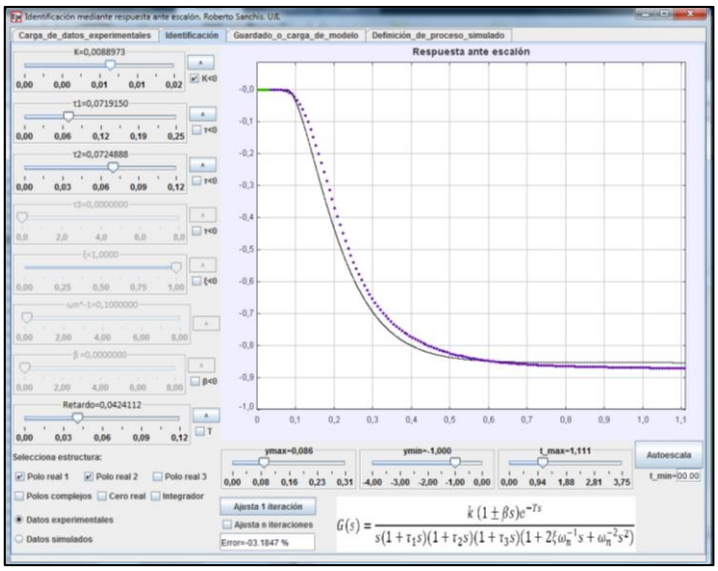

Imagen 61.- Ajuste de la curva en identescalon

7.- Para acabar de ajustar la identificación, pulsar en la casilla "*Ajusta n iteraciones*". De esta manera, la herramienta modificará los valores de los parámetros para minimizar el error cometido.

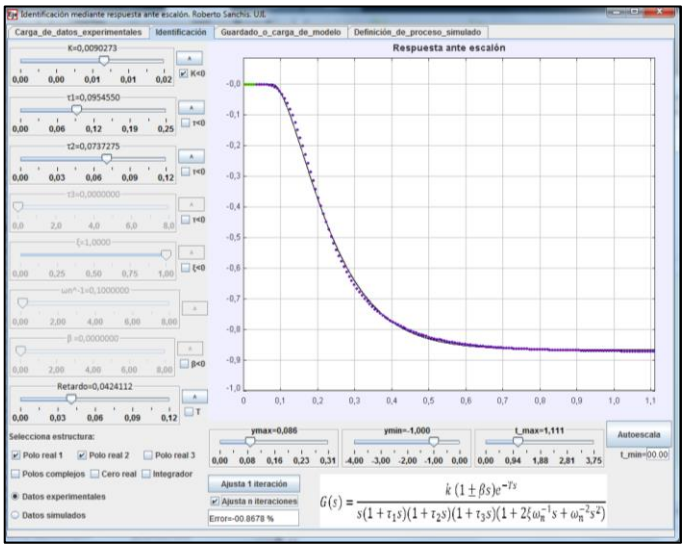

Imagen 62.- Ajuste automático de la curva en identescalon

8.- Para pasar la identificación a Matlab, hay que pulsar en la pestaña "Guardado o carga de modelo" y copiar la línea inferior en el área de texto (el programa deja por defecto los parámetros no usados a cero, es recomendable borrarlos). También puede guardarse el archivo con la identificación pulsando "Guarda modelo en disco".

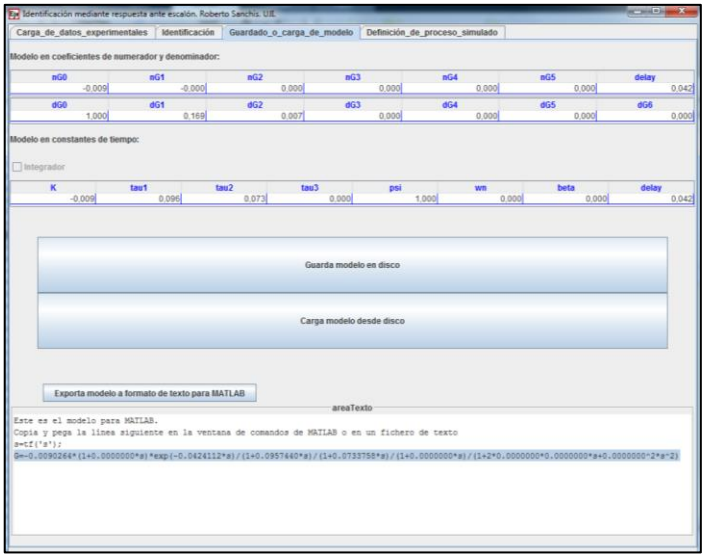

Imagen 63.- Obtención del modelo en identescalon

2.9.2.- CURVE FITTING TOOL DE MATLAB

La aplicación de Matlab Curve Fitting Tool, como su nombre indica, es una herramienta muy útil que permite ajustar una serie de datos obtenidos a una curva o serie de curvas. Su uso es muy simple:

1.- Realizar el experimento para obtener los datos y almacenar tanto los vectores de tiempo (eje X) como el vector de datos (eje Y).

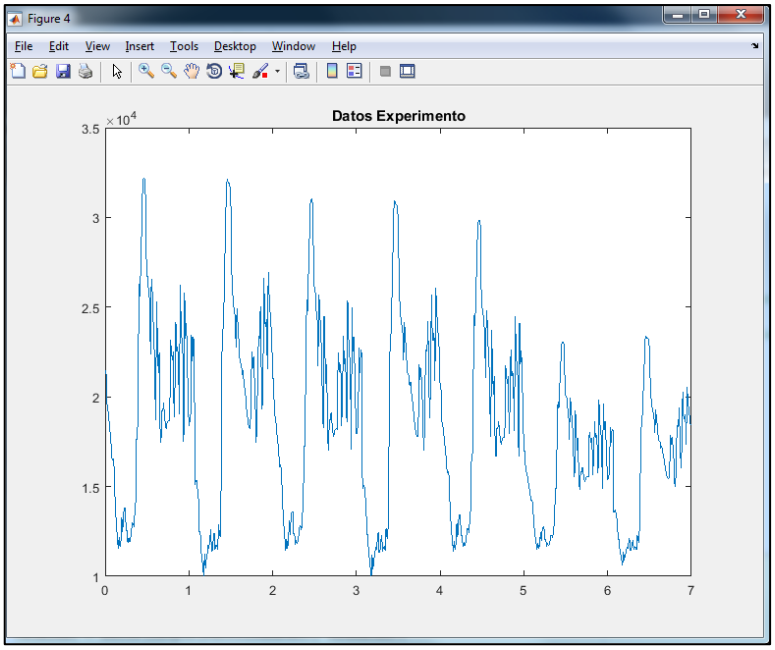

Imagen 64.- Realización del experimento

2.- Abrir la aplicación Curve Fitting Tool, e introducir en los apartados de la esquina superior izquierda las variables de los ejes donde correspondan. En la pestaña que se desplegará aparecen todas las variables del Workspace que puede utilizar la aplicación.

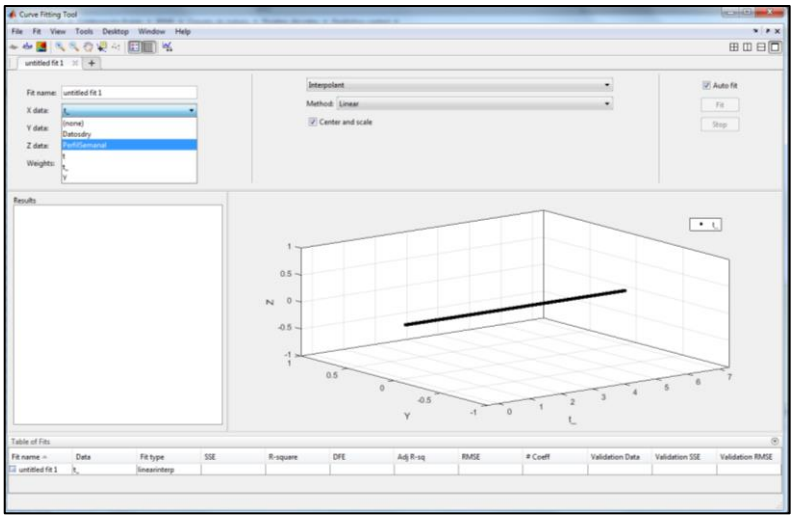

Imagen 65.- Introducción de las variables

3.- Seleccionar en la esquina superior izquierda el tipo de ajuste que se desea (polinomial, Fourier, smoothing spline, custom equation, etc.). En función del ajuste seleccionado, aparecerán otras pestañas donde se puede ajustar el número de parámetros utilizados para el ajuste de curva.

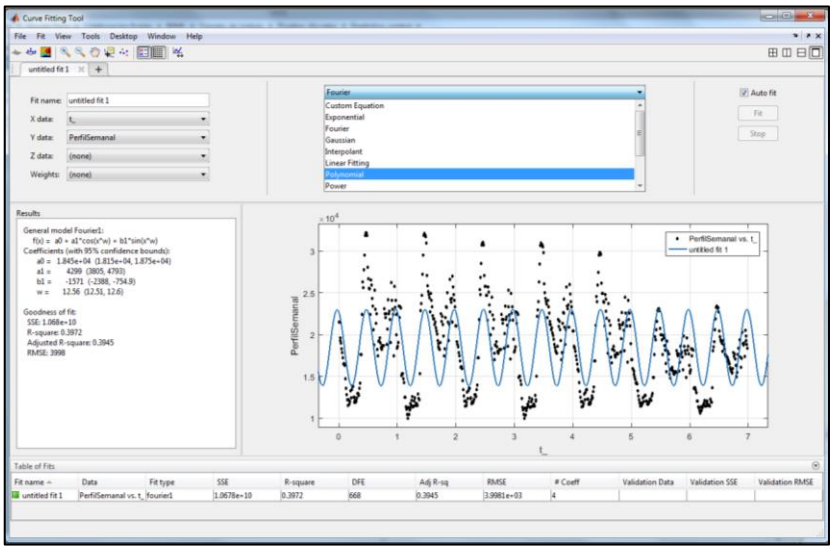

Imagen 66.- Selección del método de aproximación

4.- Modificar el número de parámetros para encontrar un adecuado equilibrio respecto a la precisión de la curva aproximada. Para ello se puede observar la tabla inferior, donde aparecen los ajustes de R^2, entre otros.

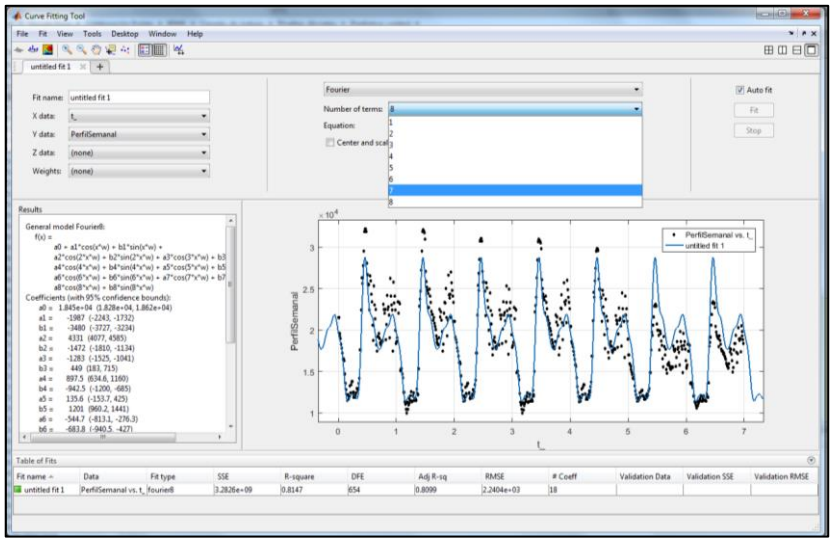

Imagen 67.- Determinación del número de parámetros deseado

2.9.3.- CÓMO DESCARGAR E INSTALAR YALMIP EN MATLAB

Para poder utilizar YALMIP en Matlab, primero hay que descargarlo. El paso es muy sencillo, solo hay que escribir el siguiente código en la ventana de comandos de Matlab, estando en el directorio donde quieres que se instale:

```
cd YALMIPfolderShouldbeHere
urlwrite('https://github.com/yalmip/yalmip/archive/master.zip','yalmip.zip');
unzip('yalmip.zip','yalmip')
addpath(genpath([pwd filesep 'yalmip']));
savepath
```

Y la descarga e instalación se efectuará de manera automática.

Debido a la complejidad de funcionamiento de YALMIP, se recomienda consultar su página web (citada en el Apartado 1.4.4.- Otras referencias) para obtener más información sobre el diseño de algoritmos de optimización con YALMIP.

2.10.- CÓDIGO EMPLEADO

El código empleado para el desarrollo del algoritmo de la EDAR en Matlab se encuentra en el Apéndice A.

Por otro lado, el código empleado en el desarrollo del modelo de la EDAR en Simulink se encuentra en el Apéndice B.

3.- PLIEGO DE CONDICIONES

3.1.- INTRODUCCIÓN

Debido al alcance de este Proyecto, así como de sus características técnicas, la extensión del Pliego de Condiciones solo incluye los siguientes tres apartados.

3.2.- REQUISITOS

Todo los experimentos y resultados obtenidos se han realizado partiendo de las ecuaciones y premisas que plantea el Benchmark Simulation Model no. 1 (BSM1), de J. Álex, L. Benedetti et al. Por tanto, todos los parámetros de control aplicados, así como las restricciones y limitaciones implementadas en el diseño del algoritmo de control del sistema son válidos exclusivamente para el modelo de la EDAR mostrado en dicho documento.

Este algoritmo, editando y adaptando los parámetros y restricciones de manera oportuna, podría implementarse en una EDAR real con las mismas características que la descrita en el BSM1. Sin embargo, la implementación de dicho algoritmo de control a una EDAR real excede las competencias del este Proyecto, y requeriría la realización de uno distinto al presente.

3.3.- MATERIALES EMPLEADOS

3.3.1.- SOFTWARE

Para la programación, diseño de algoritmos, simulación y ejecución de todos los programas escritos en este proyecto se requiere un Matlab de versión R2016a o posterior. Una versión más antigua de dicho programa podría no disponer de todas las funciones, bloques o aplicaciones mostradas en el Proyecto.

3.3.2.- HARDWARE

Para poder ejecutar este proyecto se requiere un ordenador con suficiente potencia como para poder ejecutar Matlab, y tenga unas características mínimas tales como 8 GB de RAM o un procesador de 2.90 GHz.

3.4.- CRITERIOS DE TOMA DE DATOS

Todos los datos tomados en este Proyecto se han adquirido siguiendo los pasos e instrucciones indicadas en los Anexos 2.4., 2.5., 2.6. y 2.7., así como en los manuales de instrucciones del Anexo 2.10. Las técnicas de toma de datos se han basado en los métodos aprendidos en las asignaturas de control citadas en el Apartado 1.4.3.- Bibliografía.

4.- PRESUPUESTO

4.1.- PRESUPUESTO DE EJECUCIÓN MATERIAL

Tabla 6.- Costes de Material y Licencias

COSTES DE MATERIAL Y LICENCIAS			
COMPONENTES	UNIDADES	PRECIO UNITARIO (€)	COSTE (€)
LICENCIA DE MATLAB	1	500	500
ORDENADOR DE SOBREMESA	1	820	820
COSTE TOTAL (€)			**1320**

Tabla 7.- Costes de Ingeniería

COSTES DE INGENIERÍA			
CONCEPTO	HORAS	PRECIO POR HORA (€)	COSTE (€)
DESARROLLO DEL ALGORITMO	400	20	8000
COSTE TOTAL (€)			**8000**

Tabla 8.- Presupuesto de Ejecución Material

PRESUPUESTO DE EJECUCIÓN MATERIAL	
CONCEPTO	COSTE (€)
COSTES DE MATERIAL Y LICENCIAS	1320
COSTES DE INGENIERÍA	8000
TOTAL	**9320**

4.2.- PRESUPUESTO DE EJECUCIÓN POR CONTRATA

Tabla 9.- Presupuesto de Ejecución por Contrata Parcial

PRESUPUESTO DE EJECUCIÓN POR CONTRATA PARCIAL	
CONCEPTO	COSTE (€)
PRESUPUESTO DE EJECUCIÓN MATERIAL	9320
GASTOS GENERALES (18%)	1677,6
BENEFICIO INDUSTRIAL (6%)	559,2
TOTAL	**11556,8**

Tabla 10.- Presupuesto por Ejecución por Contrata Total

PRESUPUESTO POR EJECUCIÓN POR CONTRATA TOTAL	
CONCEPTO	COSTE (€)
PRESUPUESTO POR EJECUCIÓN POR CONTRATA PARCIAL	11556,8
21% IVA	2426,93
TOTAL	**13983,73**

APÉNDICE A

A.1.- INTRODUCCIÓN

En este apéndice se listan, de manera ordenada, los programas y funciones utilizados para el modelado, simulación y control predictivo de la EDAR en Matlab. El código se muestra en el siguiente orden:

A.2.- INICIALIZACIÓN DEL ALGORITMO DE CONTROL

A.2.1.- PROGRAMA PRINCIPAL

```
%% Programa principal
% Este programa llama a todos los otros subprogramas
% Cada subprograma resuelve una parte del problema de diseño del
% algoritmo
format compact
%% Inicio del experimento
Inicializa_experimento
%% Obtención del modelo simplificado de la EDAR
Obtencion_modelo_simplificado
%% Diseño del sistema de control predictivo
Design_sistema_control_predictivo
%% Aplicación del control predictivo
Aplicacion_control_predictivo
%% Gráficas
Graficas_control_predictivo
```

A.2.2.- INICIALIZACIÓN DE VARIABLES Y DE MODOS DE CONTROL

```
%% Inicializa_experimento
% En este programa se inicializan todas las variables más relevantes.
% También se pueden editar los distintos modos de cálculo del
% algoritmo
%% Cerrar todo y limpiar
close all
clear all
clc
%% Intervalos de tiempo
f_muestreo = 1; %60 % f_muestreo es el divisor de tiempo de la
simulación.
N = 60000*f_muestreo; % N es el número de iteraciones temporales
(minutos*f_muestreo)
Ts = 1/60/24/f_muestreo; %Ts=1min/f_muestreo; % Es la frecuencia de
muestreo, en días
ts_enmin = 1/f_muestreo;
%% ¿Recalcular datos del regresor?
recalcular = 0;
%% Decimator
decimator = 60;  % se decima cada X min % Si decimator es 60, las
muestras se han tomado cada hora
decimation = decimator*f_muestreo; % Para realizar la conversión de
muestras decimadas a min
decimator_en_horas = decimator/60;
%% Orden del regresor
ordenregr_enhoras = 13;
ordenregr = round(ordenregr_enhoras*60/decimator); % Orden del
regresor
%% Varianza del regresor
peso_var_th(1) = 50;
peso_var_th(2) = 50;
%% Tipo de regresor
regresor_tipo = 0;
% regresor_tipo 0 -> y u
% regresor_tipo 1 -> u
% regresor_tipo 2 -> G
%% Inicialización del experimento de control predictivo
num_horas_experimento = 1000-48;
num_optimiz = round(num_horas_experimento*60/decimator); % Número de
optimizaciones, realizadas cada "decimator minutos"
num_muestras_experimento = decimation*num_optimiz;
v = 0; % Cada v es una muestra distinta del experimento controlado
%% Horizonte de predicción
horpred_enhoras = 45;
horpred = horpred_enhoras*60/decimator; % Horizonte de predicción del
control predictivo, en unidades del decimator
%% Variables de decisión (KLa, SRI) % Actualmente sobreescritas,
arreglar
peso_Q = 1*eye(1);
peso_R = 1*eye(1);
%% Seguimiento de referencias
refval1_t1 = 0.8;
refval2_t1 = 5.0;
refesc_t1 = int32(0.3*(num_optimiz+horpred));
refval1_t2 = 2461690;% 2.8e6; %
refval2_t2 = 2560000;%3.2e6; %2461690;
refesc_t2 = int32(0.3*(num_optimiz+horpred));
%% Useful info
```

```
Info_entre_muestras = 'Entre muestra y muestra pasan %4.2f minutos.
\n';
fprintf(Info_entre_muestras, ts_enmin)
Info_decimator = 'Los datos usados en el experimento se toman cada
%4.1f minutos, es decir, cada %4.2f horas. \n';
fprintf(Info_decimator,decimator,decimator_en_horas);
Info_regresor = 'El regresor utiliza los datos de las anteriores %3.0f
horas (orden %3.0f). \n';
fprintf(Info_regresor,ordenregr_enhoras, ordenregr);
Info_optimizaciones = 'El experimento dura %3.0f horas= %3.0f días, y
utiliza %5.0f muestras. \n'; % Arreglar, está mal
fprintf(Info_optimizaciones,
num_horas_experimento,num_horas_experimento/24,
num_muestras_experimento);
Info_horpred = 'La predicción se realiza a %3.0f horas. \n';
fprintf(Info_horpred, horpred_enhoras);
%% ¿Activar control relé?
controlrele=0;
```

A.3.- CÁLCULO DEL MODELO SIMPLIFICADO DE LA EDAR

A.3.1.- OBTENCIÓN DEL MODELO SIMPLIFICADO

```
%% Obtencion_modelo_simplificado
% Este modelo realiza el cálculo del modelo lineal simplificado.
% Primero se prepara el vector de entrada para la simulación del
% comportamiento de la depuradora, luego se realiza dicha simulación,
% y por último se realizan los cálculos para obtener el modelo
% simplificado del regresor.
%% Preparación del vector de entrada
% Selección de datos de entrada
load Datos_secos_interpolados.mat % Contiene los datos interpolados
para cada minuto para la entrada real
Influent_B = repelem(Datossecos(2:end,:),1,f_muestreo);
load Datos_estables_interpolados.mat % Contiene los datos interpolados
para cada minuto para la entrada estable
Influent_C = repelem(Datestint(2:end,:),1,f_muestreo);
Influent_D = Influent_C;%[Influent_C(1:(end-1),:);Influent_B(end,:)];
load vectal_minuto.mat;
% Generamos el vector de entrada con la dimensión temporal deseada
vectal_b = repelem(vectal(2:end, :),1,f_muestreo); % Usar -
New_vector_aleatorio.m
% Limitador rampa de SRI
Limitador_rampa_SRI_1
% pause
%% Regresor - recursivo
if recalcular
    Datos_para_el_calculo_del_regresor
    save('Datos_calculo_nuevo_regresor.mat');
else

load('Datos_calculo_nuevo_regresor.mat','randvect','Influent','Effluen
t','Psi_w');
end
Calculo_del_regresor_1
```

A.3.2.- SIMULACIÓN PARA OBTENER DATOS PARA EL CÁLCULO POR REGRESORES

```
%% Datos para el cálculo por regresor
% En este programa se aplican las ecuaciones adaptadas del Benchmark
para
% simularlas y obtener los datos para calcular el regresor.

% Multiplicamos el caudal de entrada por la ponderación calculada
Influent =
Influent_D.*[ones(13,60481*f_muestreo);randvect(3,1:60481*f_muestreo)]
;
% Valores de KLa
KLa = [zeros(2,N);240*ones(2,N);randvect(1,1:N)];
% Reserva de tamaño de los vectores
% ZsalX es el vector que almacena las variables de estado/componentes
de las aguas residuales en cada unidad de tiempo
% QsalX es el vector que almacena los valores de caudal
Zsal1 = zeros(13,N);  Zsal2 = zeros(13,N);  Zsal3 = zeros(13,N);
Zsal4 = zeros(13,N);  Zsal5 = zeros(13,N);
Qsal1 = zeros(1,N);   Qsal2 = zeros(1,N);   Qsal3 = zeros(1,N);
Qsal4 = zeros(1,N);   Qsal5 = zeros(1,N);
% Los vectores con dimensión 14,N contienen los componentes y el
caudal
% Acceso_decantador tiene los componentes solubles, los componentes
particulados agrupados y el caudal
% Fraccion_fango tiene las fracciones correspondientes a los
particulados
% respecto a la componente total particulada
Recirc_interna = zeros(14,N);
Acceso_reactor = zeros(14,N); Salida_reactor = zeros(14,N);
Acceso_decantador = zeros(9,N); Fraccion_fango = zeros(6,N);
Effluent = zeros(14,N); Underflow = zeros(14,N);
Recirc_externa = zeros(14,N); Wastage = zeros(14,N);
Suma_recirc = zeros(14,N);
Qeff = zeros(1,N);
Qundf = zeros(1,N);
% Los valores de los 8 estados de cada capa (7 solubles + 1
particulado
% total) se encuentran en la variable Estados_Decantador
Estados_Decantador = zeros(8,10,N);
Salida_inferior_decantador = zeros(9,N);
Salida_superior_decantador = zeros(9,N);
% SalDec_ se utiliza para mover los ejes de las dimensiones de la
matriz Estados_Decantador
SalDec_{1} = zeros(8,N);
SalDec_{2} = zeros(8,N);
SalDec_{3} = zeros(8,N);
SalDec_{4} = zeros(8,N);
SalDec_{5} = zeros(8,N);
SalDec_{6} = zeros(8,N);
SalDec_{7} = zeros(8,N);
SalDec_{8} = zeros(8,N);
SalDec_{9} = zeros(8,N);
SalDec_{10} = zeros(8,N);
Psi_w = zeros(1,N);

% Inicializaciones
% Valores iniciales del reactor
% Estos son los valores iniciales de los componentes de cada reactor
según el documento
```

```
Zsal1(:,1) = [30 2.81 1149 82.1 2552 148 449 0.00430 5.37 7.92 1.22
5.28 4.93]';
Zsal2(:,1) = [30 1.46 1149 76.4 2553 148 450 0.0000631 3.66 8.34 0.882
5.03 5.08]';
Zsal3(:,1) = [30 1.15 1149 64.9 2557 149 450 1.72 6.54 5.55 0.829 4.39
4.67]';
Zsal4(:,1) = [30 0.995 1149 55.7 2559 150 451 2.43 9.30 2.97 0.767
3.88 4.29]';
Zsal5(:,1) = [30 0.889 1149 49.3 2559 150 452 0.491 10.4 1.73 0.688
3.53 4.13]';
Qsal1(1) = 92230;
Qsal2(1) = 92230;
Qsal3(1) = 92230;
Qsal4(1) = 92230;
Qsal5(1) = 92230;
% Valores iniciales del decantador
Estados_Decantador(1,:,1) = ones(1,10)*30;
Estados_Decantador(2,:,1) = ones(1,10)*0.889;
Estados_Decantador(3,:,1) = ones(1,10)*0.491;
Estados_Decantador(4,:,1) = ones(1,10)*10.4;
Estados_Decantador(5,:,1) = ones(1,10)*1.73;
Estados_Decantador(6,:,1) = ones(1,10)*0.688;
Estados_Decantador(7,:,1) = ones(1,10)*4.13;
Estados_Decantador(8,:,1) = [6394 356 356 356 356 356 69.0 29.5 18.1
12.5];
% Modificar parámetros
% SRI=0.6; % Selector de Recirculación Interna
SRE=0.9796; % Selector de Recirculación Externa
DF=0.4896; % Porcentaje de flujo de entrada al decantador que sale
como effluent
SRI_ = randvect(2,1:N);

% Iteraciones
for k=1:N-1
    if mod(k, 1000) == 0
        k
    end
    % Se filtra la válvula
    if k==1
        SRI(k)=0.9995*0.6+(1-0.9995)*SRI_(k);
    else
        SRI(k)=0.9995*SRI(1,k-1)+(1-0.9995)*SRI_(k);
    end
    % Operaciones estáticas (la salida está en el mismo tiempo que la
entrada)
    % Datentint es Qo
    % A la salida del reactor 5 se divide el flujo, en este caso esta
parte
    % se dirige al decantador
    Salida_reactor(:,k) = [Zsal5(:,k); (1-SRI(1,k))*Qsal5(:,k)]; % Qf
    Qundf(k) = (1-SRI(1,k))*Qsal5(:,k)*(1-DF);
    Qeff(k) = (1-SRI(1,k))*Qsal5(:,k)*DF;
    % Para realizar los cálculos con el decantador se deben agrupar
los
    % componentes que sean particulados. También debe obtenerse la
    % proporción entre cada componente particulados y el agrupado.
    [Acceso_decantador(:,k), Fraccion_fango(:,k)] =
conversor_fango(Salida_reactor(:,k)); % Paso a fangos totales (14 a 9
variables)
    % SalDec_ se utiliza para mover las dimensiones de los datos
    % procedentes de la matriz 3D Estados_Decantador
```

```
    SalDec_{1}(:,k) = Estados_Decantador(:,1,k);
    SalDec_{2}(:,k) = Estados_Decantador(:,2,k);
    SalDec_{3}(:,k) = Estados_Decantador(:,3,k);
    SalDec_{4}(:,k) = Estados_Decantador(:,4,k);
    SalDec_{5}(:,k) = Estados_Decantador(:,5,k);
    SalDec_{6}(:,k) = Estados_Decantador(:,6,k);
    SalDec_{7}(:,k) = Estados_Decantador(:,7,k);
    SalDec_{8}(:,k) = Estados_Decantador(:,8,k);
    SalDec_{9}(:,k) = Estados_Decantador(:,9,k);
    SalDec_{10}(:,k) = Estados_Decantador(:,10,k);
    % Salida_inferior_decantador agrupa el vector de componentes con
    % particulados agrupados junto al caudal
    Salida_inferior_decantador(:,k) = [SalDec_{1}(:,k); Qundf(k)];
    % Underflow es el resultado de convertir el vector con
particulados
    % agrupados al vector general con cada componente individual. Para
ello
    % se utiliza el vector Fraccion_fango que contiene las
proporciones
    % obtenidas antes del decantador
    Underflow(:,k) = inversor_fango(Salida_inferior_decantador(:,k),
Fraccion_fango(:,k)); % Qu (9 a 14 variables)
    % Underflow se divide en dos flujos, Recirc_esterna y Wastage
    Recirc_externa(:,k) =[ones(1,13), SRE].*Underflow(:,k)'; % Qr
    Wastage(:,k) =[ones(1,13), (1-SRE)].*Underflow(:,k)'; % Qw
    Psi_w(:,k) = 0.75*sum(Wastage(3:7,k))*Wastage(14,k); % Masa de
fangos generada
    % La otra parte del flujo del reactor 5 se recircula
    Recirc_interna(:,k)=[Zsal5(:,k); SRI(1,k)*Qsal5(:,k)]; % Qa
    % Se suman la recirculación interna y la externa
    Suma_recirc(:,k) = sumacaudal(Recirc_interna(:,k),
Recirc_externa(:,k)); % Suma de ambas recirculaciones
    % Se suma el caudal de entrada y el de recirculación total
    % Opciones de recirculación
    % Opción de recircular tanto la recirculación interna como la
externa
    Acceso_reactor(:,k) = sumacaudal(Influent(:,k),Suma_recirc(:,k));
% Para cuando hay datos estables, Ts modificable

    % El flujo hacia la parte superior o effluent se obtiene juntando
los
    % componentes con el caudal, y realizando la conversión separando
los
    % componentes particulados
    Salida_superior_decantador(:,k) = [SalDec_{10}(:,k); Qeff(k)];
    Effluent(:,k) = inversor_fango(Salida_superior_decantador(:,k),
Fraccion_fango(:,k)); % Qe (9 a 14 variables)

    % Operaciones dinámicas (la salida es posterior en el tiempo a la
entrada)

%[Zsalr(:,k+1),Qsalr(:,k+1)]=depur_gen_disc(KLa(r,k),Zsal(r,k),Zsal(r-
1,k),Qsal(r-1,k),Ts,Vol)
    % Cada reactor recibe como entradas el coeficiente de
transferencia líquido-gas Kla aplicado,
    % el vector de componentes de salida del reactor en ese momento
(que será su composición en ese instante),
    % el vector de componentes procedente del reactor previo (que
corresponde al flujo de entrada a este reactor),
    % el caudal de entrada del reactor previo, las unidades de tiempo
transcurrido y el volumen del reactor.
```

```
    % Anóxicos

[Zsal1(:,k+1),Qsal1(:,k+1)]=rbio_gen_disc(KLa(1,k),Zsal1(:,k),Acceso_r
eactor(1:13,k),Acceso_reactor(14,k),Ts,1000); % Reactor biológico 1

[Zsal2(:,k+1),Qsal2(:,k+1)]=rbio_gen_disc(KLa(2,k),Zsal2(:,k),Zsal1(:,
k),Qsal1(:,k),Ts,1000); % Reactor biológico 2
    % Aerobios

[Zsal3(:,k+1),Qsal3(:,k+1)]=rbio_gen_disc(KLa(3,k),Zsal3(:,k),Zsal2(:,
k),Qsal2(:,k),Ts,1333); % Reactor biológico 3

[Zsal4(:,k+1),Qsal4(:,k+1)]=rbio_gen_disc(KLa(4,k),Zsal4(:,k),Zsal3(:,
k),Qsal3(:,k),Ts,1333); % Reactor biológico 4

[Zsal5(:,k+1),Qsal5(:,k+1)]=rbio_gen_disc(KLa(5,k),Zsal5(:,k),Zsal4(:,
k),Qsal4(:,k),Ts,1333); % Reactor biológico 5
    % El decantador tiene como entradas el flujo de acceso procedente
del
    % reactor, con los componentes particulados agrupados, el divisor
de
    % flujo hacia la parte superior e inferior del decantador, y la
unidad de tiempo transcurrido.
    % Decantador

    [Estados_Decantador(:,:,k+1), Qeff(k+1), Qundf(k+1)] =
clarif_disc(Acceso_decantador(:,k), DF, Ts,0);

end

% Última iteración para las operaciones estáticas
k = N
   SRI(k)=0.9995*SRI(k-1)+(1-0.9995)*SRI_(k);
Salida_reactor(:,k) = [Zsal5(:,k); (1-SRI(1,k))*Qsal5(:,k)];
[Acceso_decantador(:,k), Fraccion_fango(:,k)] =
conversor_fango(Salida_reactor(:,k));
SalDec_{1}(:,k) = Estados_Decantador(:,1,k);
SalDec_{2}(:,k) = Estados_Decantador(:,2,k);
SalDec_{3}(:,k) = Estados_Decantador(:,3,k);
SalDec_{4}(:,k) = Estados_Decantador(:,4,k);
SalDec_{5}(:,k) = Estados_Decantador(:,5,k);
SalDec_{6}(:,k) = Estados_Decantador(:,6,k);
SalDec_{7}(:,k) = Estados_Decantador(:,7,k);
SalDec_{8}(:,k) = Estados_Decantador(:,8,k);
SalDec_{9}(:,k) = Estados_Decantador(:,9,k);
SalDec_{10}(:,k) = Estados_Decantador(:,10,k);
Salida_inferior_decantador(:,k) = [SalDec_{1}(:,k); Qundf(k)];
Underflow(:,k) = inversor_fango(Salida_inferior_decantador(:,k),
Fraccion_fango(:,k));
Recirc_externa(:,k) =[1 1 1 1 1 1 1 1 1 1 1 1 1 SRE].*Underflow(:,k)';
Wastage(:,k) =[1 1 1 1 1 1 1 1 1 1 1 1 1 (1-SRE)].*Underflow(:,k)';
Recirc_interna(:,k)=[Zsal5(:,k); SRI(1,k)*Qsal5(:,k)];
Suma_recirc(:,k) = sumacaudal(Recirc_interna(:,k),
Recirc_externa(:,k));
Acceso_reactor(:,k) = sumacaudal(Influent(:,k),Suma_recirc(:,k));
Salida_superior_decantador(:,k) = [SalDec_{10}(:,k); Qeff(k)];
Effluent(:,k) = inversor_fango(Salida_superior_decantador(:,k),
Fraccion_fango(:,k));
Psi_w(:,k) = 0.75*sum(Wastage(3:7,k))*Wastage(14,k); % Masa de fangos
generada
```

A.3.3.- CÁLCULO DEL MODELO SIMPLIFICADO MEDIANTE REGRESORES

```
%% Cálculo por regresor
% Este programa permite distintos métodos de cálculo de regresores. La
% opción se selecciona desde el programa de inicialización de
variables.

tiempo=((1:N)*Ts);
abs_entradas_or=[randvect(1,1:decimation:N)'...
    randvect(2,1:decimation:N)',Influent(end,1:decimation:N)'];
%[u1,u2, u3]; %u3 reconvertido a valores de caudal real
as_salidas_or=sqrt([Effluent(10,1:decimation:N)',Psi_w(1,1:decimation:
N)']); %[y1,y2];

% Máximos (Valores de conversión)
u_max(1) = 1;%max(abs_entradas_or(:,1));
u_max(2) = 1;%max(abs_entradas_or(:,2));
u_max(3) = 1;%max(abs_entradas_or(:,3));
y_max(1) = 1;%max(abs_salidas_or(:,1));
y_max(2) = 1;%max(abs_salidas_or(:,2));

entradas_norm=[abs_entradas_or(:,1)/u_max(1),
abs_entradas_or(:,2)/u_max(2), abs_entradas_or(:,3)/u_max(3)];
salidas_norm=[abs_salidas_or(:,1)/y_max(1),
abs_salidas_or(:,2)/y_max(2)];
tini=int32(round(100/decimation)+1); %tini=14000;

% Conversor de incrementos a variables normalizadas (debe sumarse)
y_0(1) = 0*salidas_norm(tini,1);
y_0(2) = 0*salidas_norm(tini,2);
u_0(1) = 0*entradas_norm(tini,1);
u_0(2) = 0*entradas_norm(tini,2);
u_0(3) = 0*entradas_norm(tini,3);
% De aquí en adelante están todas normalizadas
inc_norm_entradas1=entradas_norm(tini:end,1)-u_0(1);
inc_norm_entradas2=entradas_norm(tini:end,2)-u_0(2);
inc_norm_entradas3=entradas_norm(tini:end,3)-u_0(3);
inc_norm_entradas=[inc_norm_entradas1,inc_norm_entradas2,
inc_norm_entradas3];
inc_norm_salidas1=salidas_norm(tini:end,1)-y_0(1);
inc_norm_salidas2=salidas_norm(tini:end,2)-y_0(2);
inc_norm_salidas=[inc_norm_salidas1,inc_norm_salidas2];
% A partir de aquí son incrementales sobre el valor normalizado
N2=N/decimation-tini+1; %Ajuste de +1 porque los vectores comienzan en
posición 1
if 1
    switch(regresor_tipo)
        case 0
            X1=zeros(N2-ordenregr,ordenregr*4+1);
            X2=zeros(N2-ordenregr,ordenregr*4);
            for i=1:N2-ordenregr
              X1(i,:)=[inc_norm_salidas(ordenregr+i-1:-1:1+i-1,1)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)',...
                    1];
              X2(i,:)=[inc_norm_salidas(ordenregr+i-1:-1:1+i-1,2)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
```

```
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
                    ];
            end

            % Theta1
            Y1=inc_norm_salidas(ordenregr+1:N2,1);
            theta1_old=X1\Y1;
            theta1_var=sdpvar(size(X1\Y1,1),size(X1\Y1,2),'full');
            Objetivo=sum((Y1-
X1*theta1_var).^2)+peso_var_th(1)*(sum((1/(ordenregr*4)*(theta1_var-
sum(theta1_var)/4/ordenregr).^2)));
            sol=optimize([0<=(X1*theta1_var+y_0(1))*y_max(1)<=15],
Objetivo);
            theta1=value(theta1_var);
            error1_r=(Y1-X1*theta1)'*(Y1-X1*theta1)

            % Theta2
            Y2=inc_norm_salidas(ordenregr+1:N2,2);
            theta2=X2\Y2;
            error2_r=(Y2-X2*theta2)'*(Y2-X2*theta2)

            %% Validación
            inc_salidas_pred=1*inc_norm_salidas; %Para inicializar
            for i=1:N2-ordenregr
                inc_salidas_pred(ordenregr+i,1)=[...
                    inc_salidas_pred(ordenregr+i-1:-1:1+i-1,1)'...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)',...
                    1]*theta1;

                inc_salidas_pred(ordenregr+i,2)=[...
                    inc_salidas_pred(ordenregr+i-1:-1:1+i-1,2)'...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
                    ]*theta2;
            end
            % Valores reales de las salidas (sumando los valores
iniciales)
            abs_salidas_pred_N = zeros(size(inc_salidas_pred));
            abs_salidas_pred_N(:,1) =(inc_salidas_pred(:,1)+
y_0(1))*y_max(1);
            abs_salidas_pred_N(:,2) =(inc_salidas_pred(:,2)+
y_0(2))*y_max(2);
            Real_a = Effluent(10,1:decimation:N)';
            Real_b = Psi_w(1,1:decimation:N)';
            Real_a2 = Real_a(tini:end);
            Real_b2 = Real_b(tini:end);

        case 1
            X1=zeros(N2-ordenregr,ordenregr*3+1);
            X2=zeros(N2-ordenregr,ordenregr*3);
            for i=1:N2-ordenregr
              X1(i,:)=[inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)'...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)'...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)' 1];
              X2(i,:)=[inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)'...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)'...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'];
```

```
            end

            % Theta1
            Y1=inc_norm_salidas(ordenregr+1:N2,1);
            theta1_old=X1\Y1;
            theta1_var=sdpvar(size(X1\Y1,1),size(X1\Y1,2),'full');
            Objetivo=sum((Y1-X1*theta1_var).^2);
            sol=optimize([0<=(X1*theta1_var+y_0(1))*y_max(1)],
Objetivo);
            theta1=value(theta1_var);
            error1_r=(Y1-X1*theta1)'*(Y1-X1*theta1)

            % Theta2
            Y2=inc_norm_salidas(ordenregr+1:N2,2);
            theta2_old=X2\Y2;
            theta2_var=sdpvar(size(X2\Y2,1),size(X2\Y2,2),'full');
            Objetivo=sum((Y2-
X2*theta2_var).^2)+peso_var_th(2)*(sum((1/(ordenregr*3)*(theta2_var-
sum(theta2_var)/3/ordenregr).^2)));
            sol=optimize([], Objetivo);
            theta2=value(theta2_var);
            error2_r=(Y2-X2*theta2)'*(Y2-X2*theta2)

            % Validación
            inc_salidas_pred=1*inc_norm_salidas; %Para inicializar
            for i=1:N2-ordenregr

inc_salidas_pred(ordenregr+i,1)=[inc_norm_entradas(ordenregr+i-1:-
1:1+i-1,1)'...
            inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)'...
            inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
            1 ]*theta1;

inc_salidas_pred(ordenregr+i,2)=[inc_norm_entradas(ordenregr+i-1:-
1:1+i-1,1)'...
            inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)'...
            inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
            ]*theta2;
            end

            % Valores reales de las salidas (sumando los valores
iniciales)
            abs_salidas_pred_N = zeros(size(inc_salidas_pred));
            abs_salidas_pred_N(:,1) =(inc_salidas_pred(:,1)+
y_0(1))*y_max(1);
            abs_salidas_pred_N(:,2) =(inc_salidas_pred(:,2)+
y_0(2))*y_max(2);
            Real_a = Effluent(10,1:decimation:N)';
            Real_b = Psi_w(1,1:decimation:N)';
            Real_a2 = Real_a(tini:end);
            Real_b2 = Real_b(tini:end);

        case 2
            X1=zeros(N2-ordenregr,ordenregr*4);
            X2=zeros(N2-ordenregr,ordenregr*4);
            for i=1:N2-ordenregr
              X1(i,:)=[inc_norm_salidas(ordenregr+i-1:-1:1+i-1,1)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
```

```
                ];
            X2(i,:)=[inc_norm_salidas(ordenregr+i-1:-1:1+i-1,2)',...
                inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
                inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
                inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
                ];
        end

        % Theta1
        Y1=inc_norm_salidas(ordenregr+1:N2,1);
        theta1_old=X1\Y1;
        theta1_var=sdpvar(size(X1\Y1,1),size(X1\Y1,2),'full');
        Objetivo=sum((Y1-
X1*theta1_var).^2)+peso_var_th(1)*(sum((1/(ordenregr*4)*(theta1_var-
sum(theta1_var)/4/ordenregr).^2)));

        %% Cálculo regresor por G(s)
        s=tf('s');  % s en dias
        Tcontrol_dias=1/24;
        G_am_KLa=-0.0090167*exp(-
0.0356497*s)/(1+0.1102481*s)/(1+0.0366734*s)/(1+0.0297943*s);
        G_am_KLa=G_am_KLa*u_max(1)/y_max(1);
        G_am_KLa_disc=c2d(G_am_KLa,Tcontrol_dias,'zoh');
        G_am_KLa_disc = absorbDelay(G_am_KLa_disc)
        [numG_am_KLa_disc,denG_am_KLa_disc] =
tfdata(G_am_KLa_disc,'v')
        figure
        step(G_am_KLa)
        hold on
        step(G_am_KLa_disc)
        theta_am_KLa=[-denG_am_KLa_disc(2:end),numG_am_KLa_disc];
        G_am_SRI=0.8123962*(1+0.5114600*s)*exp(-
0.0848674*s)/(1+0.1102481*s)/(1+0.0366734*s)/(1+0.0297943*s)/s;
        G_am_SRI=0.4147*exp(-
0.0848674*s)/(1+0.1102481*s)/(1+0.0366734*s)/(1+0.0297943*s);
        G_am_SRI=G_am_SRI*u_max(2)/y_max(1);
        G_am_SRI_disc=c2d(G_am_SRI,Tcontrol_dias,'zoh');
        G_am_SRI_disc = absorbDelay(G_am_SRI_disc)
        [numG_am_SRI_disc,denG_am_SRI_disc] =
tfdata(G_am_SRI_disc,'v')
        theta_am_SRI=[-denG_am_SRI_disc(2:end),numG_am_SRI_disc];
        theta_am_KLa=[-
denG_am_SRI_disc(2:end),conv(numG_am_KLa_disc,[1,-1])];

sol=optimize([0<=(X1*theta1_var+y_0(1))*y_max(1)<=15,theta1_var(ordenr
egr+1:2*ordenregr)<=0,theta1_var(2*ordenregr+1:end)>=0,...
            theta1_var(1:3*ordenregr)==[-denG_am_SRI_disc(2:end),
numG_am_KLa_disc, numG_am_SRI_disc]'...
            ], Objetivo);
        theta1Q=value(theta1_var);
        theta1_partQ=theta1Q(3*ordenregr+1:end);
        error1_r=(Y1-X1*theta1Q)'*(Y1-X1*theta1Q)
        theta1=[[-denG_am_SRI_disc(2:end), numG_am_KLa_disc,
numG_am_SRI_disc]' ;theta1_partQ];
        figure
        step(G_am_SRI,1)
        hold on
        step(G_am_SRI_disc)
        z=tf('z')
        Gtotal=tf(minreal([G_am_KLa_disc*(1-z^-1)/(1-z^-1)
G_am_SRI_disc]))
```

```
            conv(numG_am_SRI_disc,[1 -1])
            Y2=inc_norm_salidas(ordenregr+1:N2,2);
            theta2_old=X2\Y2;
            theta2_var=sdpvar(size(X2\Y2,1),size(X2\Y2,2),'full');
            Objetivo=sum((Y2-
X2*theta2_var).^2)+peso_var_th(2)*(sum((1/(ordenregr*3)*(theta2_var-
sum(theta2_var)/3/ordenregr).^2)));
            sol=optimize([], Objetivo);
            theta2=value(theta2_var);
            error2_r=(Y2-X2*theta2)'*(Y2-X2*theta2)


            inc_salidas_pred=1*inc_norm_salidas; %Para inicializar
            for i=1:N2-ordenregr
                inc_salidas_pred(ordenregr+i,1)=[...
                    inc_salidas_pred(ordenregr+i-1:-1:1+i-1,1)'...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
                    ]*theta1;

                inc_salidas_pred(ordenregr+i,2)=[...
                    inc_salidas_pred(ordenregr+i-1:-1:1+i-1,2)'...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
                    inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
                    ]*theta2;
            end
            % Valores reales de las salidas (sumando los valores
iniciales)
            abs_salidas_pred_N = zeros(size(inc_salidas_pred));
            abs_salidas_pred_N(:,1) =(inc_salidas_pred(:,1)+
y_0(1))*y_max(1);
            abs_salidas_pred_N(:,2) =(inc_salidas_pred(:,2)+
y_0(2))*y_max(2);
            Real_a = Effluent(10,1:decimation:N)';
            Real_b = Psi_w(1,1:decimation:N)';
            Real_a2 = Real_a(tini:end);
            Real_b2 = Real_b(tini:end);
    end

    %% Figuras

    figure('Name','R; Amonio-> Comparación real vs pred')
    plot(abs_salidas_pred_N(:,1),'--'); hold on;  plot(sqrt(Real_a2));
    legend('Predicción','Real'); title('Comparación entre la salida
real y la predicción');
    xlabel('Horas'); ylabel('S_NH (g N/m3)')
    error111=(abs_salidas_pred_N(:,1)-
Real_a2)'*(abs_salidas_pred_N(:,1)-Real_a2);

    figure('Name','R; Fangos-> Comparación real vs pred')
    plot(abs_salidas_pred_N(:,2),'--'); hold on;  plot(Real_b2(1:end-
1));
    legend('Predicción','Real')
    error222=(abs_salidas_pred_N(:,2)-
Real_b2)'*(abs_salidas_pred_N(:,2)-Real_b2);

end
return
```

```
%% Amonio respecto a KLa
s=tf('s');
G_am_KLa=-0.0090794*exp(-0.0366662*s)/(1+0.1013651*s)/(1+0.0729952*s)
*   u_max(1)/y_max(1);
%% Amonio respecto a SRI
G_am_SRI=1.0841671*exp(-0.0891836*s)/(1+0.4548187*s)/(1+0.0321804*s)
*   u_max(2)/y_max(1);
%% Amonio respecto a Q
G_am_Q=0.0010788*exp(-0.0581995*s)/(1+0.3948527*s)/(1+0.0171896*s)    *
u_max(3)/y_max(1);
Tcontrol_dias=1/24;
G_am_KLa_SRI_Q=[G_am_KLa G_am_SRI G_am_Q];
G_am_KLa_SRI_Q_disc=c2d(G_am_KLa_SRI_Q,Tcontrol_dias,'zoh');
G_am_KLa_SRI_Q_disc = absorbDelay(G_am_KLa_SRI_Q_disc);
G_am_KLa_SRI_Q_disc_SS=ss(G_am_KLa_SRI_Q_disc);
A=G_am_KLa_SRI_Q_disc_SS.a;
B=G_am_KLa_SRI_Q_disc_SS.b;
B1=B(:,[1 2]);
B2=B(:,3);
C=G_am_KLa_SRI_Q_disc_SS.c;

[num,den] = tfdata(G_am_KLa_SRI_Q_disc,'v');
dencomun=conv(conv(den{1},den{2}),den{3});
num1=conv(conv(num{1},den{2}),den{3});
num2=conv(conv(den{1},num{2}),den{3});
num3=conv(conv(den{1},den{2}),num{3});

theta1=[-dencomun(2:10), num1(2:10), num2(2:10), num3(2:10)]';
ordenregr=9;

X1=zeros(N2-ordenregr,ordenregr*4);
X2=zeros(N2-ordenregr,ordenregr*4);
for i=1:N2-ordenregr
    X1(i,:)=[inc_norm_salidas(ordenregr+i-1:-1:1+i-1,1)',...
        inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
        inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
        inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
        ];
end

% Theta1
Y1=inc_norm_salidas(ordenregr+1:N2,1);
theta1_old=X1\Y1;
error1_r=(Y1-X1*theta1)'*(Y1-X1*theta1)

inc_salidas_pred=1*inc_norm_salidas; %Para inicializar
for i=1:N2-ordenregr
    inc_salidas_pred(ordenregr+i,1)=[...
        inc_salidas_pred(ordenregr+i-1:-1:1+i-1,1)'...
        inc_norm_entradas(ordenregr+i-1:-1:1+i-1,1)',...
        inc_norm_entradas(ordenregr+i-1:-1:1+i-1,2)',...
        inc_norm_entradas(ordenregr+i-1:-1:1+i-1,3)'...
        ]*theta1;
end
% Valores reales de las salidas (sumando los valores iniciales)
abs_salidas_pred_N = zeros(size(inc_salidas_pred));
abs_salidas_pred_N(:,1) =(inc_salidas_pred(:,1)+ y_0(1))*y_max(1);
Real_a = Effluent(10,1:decimation:N)';
Real_b = Psi_w(1,1:decimation:N)';
```

```
Real_a2 = Real_a(tini:end);
Real_b2 = Real_b(tini:end);


figure('Name','R; Amonio-> Comparación real vs pred')
plot(abs_salidas_pred_N(:,1),'--'); hold on;  plot(Real_a2);
legend('Predicción','Real')
error111=(abs_salidas_pred_N(:,1)-Real_a2)'*(abs_salidas_pred_N(:,1)-
Real_a2);

theta2=0*theta1;
```

A.4.- ESTRUCTURA DEL ALGORITMO DE CONTROL PREDICTIVO

A.4.1.- DISEÑO DEL ALGORITMO DE CONTROL PREDICTIVO

```
%% Design_sistema_control_predictivo
% En este programa se inicializan las variables necesarias para
% realizar el diseño del control predictivo. Para ello, se llenan los
% vectores y las matrices, y se deja preparada la estructura del
% cálculo de la optimización.

%% Control predictivo - Multivariable - Dos salidas
% Diseño del método de optimización
yalmip('clear')
ref_t1 = [((refval1_t1/y_max(1))-y_0(1))*ones(1,refesc_t1),
((refval2_t1/y_max(1))-y_0(1))*ones(1,num_optimiz+horpred-refesc_t1)];
% Escalón
ref_t2 = [((refval1_t2/y_max(2))-y_0(2))*ones(1,refesc_t2),
((refval2_t2/y_max(2))-y_0(2))*ones(1,num_optimiz+horpred-refesc_t2)];
% Escalón
th1 = theta1(1:end-1);
th2 = theta2;
yexp_t1 = ((1.73/y_max(1))-y_0(1))*ones(ordenregr+1,1);
yexp_t2 = ((2461690/y_max(2))-y_0(2))*ones(ordenregr+1,1);
KLaexp = ((84/u_max(1))-u_0(1))*ones(ordenregr+1,1);
SRIexp = ((0.6/u_max(2))-u_0(2))*ones(ordenregr+1,1);
Qentexp =ones(ordenregr+1,1);

%% Se montan las matrices A y B
Monta_matrices_A_B

%%
uini=[0;0];
nu=2; % Número de u a calcular (número de entradas)

u = sdpvar(repmat(nu,1,horpred),repmat(1,1,horpred));
refvar_t1=sdpvar(horpred,1);
refvar_t2=sdpvar(horpred,1);
x0 = sdpvar(2*ordenregr+3*(ordenregr-1),1);
nx=2*ordenregr+3*(ordenregr-1);
x_llaves = sdpvar(repmat(nx,1,horpred+1),repmat(1,1,horpred+1));

precio_sdpvar=sdpvar(horpred,1);
u0 = sdpvar(nu,1);
```

```
delQfutura = sdpvar(horpred,1);
constraints = [];
constraints_light = [];
constraints_llaves = [];
objective = 0;
objective_light = 0;
objective_light_llaves=0;
x = x0;
yfut=sdpvar(horpred,1);
yfut_llaves=sdpvar(horpred,1);

correcc_estim_real_var_t1=sdpvar;
correcc_estim_real_var_t2=sdpvar;

for k = 1:horpred
    x = A*x + B1*u{k}+B2*delQfutura(k)+[theta1(end); zeros(length(A)-
1,1)];
    yfut(k)=x(1);
    yfut_llaves(k)=x_llaves{k+1}(1);
    objective =objective+precio_sdpvar(k)* (u{k}(1)+u_0(1))*u_max(1);
    objective_light =objective_light+norm(2-
((x(1)+correcc_estim_real_var_t1+y_0(1))*y_max(1)));%...
    if k==1
        constraints = [constraints, -500<=(u{k}(1)-u0(1))<=500, -
0.5<=(u{k}(2)-u0(2))<=0.5];
        constraints_light = [constraints_light, -500<=(u{k}(1)-
u0(1))<=500, -0.5<=(u{k}(2)-u0(2))<=0.5];
    else
        constraints = [constraints, -500<=(u{k}(1)-u{k-1}(1))<=500, -
0.5<=(u{k}(2)-u{k-1}(2))<=0.5];
        constraints_light = [constraints_light, -500<=(u{k}(1)-u{k-
1}(1))<=500, -0.5<=(u{k}(2)-u{k-1}(2))<=0.5];
    end
    constraints = [constraints, 0<=((u{k}(1)+u_0(1))*u_max(1))<=240,
0.1<=(u{k}(2) + u_0(2))*u_max(2)<=0.9];
    constraints_light = [constraints_light,
0<=((u{k}(1)+u_0(1))*u_max(1))<=240, 0.1<=(u{k}(2) +
u_0(2))*u_max(2)<=0.9];

    constraints = [constraints,
0<=((x(1)+correcc_estim_real_var_t1+y_0(1))*y_max(1))<=3];

ops = sdpsettings('verbose',0,'solver','MOSEK');  %MOSEK va, SEDUMI va
pero lento, CUTSDP va mediolento, fmincon lentísimo
controller = optimizer(constraints, objective, ops, [x0; u0;
delQfutura;refvar_t1;refvar_t2;correcc_estim_real_var_t1;correcc_estim
_real_var_t2;precio_sdpvar] ,u{1});

controller2 = optimizer(constraints, objective, ops, [x0; u0;
delQfutura;refvar_t1;refvar_t2;correcc_estim_real_var_t1;correcc_estim
_real_var_t2;precio_sdpvar],yfut);

controller3 = optimizer(constraints, objective, ops, [x0; u0;
delQfutura;refvar_t1;refvar_t2;correcc_estim_real_var_t1;correcc_estim
_real_var_t2;precio_sdpvar],u);

controller_light = optimizer(constraints_light, objective_light, ops,
[x0;u0;delQfutura;refvar_t1;refvar_t2;correcc_estim_real_var_t1;correc
c_estim_real_var_t2;precio_sdpvar],u{1});
```

```
controller_light_yfut = optimizer(constraints_light, objective_light,
ops,[x0;u0;delQfutura;refvar_t1;refvar_t2;correcc_estim_real_var_t1;co
rrecc_estim_real_var_t2;precio_sdpvar],yfut);

controller_llaves = optimizer(constraints_llaves, objective, ops,
[x_llaves{1};u0;delQfutura;refvar_t1;refvar_t2;correcc_estim_real_var_
t1;correcc_estim_real_var_t2;precio_sdpvar],u{1});
```

A.4.2.- MONTAJE DE MATRICES PROCEDENTES DE LOS CÁLCULOS DEL MODELO SIMPLIFICADO

```
%% Monta_matrices_A_B
% Esta función permite reorganizar los vectores theta en matrices A y
% B que permite que el optimizador realice mejor los cálculos del
% control predictivo

if ordenregr == 1
    A = 0;
    B1a = [th1(1*ordenregr+1);th2(1*ordenregr+1)];
    B1b = [th1(2*ordenregr+1);th2(2*ordenregr+1)];
    B2 = [th1(3*ordenregr+1);th2(3*ordenregr+1)];
else
    if (regresor_tipo ~= 1)
        A1 = [th1(1:ordenregr)';eye(ordenregr-1,ordenregr-
1),zeros(ordenregr-1,1);zeros(ordenregr);zeros(ordenregr-
1,ordenregr);zeros(ordenregr-1,ordenregr);zeros(ordenregr-
1,ordenregr)];
        A2 = [zeros(ordenregr);th2(1:ordenregr)';eye(ordenregr-
1,ordenregr-1),zeros(ordenregr-1,1);zeros(ordenregr-
1,ordenregr);zeros(ordenregr-1,ordenregr);zeros(ordenregr-
1,ordenregr)];
        A3 = [th1(ordenregr+2:2*ordenregr)';zeros(ordenregr-
1,ordenregr-1);th2(ordenregr+2:2*ordenregr)';zeros(ordenregr-
1,ordenregr-1);zeros(1,ordenregr-1);eye(ordenregr-2),zeros(ordenregr-
2,1);zeros(ordenregr-1,ordenregr-1);zeros(ordenregr-1,ordenregr-1)];
        A4 = [th1(2*ordenregr+2:3*ordenregr)';zeros(ordenregr-
1,ordenregr-1);th2(2*ordenregr+2:3*ordenregr)';zeros(ordenregr-
1,ordenregr-1);zeros(ordenregr-1,ordenregr-1);zeros(1,ordenregr-
1);eye(ordenregr-2),zeros(ordenregr-2,1);zeros(ordenregr-1,ordenregr-
1)];
        A5 = [th1(3*ordenregr+2:4*ordenregr)';zeros(ordenregr-
1,ordenregr-1);th2(3*ordenregr+2:4*ordenregr)';zeros(ordenregr-
1,ordenregr-1);zeros(ordenregr-1,ordenregr-1);zeros(ordenregr-
1,ordenregr-1);zeros(1,ordenregr-1);eye(ordenregr-2),zeros(ordenregr-
2,1)];
        A = [A1, A2, A3, A4, A5];

        B1a = [th1(1*ordenregr+1);zeros(ordenregr-
1,1);th2(1*ordenregr+1);zeros(ordenregr-1,1);1;zeros(ordenregr-
2,1);zeros(ordenregr-1,1);zeros(ordenregr-1,1)];
        B1b = [th1(2*ordenregr+1);zeros(ordenregr-
1,1);th2(2*ordenregr+1);zeros(ordenregr-1,1);zeros(ordenregr-
1,1);1;zeros(ordenregr-2,1);zeros(ordenregr-1,1)];
```

```
        B2 = [th1(3*ordenregr+1);zeros(ordenregr-
1,1);th2(3*ordenregr+1);zeros(ordenregr-1,1);zeros(ordenregr-
1,1);zeros(ordenregr-1,1);1;zeros(ordenregr-2,1)];


    elseif (regresor_tipo == 1)
        A1 = [zeros(1,ordenregr);eye(ordenregr-1,ordenregr-
1),zeros(ordenregr-1,1);zeros(ordenregr);zeros(ordenregr-
1,ordenregr);zeros(ordenregr-1,ordenregr);zeros(ordenregr-
1,ordenregr)];
        A2 = [zeros(ordenregr);zeros(1,ordenregr);eye(ordenregr-
1,ordenregr-1),zeros(ordenregr-1,1);zeros(ordenregr-
1,ordenregr);zeros(ordenregr-1,ordenregr);zeros(ordenregr-
1,ordenregr)];
        A3 = [th1(2:ordenregr)';zeros(ordenregr-1,ordenregr-
1);th2(2:ordenregr)';zeros(ordenregr-1,ordenregr-1);zeros(1,ordenregr-
1);eye(ordenregr-2),zeros(ordenregr-2,1);zeros(ordenregr-1,ordenregr-
1);zeros(ordenregr-1,ordenregr-1)];
        A4 = [th1(ordenregr+2:2*ordenregr)';zeros(ordenregr-
1,ordenregr-1);th2(ordenregr+2:2*ordenregr)';zeros(ordenregr-
1,ordenregr-1);zeros(ordenregr-1,ordenregr-1);zeros(1,ordenregr-
1);eye(ordenregr-2),zeros(ordenregr-2,1);zeros(ordenregr-1,ordenregr-
1)];
        A5 = [th1(2*ordenregr+2:3*ordenregr)';zeros(ordenregr-
1,ordenregr-1);th2(2*ordenregr+2:3*ordenregr)';zeros(ordenregr-
1,ordenregr-1);zeros(ordenregr-1,ordenregr-1);zeros(ordenregr-
1,ordenregr-1);zeros(1,ordenregr-1);eye(ordenregr-2),zeros(ordenregr-
2,1)];
        A = [A1, A2, A3, A4, A5];

        B1a = [th1(1);zeros(ordenregr-1,1);th2(1);zeros(ordenregr-
1,1);1;zeros(ordenregr-2,1);zeros(ordenregr-1,1);zeros(ordenregr-
1,1)];
        B1b = [th1(1*ordenregr+1);zeros(ordenregr-
1,1);th2(1*ordenregr+1);zeros(ordenregr-1,1);zeros(ordenregr-
1,1);1;zeros(ordenregr-2,1);zeros(ordenregr-1,1)];
        B2 = [th1(2*ordenregr+1);zeros(ordenregr-
1,1);th2(2*ordenregr+1);zeros(ordenregr-1,1);zeros(ordenregr-
1,1);zeros(ordenregr-1,1);1;zeros(ordenregr-2,1)];
    else
        disp('Formato de regresor no válido actualmente. Modifica y
vuelve a comenzar.')
        return
    end
end
B1 = [B1a, B1b];
B = [B1a, B1b, B2];
```

A.5.- APLICACIÓN DEL CONTROL PREDICTIVO

A.5.1.- SIMULACIÓN UTILIZANDO EL CONTROL PREDICTIVO

```
%% Aplicacion_control_predictivo
% En este programa se aplica el control predictivo. Primero se
inicializan
% las variables, y utilizando la estructura del control predictivo
diseñada
% en el apartado anterior, se realizan los cálculos de predicción
% aplicándo los resultados a una simulación del funcionamiento de la
EDAR.

%% Inicialización de las variables del control predictivo
Inicializa_variables_contr_pred_3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ups = 0;
% Es el influente normalizado e incrementado.
InfluQ = (Influent(end,1:decimation:60000)/u_max(3)-u_0(3));
%% Aplicación del control predictivo
correccion_estimacion_real_t1=0;
correccion_estimacion_real_t2=0;
precio_todo=zeros(num_optimiz,1);
precios_futuros=zeros(horpred,1)';
znormal=[];
for z = 1:min(num_optimiz,977)
    precio_todo(z)=calculaprecio(z,decimator_en_horas);
    for i=0:horpred-1
        precios_futuros(i+1)=calculaprecio(z+i,decimator_en_horas);
    end
    normval = [yexp_t1(end), ref_t1(z), yexp_t2(end), ref_t2(z)]
    xini = [yexp_t1(end:-1:(end-ordenregr+1)); yexp_t2(end:-1:(end-
ordenregr+1));KLaexp((end):-1:(end-ordenregr+2)); SRIexp((end):-
1:(end-ordenregr+2));Qentexp((end):-1:(end-ordenregr+2))];
    if z==1
        x_sim=xini;
    end
    uini=[KLaexp(end);SRIexp(end)];
    [uk,diagnostics] = controller{[xini;uini;InfluQ(z:z+horpred-
1)';ref_t1(z+1:z+horpred)';ref_t2(z+1:z+horpred)';0*correccion_estimac
ion_real_t1;correccion_estimacion_real_t2;precios_futuros'   ]};
    [yfutk,diagnostics] = controller2{[xini;uini;InfluQ(z:z+horpred-
1)';ref_t1(z+1:z+horpred)';ref_t2(z+1:z+horpred)';0*correccion_estimac
ion_real_t1;correccion_estimacion_real_t2;precios_futuros'   ]};
    [ufutk,diagnostics] = controller3{[xini;uini;InfluQ(z:z+horpred-
1)';ref_t1(z+1:z+horpred)';ref_t2(z+1:z+horpred)';0*correccion_estimac
ion_real_t1;correccion_estimacion_real_t2;precios_futuros'   ]};
    [uk_light,diagnostics_light] =
controller_light{[xini;uini;InfluQ(z:z+horpred-
1)';ref_t1(z+1:z+horpred)';ref_t2(z+1:z+horpred)';0*correccion_estimac
ion_real_t1;correccion_estimacion_real_t2;precios_futuros'   ]};
    [yfutk_light,diagnostics_light_yfut] =
controller_light_yfut{[xini;uini;InfluQ(z:z+horpred-
1)';ref_t1(z+1:z+horpred)';ref_t2(z+1:z+horpred)';0*correccion_estimac
ion_real_t1;correccion_estimacion_real_t2;precios_futuros'   ]};

    if ((diagnostics~=1)&&(~isnan(uk(1)))&&(~isnan(uk(2))))
        KLasolution = uk(1);
```

```
        SRIsolution = uk(2);
        znormal=[znormal;z];
    elseif
((diagnostics_light~=1)&&(~isnan(uk_light(1)))&&(~isnan(uk_light(2))))
        ups = ups+1;
        KLasolution = uk_light(1);
        SRIsolution = uk_light(2);
        yfutk=yfutk_light;
    else
        disp('Critical Error')
        pause
        return
    end
    if diagnostics~=0
    end
    numups(z) = ups;

    % Aquí se realiza el experimento relé
    if ((controlrele==1)&&(z>1))
        if (Effluent(10,v)>9)
            KLasolution=240;
        else
            KLasolution=0;
        end
        SRIsolution=0.6;
    end

    if (isnan(KLasolution)||isnan(SRIsolution))
        disp('Critical Error')
        pause
        return
    end

    xestimaunpaso = A*xini +
B1*[KLasolution;SRIsolution]+B2*InfluQ(z);
    yestimaunpaso_t1(z)=xestimaunpaso(1);
    yestimaunpaso_t2(z)=xestimaunpaso(ordenregr+1);
    vrecuerdo=v;
    figure(111)

    plot(z+1:z+horpred,yfutk','r');hold on;
    plot(z-ordenregr:z,yexp_t1(z:z+ordenregr),'k');hold off
    yfuthorpred(z+2)=yfutk(2);

    %% Avanza un paso en el proceso de simulación del comportamiento
de la depuradora usando el control predictivo
    Avanza_un_paso

    %%
    v=vrecuerdo+decimation;
    correccion_estimacion_real_t1=((Effluent(10,v)/y_max(1))-y_0(1))-
yestimaunpaso_t1(z)
    correccion_estimacion_real_t2=((Psi_w(1,v)/y_max(2))-y_0(2))-
yestimaunpaso_t2(z)
    yexp_t1 = [yexp_t1; ((sqrt(Effluent(10,v))/y_max(1))-y_0(1))];

    yexp_t2 = [yexp_t2; ((Psi_w(1,v)/y_max(2))-y_0(2))];
    KLaexp = [KLaexp;(KLaapl(1,v)/u_max(1))-u_0(1)];
    SRIexp = [SRIexp;(SRIapl(1,v)/u_max(2))-u_0(2)];
    Qentexp = [Qentexp;(Influent(end,v)/u_max(3))-u_0(3)];
```

```
    realval = [Effluent(10,v),Psi_w(1,v)/1000000, KLaapl(1,v),
SRIapl(1,v)]

end
```

A.5.2.- INICIALIZACIÓN DE LAS VARIABLES PARA LA SIMULACIÓN DEL CONTROL PREDICTIVO

```
%% Inicializa_variables_contr_pred_3
% En este programa se inicializan las distintas variables para el
% experimento del control predictivo

load Datos_estables_interpolados.mat
Influent_B = repelem(Datestint(2:end,:),1,f_muestreo);
% Influent = Influent_B;

% load Datos_entrada_interpolados.mat
% Influent_C = repelem(Datentint(2:end,:),1,f_muestreo);
% Influent = Influent_C;

load Datos_secos_interpolados.mat % Contiene los datos interpolados
para cada minuto en vez de para cada 15 min para la entrada real
Influent_D = 1.2*repelem(Datossecos(2:end,:),1,f_muestreo);

Influent = [Influent_B(1:(end-1),:);Influent_D(end,:)]; %
Concentraciones ctes

% Reserva de tamaño de los vectores
Zsal1 = zeros(13,num_muestras_experimento);  Zsal2 =
zeros(13,num_muestras_experimento);  Zsal3 =
zeros(13,num_muestras_experimento);  Zsal4 =
zeros(13,num_muestras_experimento);  Zsal5 =
zeros(13,num_muestras_experimento);
Qsal1 = zeros(1,num_muestras_experimento);   Qsal2 =
zeros(1,num_muestras_experimento);   Qsal3 =
zeros(1,num_muestras_experimento);   Qsal4 =
zeros(1,num_muestras_experimento);   Qsal5 =
zeros(1,num_muestras_experimento);
Recirc_interna = zeros(14,num_muestras_experimento);
Acceso_reactor = zeros(14,num_muestras_experimento); Salida_reactor =
zeros(14,num_muestras_experimento);
Acceso_decantador = zeros(9,num_muestras_experimento); Fraccion_fango
= zeros(6,num_muestras_experimento);
Effluent = zeros(14,num_muestras_experimento); Underflow =
zeros(14,num_muestras_experimento);
Recirc_externa = zeros(14,num_muestras_experimento); Wastage =
zeros(14,num_muestras_experimento);
Suma_recirc = zeros(14,num_muestras_experimento);
Qeff = zeros(1,num_muestras_experimento);
Qundf = zeros(1,num_muestras_experimento);
Estados_Decantador = zeros(8,10,num_muestras_experimento);
Salida_inferior_decantador = zeros(9,num_muestras_experimento);
Salida_superior_decantador = zeros(9,num_muestras_experimento);
SalDec_{1} = zeros(8,num_muestras_experimento);
SalDec_{2} = zeros(8,num_muestras_experimento);
```

```
SalDec_{3} = zeros(8,num_muestras_experimento);
SalDec_{4} = zeros(8,num_muestras_experimento);
SalDec_{5} = zeros(8,num_muestras_experimento);
SalDec_{6} = zeros(8,num_muestras_experimento);
SalDec_{7} = zeros(8,num_muestras_experimento);
SalDec_{8} = zeros(8,num_muestras_experimento);
SalDec_{9} = zeros(8,num_muestras_experimento);
SalDec_{10} = zeros(8,num_muestras_experimento);
Psi_w = zeros(1,num_muestras_experimento);

% Inicializaciones
% Valores iniciales del reactor
% Estos son los valores iniciales de los componentes de cada reactor
según el documento
Zsal1(:,1) = [30 2.81 1149 82.1 2552 148 449 0.00430 5.37 7.92 1.22
5.28 4.93]';
Zsal2(:,1) = [30 1.46 1149 76.4 2553 148 450 0.0000631 3.66 8.34 0.882
5.03 5.08]';
Zsal3(:,1) = [30 1.15 1149 64.9 2557 149 450 1.72 6.54 5.55 0.829 4.39
4.67]';
Zsal4(:,1) = [30 0.995 1149 55.7 2559 150 451 2.43 9.30 2.97 0.767
3.88 4.29]';
Zsal5(:,1) = [30 0.889 1149 49.3 2559 150 452 0.491 10.4 1.73 0.688
3.53 4.13]';
Qsal1(1) = 92230;
Qsal2(1) = 92230;
Qsal3(1) = 92230;
Qsal4(1) = 92230;
Qsal5(1) = 92230;

% Valores iniciales del decantador
Estados_Decantador(1,:,1) = ones(1,10)*30;
Estados_Decantador(2,:,1) = ones(1,10)*0.889;
Estados_Decantador(3,:,1) = ones(1,10)*0.491;
Estados_Decantador(4,:,1) = ones(1,10)*10.4;
Estados_Decantador(5,:,1) = ones(1,10)*1.73;
Estados_Decantador(6,:,1) = ones(1,10)*0.688;
Estados_Decantador(7,:,1) = ones(1,10)*4.13;
Estados_Decantador(8,:,1) = [6394 356 356 356 356 356 69.0 29.5 18.1
12.5];

% Modificar parámetros
SRE=0.9796; % Selector de Recirculación Externa
DF=0.4896; % Porcentaje de flujo de entrada al decantador que sale
como effluent

KLa = [zeros(2,num_muestras_experimento);
240*ones(2,num_muestras_experimento);zeros(1,num_muestras_experimento)
];
```

A.5.3.- SIMULACIÓN DE LA EDAR EN CADA ITERACIÓN DEL CONTROL PREDICTIVO

```
%% Avanza_un_paso
% Este programa simula el comportamiento de la EDAR en cada iteracion
del
% control predictivo
for w = 1:decimation
    v = v+1;
    KLaapl(1,v) = (KLasolution + u_0(1))*u_max(1);
    SRIapl_(1,v) = (SRIsolution + u_0(2))*u_max(2);

    if (KLaapl(1,v)>240)
        KLaapl(1,v) = 240;
    end
    if (KLaapl(1,v)<0)
        KLaapl(1,v) = 0;
    end
    if (SRIapl_(1,v)>1)
        SRIapl_(1,v) = 1;
    end
    if (SRIapl_(1,v)<0)
        SRIapl_(1,v) = 0;
    end

    KLa(5,v) = KLaapl(1,v);
    if ~exist('SRIapl')
        SRIapl=0.6*ones(1,v);
    end
    if v==1
        SRIapl(v)=0.9995*0.6+(1-0.9995)*SRIapl_(v);
    else
        SRIapl(v)=0.9995*SRIapl(v-1)+(1-0.9995)*SRIapl_(v);
    end

    % Operaciones estáticas (la salida está en el mismo tiempo que la
entrada)
    Salida_reactor(:,v) = [Zsal5(:,v); (1-SRIapl(1,v))*Qsal5(:,v)]; %
Qf
    Qundf(v) = (1-SRIapl(1,v))*Qsal5(:,v)*(1-DF);
    Qeff(v) = (1-SRIapl(1,v))*Qsal5(:,v)*DF;
    [Acceso_decantador(:,v), Fraccion_fango(:,v)] =
conversor_fango(Salida_reactor(:,v));
    SalDec_{1}(:,v) = Estados_Decantador(:,1,v);
    SalDec_{2}(:,v) = Estados_Decantador(:,2,v);
    SalDec_{3}(:,v) = Estados_Decantador(:,3,v);
    SalDec_{4}(:,v) = Estados_Decantador(:,4,v);
    SalDec_{5}(:,v) = Estados_Decantador(:,5,v);
    SalDec_{6}(:,v) = Estados_Decantador(:,6,v);
    SalDec_{7}(:,v) = Estados_Decantador(:,7,v);
    SalDec_{8}(:,v) = Estados_Decantador(:,8,v);
    SalDec_{9}(:,v) = Estados_Decantador(:,9,v);
    SalDec_{10}(:,v) = Estados_Decantador(:,10,v);
    Salida_inferior_decantador(:,v) = [SalDec_{1}(:,v); Qundf(v)];
    Underflow(:,v) = inversor_fango(Salida_inferior_decantador(:,v),
Fraccion_fango(:,v));
    Recirc_externa(:,v) =[ones(1,13), SRE].*Underflow(:,v)';
    Wastage(:,v) =[ones(1,13), (1-SRE)].*Underflow(:,v)';
    Psi_w(:,v) = 0.75*sum(Wastage(3:7,v))*Wastage(14,v);
    Recirc_interna(:,v)=[Zsal5(:,v); SRIapl(1,v)*Qsal5(:,v)];
```

```
    Suma_recirc(:,v) = sumacaudal(Recirc_interna(:,v),
Recirc_externa(:,v));
    %         Acceso_reactor(:,v) =
sumacaudal(Influent(:,v),Suma_recirc(:,v));
    Acceso_reactor(:,v) = sumacaudal(Influent(:,v),Suma_recirc(:,v));
    Salida_superior_decantador(:,v) = [SalDec_{10}(:,v); Qeff(v)];
    Effluent(:,v) = inversor_fango(Salida_superior_decantador(:,v),
Fraccion_fango(:,v));

    % Operaciones dinámicas (la salida es posterior en el tiempo a la
entrada)
    % Anóxicos

[Zsal1(:,v+1),Qsal1(:,v+1)]=rbio_gen_disc(KLa(1,v),Zsal1(:,v),Acceso_r
eactor(1:13,v),Acceso_reactor(14,v),Ts,1000); % Reactor biológico 1

[Zsal2(:,v+1),Qsal2(:,v+1)]=rbio_gen_disc(KLa(2,v),Zsal2(:,v),Zsal1(:,
v),Qsal1(:,v),Ts,1000); % Reactor biológico 2
    % Aerobios

[Zsal3(:,v+1),Qsal3(:,v+1)]=rbio_gen_disc(KLa(3,v),Zsal3(:,v),Zsal2(:,
v),Qsal2(:,v),Ts,1333); % Reactor biológico 3

[Zsal4(:,v+1),Qsal4(:,v+1)]=rbio_gen_disc(KLa(4,v),Zsal4(:,v),Zsal3(:,
v),Qsal3(:,v),Ts,1333); % Reactor biológico 4

[Zsal5(:,v+1),Qsal5(:,v+1)]=rbio_gen_disc(KLa(5,v),Zsal5(:,v),Zsal4(:,
v),Qsal4(:,v),Ts,1333); % Reactor biológico 5
    % Decantador
    if v==1
        [Estados_Decantador(:,:,v+1), Qeff(v+1), Qundf(v+1)] =
clarif_disc(Acceso_decantador(:,v), DF, Ts,1);
    else
        [Estados_Decantador(:,:,v+1), Qeff(v+1), Qundf(v+1)] =
clarif_disc(Acceso_decantador(:,v), DF, Ts,0);
    end
end
```

A.5.4.- FUNCIÓN PARA DETERMINAR EL PRECIO DE LA ENERGÍA

```
%% Función calcula_precio
% Esta función asigna precios en función del día
% Ampliable a que sea variable en función de la semana
function precio=calculaprecio(z,decimator_en_horas)
tiempodia=mod(round(z*decimator_en_horas),24);
switch tiempodia
    case 0
        precio=0.087564;
    case 1
        precio=0.087564;
    case 2
        precio=0.087564;
    case 3
        precio=0.087564;
    case 4
        precio=0.087564;
    case 5
```

```
        precio=0.087564;
    case 6
        precio=0.087564;
    case 7
        precio=0.087564;
    case 8
        precio=0.130477;
    case 9
        precio=0.130477;
    case 10
        precio=0.146984;
    case 11
        precio=0.146984;
    case 12
        precio=0.146984;
    case 13
        precio=0.130477;
    case 14
        precio=0.130477;
    case 15
        precio=0.130477;
    case 16
        precio=0.130477;
    case 17
        precio=0.130477;
    case 18
        precio=0.146984;
    case 19
        precio=0.146984;
    case 20
        precio=0.146984;
    case 21
        precio=0.130477;
    case 22
        precio=0.130477;
    case 23
        precio=0.130477;

end
```

A.6.- GRÁFICAS

```
%% Graficas_control_predictivo
% Gráficas varias para observar el comportamiento del control

zultima=z;
percentups = ups/num_optimiz*100

figure('Name','Amonio controlado'); plot((yexp_t1+y_0(1))*y_max(1));
hold on; plot((ref_t1+y_0(1))*y_max(1),'r')
legend('Salida del amonio en iteraciones controladas','Referencia');
title('Seguimiento de la referencia del amonio')

figure('Name','Fango controlado'); plot((yexp_t2+y_0(2))*y_max(2));
hold on; plot((ref_t2+y_0(2))*y_max(2),'r')
```

```
legend('Salida del fango en iteraciones controladas','Referencia');
title('Seguimiento de la referencia del fango')

figure('Name', 'Valores del amonio en distintos puntos de la
depuradora');
plot(Effluent(10,:)'); hold on; plot(Zsal5(10,:)','--
');plot(Underflow(10,:)','*')
legend('Effluent','Salida Reactor 5','Underflow'); title('Amonio
en...')

figure('Name','Entradas a la
depuradora');plot((KLaexp+u_0(1))*u_max(1));hold
on;plot((SRIexp+u_0(2))*u_max(2)*100)
legend('KLaexp','SRIexp'); title('Entradas a la depuradora')

Amon = Influent(10,1:decimation:N);
figure('Name',
'Amonio');plot((yexp_t1(ordenregr:end)+y_0(1))*y_max(1));hold
on;plot(Influent(end,1:decimation:end)/10e3);
legend('Salida','Entrada')

Amon = Influent(10,1:decimation:N);
figure('Name',
'Amonio');plot((yexp_t1(ordenregr:end).^2+y_0(1))*y_max(1));hold
on;plot(Influent(end,1:decimation:end)/10e3);
legend('Salida','Entrada')

figure('Name','Número de fallos, en porcentaje, del optimizador')
plot(numups/num_optimiz*100)

figure('Name','Precio y Entradas a la
depuradora');plot((KLaexp+u_0(1))*u_max(1));hold
on;plot((SRIexp+u_0(2))*u_max(2)*100);plot(precio_todo*1000)
legend('KLaexp','SRIexp'); title('Entradas a la depuradora')


Undia=24/decimator_en_horas; %muestras
coste_dia=0;
amoniomedio_dia=0;
coste_total=0;
for z = 1:zultima
    if mod(z,Undia)==0
        coste_total=coste_total+sum(precio_todo(z:-1:z-
24/decimator_en_horas+1).*((KLaexp(z:-1:z-
24/decimator_en_horas+1)+u_0(1))*u_max(1)))*decimator_en_horas;
        coste_dia=[coste_dia; sum(precio_todo(z:-1:z-
24/decimator_en_horas+1).*((KLaexp(z:-1:z-
24/decimator_en_horas+1)+u_0(1))*u_max(1)))*decimator_en_horas];
        amoniomedio_dia=[amoniomedio_dia;    sum((yexp_t1(z:-1:z-
24/decimator_en_horas+1)+y_0(1))*y_max(1))/(24/decimator_en_horas)];
    else
        coste_dia=[coste_dia;coste_dia(end)];
        amoniomedio_dia=[amoniomedio_dia;amoniomedio_dia(end)];
    end
end
coste_total

figure('Name',
'Amonio');plot((yexp_t1(ordenregr:end)+y_0(1))*y_max(1));hold
on;plot(amoniomedio_dia);
```

```
legend('Salida','media')

figure('Name','Gasto entradas a la
depuradora');plot((KLaexp+u_0(1))*u_max(1));hold
on;plot((SRIexp+u_0(2))*u_max(2)*100);plot(precio_todo*1000)
plot(coste_dia/10)

plot(znormal,(KLaexp(znormal)+u_0(1))*u_max(1),'.');hold
on;plot(znormal,(SRIexp(znormal)+u_0(2))*u_max(2)*100,'.');

legend('KLaexp','SRIexp'); title('Entradas a la depuradora')

figure,plot(yfuthorpred)
hold on
plot(yexp_t1(ordenregr+1:end),'.'),legend('pred','real')

figure,plot(yfuthorpred.^2)
hold on
plot(yexp_t1(ordenregr+1:end).^2,'.'),legend('pred','real')
```

A.7.- AYUDAS AL DISEÑO DEL ALGORITMO

A.7.1.- CREACIÓN DEL VECTOR ALEATORIO PARA EL CÁLCULO POR REGRESORES

```
%% Crea_vector_aleatorio
% En este programa se crea el vector aleatorio para excitar al sistema
% para calcular el regresor. Se pueden realizar cambios cada minuto

clear all
minutscanvi=60;
tactual_min = 0;
vrandom(:,1) = [0;84;0.6;1.0];
i=0;

while (tactual_min<42*24*60)
    i=i+1;
    for j = 0:2
        3*i+j;
        vrandom(:,3*i+j-1) = vrandom(:,3*i+j-2); % Se copian los
valores anteriores
        % Se determina el siguiente valor de tiempo (en minutos)
        tprox_min = tactual_min + minutscanvi +
round(2*rand)*minutscanvi;
        vrandom(1,3*i+j-1) = tprox_min; % Se introduce dicho valor de
tiempo

        switch j % Cada vez cambia uno de los valores
            case 0
                vrandom(j+2, 3*i+j-1) = 0+rand*(240-0);  %KLA
            case 1
                vrandom(j+2, 3*i+j-1) = 0.1+rand*(0.9-0.1);%
vrandom(3,1) + 0.1*(2*rand-1); %SRI
            case 2
```

```
                vrandom(j+2, 3*i+j-1) = 0.3+rand*(1.8-0.3);%
vrandom(4,1) + 0.1*(2*rand-1);  %Q
        end

        tactual_min = tprox_min;
    end
end

vrandom;
[m,n]=size(vrandom); % Se determina el tamaño del vector original
vectal=zeros(m,vrandom(1,n));
for q = 1:n
    if mod((n-q), 100) == 0
        n-q
    end

    vrandtemp = vrandom(1,q); % Se traspasa el vector de tiempos a un
nuevo vector
    vectal(:,vrandtemp+1) = vrandom(:,q); % Se copian los valores en
la nueva posición, que es la que corresponde al tiempo con los huecos
intermedios
    for w = (vrandtemp+2):(vrandom(1,n)+1)
        vectal(2:end,w) = vectal(2:end,vrandtemp+1); % Se copian los
valores de la nueva posición al resto del vector,
        % para rellenar el vacío entre tiempos
    end
    vectal(1,:) = (0:1:vrandom(:,n))/24/60; % Se pasa de minutos a
días
end
vectal;
save('vectal_minuto.mat', 'vectal')
figure;
plot(vectal(1,:), (vectal(2,:)/84),vectal(1,:), (vectal(3,:)+0.4),
vectal(1,:), vectal(4,:))
```

A.7.2.- MODIFICACIONES DEL VECTOR DE CAUDAL AFLUENTE

```
%% Crea vector inicial
% En este programa se modifican los datos del Benchmark para obtener
un
% vector más largo, o con más datos interpolados

% Juntar datos
load('Inf_dry_2006_N.txt')
Datostotales=Inf_dry_2006_N';

Datos2=Inf_dry_2006_N';
Datos2(1,:)=Datos2(1,:)+Datostotales(1,end)+0.25/24;
Datostotales=[Datostotales Datos2];

Datos3=Inf_dry_2006_N';
Datos3(1,:)=Datos3(1,:)+Datostotales(1,end)+0.25/24;
Datostotales=[Datostotales Datos3];

% Interpolar datos
torig=Datostotales(1,:);
```

```
Ts=1/60/24; %Ts=1min;
t=(0:Ts:42)';
Datossecos(1,:)=t;
for i=2:15
  Datossecos(i,:)= interp1(torig,Datostotales(i,:),t);
end
% Datentintspl(1,:)=t;
% for i=2:15
%   Datentintspl(i,:)= spline(torig,Datostotales(i,:),t);
% end

save('Datos_secos_interpolados.mat', 'Datossecos');
```

A.7.3.- EXPERIMENTO DE ENTRADAS ESCALÓN

```
%% Experimento_escalon
% Este experimento aplica escalones entrada al sistema para su
posterior
% identificación

% Preparativos
close all
clear all
clc

% Intervalos de tiempo
tdiv = 1;
N = 2*60000*tdiv;
Ts = 1/60/24/tdiv;
guarda_datos = zeros(6,N);


% Selección de datos de entrada
load Datos_estables_interpolados.mat % Contiene los datos interpolados
para cada minuto en vez de para cada 15 min para la entrada estable
Influent_B = repelem(Datestint(2:end,:),1,2*tdiv);
M = size(Influent_B,2);

for j = 1:3
    vect_entrada = [84*ones(1,M);0.6*ones(1,M);ones(1,M)];
    tescalon = round(0.4*M);
    switch j
        case 1
            vect_entrada(1,:) = [84*ones(1,tescalon), 20*ones(1,M-
tescalon)]; %KLa
        case 2
            vect_entrada(2,:) = [0.6*ones(1,tescalon), 0.45*ones(1,M-
tescalon)]; %Valvula
        case 3
            vect_entrada(3,:) = [1*ones(1,tescalon), 0.8*ones(1,M-
tescalon)]; %Factor de caudal
    end
    vector_rep = vect_entrada;
    % Multiplicamos el caudal de entrada por la ponderación calculada
```

```
    Influent = Influent_B.*[ones(13,
2*60481*tdiv);vector_rep(3,1:60481*2*tdiv)];
    % Valores de KLa
    KLa = [zeros(2,N);240*ones(2,N);vector_rep(1,1:N)];
    % Reserva de tamaño de los vectores
    % ZsalX es el vector que almacena las variables de
estado/componentes de las aguas residuales en cada unidad de tiempo
    % QsalX es el vector que almacena los valores de caudal
    Zsal1 = zeros(13,N);  Zsal2 = zeros(13,N);  Zsal3 = zeros(13,N);
Zsal4 = zeros(13,N);  Zsal5 = zeros(13,N);
    Qsal1 = zeros(1,N);   Qsal2 = zeros(1,N);   Qsal3 = zeros(1,N);
Qsal4 = zeros(1,N);   Qsal5 = zeros(1,N);
    % Los vectores con dimensión 14,N contienen los componentes y el
caudal
    % Acceso_decantador tiene los componentes solubles, los
componentes particulados agrupados y el caudal
    % Fraccion_fango tiene las fracciones correspondientes a los
particulados
    % respecto a la componente total particulada
    Recirc_interna = zeros(14,N);
    Acceso_reactor = zeros(14,N); Salida_reactor = zeros(14,N);
    Acceso_decantador = zeros(9,N); Fraccion_fango = zeros(6,N);
    Effluent = zeros(14,N); Underflow = zeros(14,N);
    Recirc_externa = zeros(14,N); Wastage = zeros(14,N);
    Suma_recirc = zeros(14,N);
    Qeff = zeros(1,N);
    Qundf = zeros(1,N);
    % Los valores de los 8 estados de cada capa (7 solubles + 1
particulado
    % total) se encuentran en la variable Estados_Decantador
    Estados_Decantador = zeros(8,10,N);
    Salida_inferior_decantador = zeros(9,N);
    Salida_superior_decantador = zeros(9,N);
    % SalDec_ se utiliza para mover los ejes de las dimensiones de la
matriz Estados_Decantador
    SalDec_{1} = zeros(8,N);
    SalDec_{2} = zeros(8,N);
    SalDec_{3} = zeros(8,N);
    SalDec_{4} = zeros(8,N);
    SalDec_{5} = zeros(8,N);
    SalDec_{6} = zeros(8,N);
    SalDec_{7} = zeros(8,N);
    SalDec_{8} = zeros(8,N);
    SalDec_{9} = zeros(8,N);
    SalDec_{10} = zeros(8,N);
    Psi_w = zeros(1,N);
    % Inicializaciones
    % Valores iniciales del reactor
    % Estos son los valores iniciales de los componentes de cada
reactor según el documento
    Zsal1(:,1) = [30 2.81 1149 82.1 2552 148 449 0.00430 5.37 7.92
1.22 5.28 4.93]';
    Zsal2(:,1) = [30 1.46 1149 76.4 2553 148 450 0.0000631 3.66 8.34
0.882 5.03 5.08]';
    Zsal3(:,1) = [30 1.15 1149 64.9 2557 149 450 1.72 6.54 5.55 0.829
4.39 4.67]';
    Zsal4(:,1) = [30 0.995 1149 55.7 2559 150 451 2.43 9.30 2.97 0.767
3.88 4.29]';
    Zsal5(:,1) = [30 0.889 1149 49.3 2559 150 452 0.491 10.4 1.73
0.688 3.53 4.13]';
    Qsal1(1) = 92230;
```

```
    Qsal2(1) = 92230;
    Qsal3(1) = 92230;
    Qsal4(1) = 92230;
    Qsal5(1) = 92230;
    % Valores iniciales del decantador
    Estados_Decantador(1,:,1) = ones(1,10)*30;
    Estados_Decantador(2,:,1) = ones(1,10)*0.889;
    Estados_Decantador(3,:,1) = ones(1,10)*0.491;
    Estados_Decantador(4,:,1) = ones(1,10)*10.4;
    Estados_Decantador(5,:,1) = ones(1,10)*1.73;
    Estados_Decantador(6,:,1) = ones(1,10)*0.688;
    Estados_Decantador(7,:,1) = ones(1,10)*4.13;
    Estados_Decantador(8,:,1) = [6394 356 356 356 356 356 69.0 29.5
18.1 12.5];
    % Modificar parámetros
    % SRI=0.6; % Selector de Recirculación Interna
    SRE=0.9796; % Selector de Recirculación Externa
    DF=0.4896; % Porcentaje de flujo de entrada al decantador que sale
como effluent
    SRI = vector_rep(2,1:N);
    % Iteraciones
    for k=1:N-1
        % Operaciones estáticas (la salida está en el mismo tiempo que
la entrada)
        if mod(k, 1000) == 0
            k
        end
        % Se filtra la válvula
        if k==1
            SRI(1,k)=0.9995*0.6+(1-0.9995)*vector_rep(2,k);
        else
            SRI(1,k)=0.9995*SRI(1,k-1)+(1-0.9995)*vector_rep(2,k);
        end
        % Datentint es Qo
        % A la salida del reactor 5 se divide el flujo, en este caso
esta parte
        % se dirige al decantador
        Salida_reactor(:,k) = [Zsal5(:,k); (1-SRI(1,k))*Qsal5(:,k)]; %
Qf
        Qundf(k) = (1-SRI(1,k))*Qsal5(:,k)*(1-DF);
        Qeff(k) = (1-SRI(1,k))*Qsal5(:,k)*DF;
        % Para realizar los cálculos con el decantador se deben
agrupar los
        % componentes que sean particulados. También debe obtenerse la
        % proporción entre cada componente particulados y el agrupado.
        [Acceso_decantador(:,k), Fraccion_fango(:,k)] =
conversor_fango(Salida_reactor(:,k)); % Paso a fangos totales (14 a 9
variables)
        % SalDec_ se utiliza para mover las dimensiones de los datos
        % procedentes de la matriz 3D Estados_Decantador
        SalDec_{1}(:,k) = Estados_Decantador(:,1,k);
        SalDec_{2}(:,k) = Estados_Decantador(:,2,k);
        SalDec_{3}(:,k) = Estados_Decantador(:,3,k);
        SalDec_{4}(:,k) = Estados_Decantador(:,4,k);
        SalDec_{5}(:,k) = Estados_Decantador(:,5,k);
        SalDec_{6}(:,k) = Estados_Decantador(:,6,k);
        SalDec_{7}(:,k) = Estados_Decantador(:,7,k);
        SalDec_{8}(:,k) = Estados_Decantador(:,8,k);
        SalDec_{9}(:,k) = Estados_Decantador(:,9,k);
        SalDec_{10}(:,k) = Estados_Decantador(:,10,k);
```

```
        % Salida_inferior_decantador agrupa el vector de componentes
con
        % particulados agrupados junto al caudal
        Salida_inferior_decantador(:,k) = [SalDec_{1}(:,k); Qundf(k)];
        % Underflow es el resultado de convertir el vector con
particulados
        % agrupados al vector general con cada componente individual.
Para ello
        % se utiliza el vector Fraccion_fango que contiene las
proporciones
        % obtenidas antes del decantador
        Underflow(:,k) =
inversor_fango(Salida_inferior_decantador(:,k), Fraccion_fango(:,k));
% Qu (9 a 14 variables)
        % Underflow se divide en dos flujos, Recirc_esterna y Wastage
        Recirc_externa(:,k) =[ones(1,13), SRE].*Underflow(:,k)'; % Qr
        Wastage(:,k) =[ones(1,13), (1-SRE)].*Underflow(:,k)'; % Qw
        Psi_w(:,k) = 0.75*sum(Wastage(3:7,k))*Wastage(14,k); % Masa de
fangos generada
        % La otra parte del flujo del reactor 5 se recircula
        Recirc_interna(:,k)=[Zsal5(:,k); SRI(1,k)*Qsal5(:,k)]; % Qa
        % Se suman la recirculación interna y la externa
        Suma_recirc(:,k) = sumacaudal(Recirc_interna(:,k),
Recirc_externa(:,k)); % Suma de ambas recirculaciones
        % Se suma el caudal de entrada y el de recirculación total
        % Opciones de recirculación
        % Opción de recircular tanto la recirculación interna como la
externa
        Acceso_reactor(:,k) =
sumacaudal(Influent(:,k),Suma_recirc(:,k)); % Para cuando hay datos
estables, Ts modificable
        % El flujo hacia la parte superior o effluent se obtiene
juntando los
        % componentes con el caudal, y realizando la conversión
separando los
        % componentes particulados
        Salida_superior_decantador(:,k) = [SalDec_{10}(:,k); Qeff(k)];
        Effluent(:,k) =
inversor_fango(Salida_superior_decantador(:,k), Fraccion_fango(:,k));
% Qe (9 a 14 variables)


        % Operaciones dinámicas (la salida es posterior en el tiempo a
la entrada)

%[Zsalr(:,k+1),Qsalr(:,k+1)]=depur_gen_disc(KLa(r,k),Zsal(r,k),Zsal(r-
1,k),Qsal(r-1,k),Ts,Vol)
        % Cada reactor recibe como entradas el coeficiente de
transferencia líquido-gas Kla aplicado,
        % el vector de componentes de salida del reactor en ese
momento (que será su composición en ese instante),
        % el vector de componentes procedente del reactor previo (que
corresponde al flujo de entrada a este reactor),
        % el caudal de entrada del reactor previo, las unidades de
tiempo transcurrido y el volumen del reactor.
        % Anóxicos

[Zsal1(:,k+1),Qsal1(:,k+1)]=rbio_gen_disc(KLa(1,k),Zsal1(:,k),Acceso_r
eactor(1:13,k),Acceso_reactor(14,k),Ts,1000); % Reactor biológico 1

[Zsal2(:,k+1),Qsal2(:,k+1)]=rbio_gen_disc(KLa(2,k),Zsal2(:,k),Zsal1(:,
k),Qsal1(:,k),Ts,1000); % Reactor biológico 2
```

```
        % Aerobios

[Zsal3(:,k+1),Qsal3(:,k+1)]=rbio_gen_disc(KLa(3,k),Zsal3(:,k),Zsal2(:,
k),Qsal2(:,k),Ts,1333); % Reactor biológico 3

[Zsal4(:,k+1),Qsal4(:,k+1)]=rbio_gen_disc(KLa(4,k),Zsal4(:,k),Zsal3(:,
k),Qsal3(:,k),Ts,1333); % Reactor biológico 4

[Zsal5(:,k+1),Qsal5(:,k+1)]=rbio_gen_disc(KLa(5,k),Zsal5(:,k),Zsal4(:,
k),Qsal4(:,k),Ts,1333); % Reactor biológico 5
        % El decantador tiene como entradas el flujo de acceso
procedente del reactor, con los componentes particulados agrupados, el
divisor de flujo hacia la parte superior e inferior del decantador, y
la unidad de tiempo transcurrido.
        % Decantador
        [Estados_Decantador(:,:,k+1), Qeff(k+1), Qundf(k+1)] =
clarif_disc_Z(Acceso_decantador(:,k), DF, Ts);

    end
    % Última iteración para las operaciones estáticas
    k = N
    SRI(1,k)=0.9995*SRI(1,k-1)+(1-0.9995)*vector_rep(2,k);
    Salida_reactor(:,k) = [Zsal5(:,k); (1-SRI(1,k))*Qsal5(:,k)];
    [Acceso_decantador(:,k), Fraccion_fango(:,k)] =
conversor_fango(Salida_reactor(:,k));
    SalDec_{1}(:,k) = Estados_Decantador(:,1,k);
    SalDec_{2}(:,k) = Estados_Decantador(:,2,k);
    SalDec_{3}(:,k) = Estados_Decantador(:,3,k);
    SalDec_{4}(:,k) = Estados_Decantador(:,4,k);
    SalDec_{5}(:,k) = Estados_Decantador(:,5,k);
    SalDec_{6}(:,k) = Estados_Decantador(:,6,k);
    SalDec_{7}(:,k) = Estados_Decantador(:,7,k);
    SalDec_{8}(:,k) = Estados_Decantador(:,8,k);
    SalDec_{9}(:,k) = Estados_Decantador(:,9,k);
    SalDec_{10}(:,k) = Estados_Decantador(:,10,k);
    Salida_inferior_decantador(:,k) = [SalDec_{1}(:,k); Qundf(k)];
    Underflow(:,k) = inversor_fango(Salida_inferior_decantador(:,k),
Fraccion_fango(:,k));
    Recirc_externa(:,k) =[1 1 1 1 1 1 1 1 1 1 1 1 1
SRE].*Underflow(:,k)';
    Wastage(:,k) =[1 1 1 1 1 1 1 1 1 1 1 1 1 (1-
SRE)].*Underflow(:,k)';
    Recirc_interna(:,k)=[Zsal5(:,k); SRI(1,k)*Qsal5(:,k)];
    Suma_recirc(:,k) = sumacaudal(Recirc_interna(:,k),
Recirc_externa(:,k));
    Acceso_reactor(:,k) = sumacaudal(Influent(:,k),Suma_recirc(:,k));
    Salida_superior_decantador(:,k) = [SalDec_{10}(:,k); Qeff(k)];
    Effluent(:,k) = inversor_fango(Salida_superior_decantador(:,k),
Fraccion_fango(:,k));
    Psi_w(:,k) = 0.75*sum(Wastage(3:7,k))*Wastage(14,k); % Masa de
fangos generada
    if j~=3
        guarda_datos(2*j-1,:) = vector_rep(j,1:N);% - vector_rep(j,1);
    else
        guarda_datos(2*j-1,:) = (vector_rep(j,1:N))*18446;% -
vector_rep(j,1);
    end
    guarda_datos(2*j,:) = Effluent(10,1:N);% - Effluent(10,1);
end

t = (1:2*60000)/60/24;
```

```
figure;
plot(t,guarda_datos(1,:)');
hold on;
plot(t,guarda_datos(2,:)');
legend('KLa','Amonio al variar KLa');

figure;
plot(t,guarda_datos(3,:)');
hold on;
plot(t,guarda_datos(4,:)');
legend('SRI','Amonio al variar SRI')

figure;
plot(t,guarda_datos(5,:)');
hold on;
plot(t,guarda_datos(6,:)');
legend('Q','Amonio al variar Q')

%% guarda datos
k1=48e3;
k2=50e3;
cada=10;
datosss=[t(k1:cada:k2)',guarda_datos(1,k1:cada:k2)',guarda_datos(2,k1:
cada:k2)'];
save datosamKLa_N.txt datosss -ascii -tabs

datosss=[t(k1:cada:k2)',guarda_datos(3,k1:cada:k2)',guarda_datos(4,k1:
cada:k2)'];
save datosamSRI_N.txt datosss -ascii -tabs
datosss=[t(k1:cada:k2)',guarda_datos(5,k1:cada:k2)',guarda_datos(6,k1:
cada:k2)'];
save datosamQ_N.txt datosss -ascii -tabs

%% incrementales
tini=18000;
cada_muestras=600;
tend=length(t);
dt=t(tini:cada_muestras:tend);
vector_dt=tini:cada_muestras:tend;
figure;
plot(dt,guarda_datos(1,vector_dt)'-guarda_datos(1,tini)','o');
hold on;
plot(dt,guarda_datos(2,vector_dt)'-guarda_datos(2,tini)','o');
legend('KLa','Amonio al variar KLa');

figure;
plot(dt,guarda_datos(3,vector_dt)'-guarda_datos(3,tini)','o');
hold on;

plot(dt,guarda_datos(4,vector_dt)'-guarda_datos(4,tini)','o');
legend('SRI','Amonio al variar SRI')

figure;
plot(dt,guarda_datos(5,vector_dt)'-guarda_datos(5,tini)','o');
hold on;

plot(dt,guarda_datos(6,vector_dt)'-guarda_datos(6,tini)','o');
legend('Q','Amonio al variar Q')
```

A.8.- ANÁLISIS Y TRATAMIENTO DEL CAUDAL AFLUENTE

A.8.1.- PREDICCIÓN POR SERIES DE FOURIER

```
%% Predictor_Caudal_Fourier
% Este caudal analiza el comportamiento del caudal y luego intenta
% predecirlo mediante series de Fourier

clear all, clc, close all % Resetea

load Datos_dry_2006.mat % Carga los datos
Ts=15/60/24; % en dias, periodo entre datos
N=length(Datosdry);
t=(0:N-1)'*Ts; % Así se crea el vector de tiempos, restando uno a la
longitud máxima, pues el primero es tiempo 0.

% Estas operaciones permiten obtener el espectro de frecuencias.
% Las ecuaciones proceden de la pag web de Matlab
Y = fft(Datosdry(15,:)');
L=N-1;
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
Fs=1/(t(2)-t(1));
f = Fs*(0:(L/2))/L;
figure('Name','fft')
plot(f,P1)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')

% En esta parte se procederá a realizar una predicción del caudal
futuro.
Y = Datosdry(15,:)'; % Estos son los datos reales, de donde se parte
para obtener las bases de la predicción.
w=2*pi/1; % Frecuencia fundamental de la oscilación diaria
wsem=2*pi/7; % Frecuencia fundamental de la oscilación semanal

nf = 7; %nf=7; % Número máximo de harmónicos considerados para la
oscilación diaria
nfsem = 6; %nfsem=6; % Número máximo de harmónicos considerados para
la oscilación semanal

X = ones(N,1); % Valor medio considerado
Xcuad = ones (N,1);
Xcube = ones (N,1);

for i=1:nf
    X=[X,sin(i*w*t), cos(i*w*t)]; % Obtención de la matriz X (tiempo x
serie de Fourier)
end
for i=1:nfsem
    X=[X,sin(i*wsem*t), cos(i*wsem*t)]; % Ídem pero con la oscilación
semanal
end

for i=1:nf
```

```
    Xcuad=[Xcuad, sin(i*w*t), cos(i*w*t),(sin(i*w*t)).^2,
(cos(i*w*t)).^2 ];
end
for i=1:nfsem
    Xcuad=[Xcuad,sin(i*wsem*t), cos(i*wsem*t)];
end

for i=1:nf
    Xcube=[Xcube, sin(i*w*t), cos(i*w*t),(sin(i*w*t)).^2,
(cos(i*w*t)).^2,(sin(i*w*t)).^3, (cos(i*w*t)).^3]; % Obtención de la
matriz X (tiempo x serie de Fourier)
end
for i=1:nfsem
    Xcube=[Xcube,sin(i*wsem*t), cos(i*wsem*t)];
end

% El siguiente paso es usar mldivide para obtener Theta tal que Y =
X*Theta
% Theta es un vector de parámetros (serie de Fourier x 1) que se usan
para realizar la combinación lineal de la Serie de Fourier.

% No obstante, en este caso el sistema está sobredimensionado, así que
los valores de los parámetros calculados son los que minimizan la
diferencia respecto al valor de Y.

Theta=X\Y;
Yhat=X*Theta; % Yhat es distinto de Y por la aproximación antes
mencionada
% Con esto se obtiene la predicción del sistema
Thetacuad=Xcuad\Y;
Yhatcuad=Xcuad*Thetacuad;
Thetacube=Xcube\Y;
Yhatcube=Xcube*Thetacube;

figure('Name','Predicción usando todos los datos')
plot(t,Y)
hold on
plot(t,Yhat)
legend('Y','Yhat')
figure('Name','Error en la predicción usando todos los datos')
plot(t,Y-Yhat)
figure('Name','% en la predicción usando todos los datos')
plot(t,(Y-Yhat)./Y*100)
ind1 = sqrt(sum((Y-Yhat).^2)) % Indicador del error cometido

figure('Name','Cuad Predicción usando todos los datos')
plot(t,Y)
hold on
plot(t,Yhatcuad)
legend('Y','Yhatcuad')
figure('Name','Cuad Error en la predicción usando todos los datos')
plot(t,Y-Yhatcuad)
figure('Name','Cuad % en la predicción usando todos los datos')
plot(t,(Y-Yhatcuad)./Y*100)
ind1cuad = sqrt(sum((Y-Yhatcuad).^2)) % Indicador del error cometido

figure('Name','Cube Predicción usando todos los datos')
plot(t,Y)
hold on
plot(t,Yhatcube)
```

```
legend('Y','Yhatcube')
figure('Name','Cube Error en la predicción usando todos los datos')
plot(t,Y-Yhatcube)
figure('Name','Cube % en la predicción usando todos los datos')
plot(t,(Y-Yhatcube)./Y*100)
ind1cube = sqrt(sum((Y-Yhatcube).^2)) % Indicador del error cometido

figure('Name','Comparación Cuad')
plot(t,(Y-Yhat)./Y*100)
hold on
plot(t,(Y-Yhatcuad)./Y*100)
hold on
plot(t,(Y-Yhatcube)./Y*100)
legend('Yhat', 'Yhatcuad', 'Yhatcube')
ylabel('Desviación en %')


%%
figure
PerfilDiario=zeros(size(Y((1:96))));
Ndias=14;
for i=1:Ndias
    PerfilDiario=PerfilDiario+Y((1:96)+96*(i-1))/Ndias;
end
plot(PerfilDiario)
PrediccionBA=[];
for i=1:Ndias
    PrediccionBA=[PrediccionBA;PerfilDiario];
end
figure
plot(Y)
hold on
plot(PrediccionBA,'r')
figure
plot(Y(1:end-1)-PrediccionBA,'g')

figure('Name','Comparación entre distintos órdenes de la serie de
Fourier')
plot(t,Y,'k')
hold on
plot(t,Yhat,'r')
plot(t,Yhatcuad,'b')
plot(t,Yhatcube,'Color',[0.2 0.6 0.4])
legend('Y','Yhat','Yhatcuad', 'Yhatcube')
```

A.8.2.- OBTENCIÓN DEL PATRÓN BASE, GENERACIÓN DE CAUDALES Y DETECCIÓN DE LLUVIA

```
%% Tablas y observador
% En este código se crea un patrón base, se generan vectores
aleatorios,
% se actualizan los patrones base y se observa el fallo de la lluvia

clear all, clc, close all % Resetea

load Datos_dry_2006.mat % Carga los datos
Ts=15/60/24; % en dias, periodo entre datos
N=length(Datosdry);
t=(0:N-1)'*Ts;
Y = Datosdry(15,:)'; % Estos son los datos reales
Muestrasporhora = 4;
Mdia = 24*Muestrasporhora; % Muestras por día
PerfilSemanal=zeros(size(Y((1:Mdia*7))));
Nsembase=2;
for i=1:Nsembase
    PerfilSemanal=PerfilSemanal+Y((1:Mdia*7)+Mdia*7*(i-1))/Nsembase;
end
% Y así se ha obtenido la media de las semanas base, de manera que se
% obtiene un patrón de partida

%% Generador de semanas aleatorias
Nsemanas = 20; % Número de semanas
SemLluvia = 10; %Número de semanas en las que llueve en algún momento
Perfildup = repmat(PerfilSemanal,1,Nsemanas); % Se genera el vector de
semanas a rellenar
vecaleat = zeros(length(PerfilSemanal), Nsemanas);
% Esta parte genera una cierta aleatoriedad en el comportamiento de
los
% patrones diarios de subida y bajada
for k = 1: Nsemanas
    vecaleat(:,k) = 0.8 + (rand(length(PerfilSemanal),1))/2.5;
end
% En este apartado se aplica la aleatoriedad mencionada anteriormente
y se
% añade un valor medio aleatorio que representa cambios en el clima,
el
% consumo, o incluso la estacionalidad
Realidad = Perfildup.*vecaleat + rand*1000-2000;

%% El siguiente apartado genera lluvia
Lluvia = zeros(length(PerfilSemanal), Nsemanas);
% En este apartado se genera lluvia en días aleatorios, durante
periodos
% aleatorios de duración aleatoria
for i = 1:SemLluvia
    %
Lluvia(int16((rand/2)*length(PerfilSemanal)):int16((rand/2+0.5)*length
(PerfilSemanal)), int16((Nsemanas-1)*rand+1)) = rand*35000+5000;

Lluvia(int16((rand/4+0.25)*length(PerfilSemanal)):int16((rand/4+0.5)*l
ength(PerfilSemanal)), int16((Nsemanas-1)*rand+1)) = rand*15000+5000;
end
Realidad = Realidad + Lluvia;
```

```
%% Detección de fallos
NewPerfSem = zeros(length(PerfilSemanal), Nsemanas);
fallohat = zeros(length(PerfilSemanal)*Nsemanas+1,1);
beta = 0.5;
f = 1;
alfa = 0.01;

for i = 1:length(PerfilSemanal)
    NewPerfSem(i,1) = PerfilSemanal(i) + alfa*(Realidad(i,1)-
PerfilSemanal(i));
    fallohat(f+1) = fallohat(f) + beta*(Realidad(i,1)-NewPerfSem(i,1)-
fallohat(f));
    f = f+1;
end

for j = 2:Nsemanas
    for i = 1:length(PerfilSemanal)
        NewPerfSem(i,j) = NewPerfSem(i,j-1) + alfa*(Realidad(i,j)-
NewPerfSem(i,j-1));
        fallohat(f+1) = fallohat(f) + beta*(Realidad(i,j)-
NewPerfSem(i,j)-fallohat(f));
        f = f+1;
    end
end
% El NewPerfSem más reciente es el que debe utilizarse para la
predicción
% en el diseño del control predictivo.

%% Gráficos
c = jet(Nsemanas);
figure('Name','Lluvia')
for h = 1:Nsemanas
    plot(Lluvia(:,h), 'color', c(h,:))
    hold on
end

figure('Name','Actualización de tablas')
plot(PerfilSemanal, 'k--')
hold on
plot(Realidad, 'color', [0.8 0.8 0.8])
hold on

for h = 1:Nsemanas
    plot(NewPerfSem(:,h), 'color', c(h,:))
    hold on
end

figure('Name','Fallos detectados')
for h = 1:Nsemanas
    fhat = reshape(fallohat(1:end-1),[length(PerfilSemanal),
Nsemanas]);
    plot(fhat(:,h), 'color', c(h,:))
    hold on
end

figure('Name','Diferencia entre fallo y fallo detectado')
for h = 1:Nsemanas
    plot(Lluvia(:,h)-fhat(:,h), 'color', c(h,:))
    hold on
end
```

A.8.3.- GENERADOR DE SEMANAS ALEATORIAS

```
%% generasemana
% Esta función genera semanas aleatorias tomando los distintos días
% proporcionados por el Benchmark

function [ Semanas ] = generasemana2(Nsemanas)

load Datos_dry_2006.mat
Y = Datosdry(15,:)';

Lunes1 = Y((1:288)+30+288*4);
Lunes2 = Y((1:288)+30+288*4+288*7);
Lunes3 = Y((1:288)+30+288*4+288*7*2);

Martes1 = Y((1:288)+30+288*5);
Martes2 = Y((1:288)+30+288*5+288*7);

Miercoles1 = Y((1:288)+30+288*6);
Miercoles2 = Y((1:288)+30+288*6+288*7);

Jueves1 = Y((1:288)+30);
Jueves2 = Y((1:288)+30+288*7);
Jueves3 = Y((1:288)+30+288*7*2);

Viernes1 = Y((1:288)+30+288);
Viernes2 = Y((1:288)+30+288+288*7);
Viernes3 = Y((1:288)+30+288+288*7*2);

Sabado1 = Y((1:288)+30+288*2);
Sabado2 = Y((1:288)+30+288*2+288*7);
Sabado3 = Y((1:288)+30+288*2+288*7*2);

Domingo1 = Y((1:288)+30+288*3);
Domingo2 = Y((1:288)+30+288*3+288*7);
Domingo3 = Y((1:288)+30+288*3+288*7*2);

Semanas = zeros(288*7, Nsemanas);

for i = 1:Nsemanas

    Lrand1 = rand;
    Lrand2 = rand;
    Lrand3 = rand;
    Lrand = Lrand1 + Lrand2 + Lrand3;
    Lunes = (Lrand1.*Lunes1 + Lrand2.*Lunes2 + Lrand3.*Lunes3)./Lrand;

    Mrand1 = rand;
    Mrand2 = rand;
    Mrand = Mrand1 + Mrand2;
    Martes = (Mrand1.*Martes1 + Mrand2.*Martes2)./Mrand;

    Xrand1 = rand;
    Xrand2 = rand;
    Xrand = Xrand1 + Xrand2;
    Miercoles = (Xrand1.*Miercoles1 + Xrand2.*Miercoles2)./Xrand;
```

```
    Jrand1 = rand;
    Jrand2 = rand;
    Jrand3 = rand;
    Jrand = Jrand1 + Jrand2 + Jrand3;
    Jueves = (Jrand1.*Jueves1 + Jrand2.*Jueves2 +
Jrand3.*Jueves3)./Jrand;

    Vrand1 = rand;
    Vrand2 = rand;
    Vrand3 = rand;
    Vrand = Vrand1 + Vrand2 + Vrand3;
    Viernes = (Vrand1.*Viernes1 + Vrand2.*Viernes2 +
Vrand3.*Viernes3)./Vrand;

    Srand1 = rand;
    Srand2 = rand;
    Srand3 = rand;
    Srand = Srand1 + Srand2 + Srand3;
    Sabado = (Srand1.*Sabado1 + Srand2.*Sabado2 +
Srand3.*Sabado3)./Srand;

    Drand1 = rand;
    Drand2 = rand;
    Drand3 = rand;
    Drand = Drand1 + Drand2 + Drand3;
    Domingo = (Drand1.*Domingo1 + Drand2.*Domingo2 +
Drand3.*Domingo3)./Drand;

    Semanas(:,i) = [Lunes; Martes; Miercoles; Jueves; Viernes; Sabado;
Domingo];
end

end
```

A.9.- OTRAS FUNCIONES ÚTILES

A.9.1.- SUMADOR DE CAUDALES

```
%% Sumacaudal
% Esta función realiza la suma másica entre dos caudales de entrada

function [sal] = sumacaudal(ent1, ent2)
Q1=ent1(14);
Q2=ent2(14);
Qtot=Q1+Q2;
sal=zeros(14,1);
sal(1:13)=(ent1(1:13)*Q1+ent2(1:13)*Q2)/Qtot;
sal(14)=Qtot;
end
```

A.9.2.- CONVERSOR DE FANGO

```
%% Conversor_fango
% Esta función realiza la conversión a componentes particulados a la
% entrada del decantador

function [Qsal, Xfrac] = conversor_fango(Qent)
% Entradas: La composición y el caudal, en 14 estados
% Salidas: La composición y el caudal, con todos los componentes
% particulados agrupados en un estado Xf
% La proporción de los estados particulados respecto a Xf

Qsal=zeros(9,1);
Qsal(1)=Qent(1);
Qsal(2)=Qent(2);
Qsal(3)=Qent(8);
Qsal(4)=Qent(9);
Qsal(5)=Qent(10);
Qsal(6)=Qent(11);
Qsal(7)=Qent(13);
Qsal(8)=0.75*(Qent(3)+Qent(4)+Qent(5)+Qent(6)+Qent(7));
Qsal(9)=Qent(14);

Xfrac=zeros(6,1);
Xfrac(1)=Qent(3)/Qsal(8);
Xfrac(2)=Qent(4)/Qsal(8);
Xfrac(3)=Qent(5)/Qsal(8);
Xfrac(4)=Qent(6)/Qsal(8);
Xfrac(5)=Qent(7)/Qsal(8);
Xfrac(6)=Qent(12)/Qsal(8);
```

A.9.3.- INVERSOR DE FANGO

```
%% inversor_fango
% Esta función deshace las operaciones de conversor_fango

function Qsal = inversor_fango(Qent, Xfrac)
% Entradas: La composición y el caudal, con todos los componentes
% particulados agrupados en un estado Xf
% La proporción de los estados particulados respecto a Xf
% Salidas: La composición y el caudal, en 14 estados

%#codegen
Qsal = zeros(14,1);
Qsal(1:2) = Qent(1:2);
Qsal(8:11) = Qent(3:6);
Qsal(13) = Qent(7);
Qsal(14) = Qent(9);
%%
Qsal(3:7) = Xfrac(1:5)*Qent(8);
Qsal(12) = Xfrac(6)*Qent(8);
```

APÉNDICE B

B.1.- INTRODUCCIÓN

A continuación, se listará el código principal necesario para la ejecución de la simulación de la EDAR en Simulink. Para dicha simulación también se han utilizado las funciones indicadas en el Apéndice A.9.- Otras funciones útiles.

B.2.- INICIALIZACIÓN DE VARIABLES

```
%% Inicializa parámetros
% Es imprescindible ejecutar este fichero para ejecutar los
% distintos Simulink de la depuradora

load('Datos_entrada.mat');
dts=load('Datos_entrada.mat');

%% Biological parameter values
%% Stoichiometric parameters
YA = 0.24;
YH = 0.67;
fP = 0.08;
iXB = 0.08;
iXP = 0.06;

%% Kinetic parameters
muH = 4.0;
KS = 10.0;
KOH = 0.2;
KNO = 0.5;
bH = 0.3;
etag = 0.8;
etah = 0.8;
kh = 3.0;
KX = 0.1;
muA = 0.5;
KNH = 1.0;
bA = 0.05;
KOA = 0.4;
ka = 0.05;
% Para los compartimentos anaerobios
Vol = 1000;
parametros = [YA YH fP iXB iXP muH KS KOH KNO bH etag etah kh KX muA
KNH bA KOA ka Vol];
% Para los compartimentos aerobios
Vol2=1333;
parametros2 = [YA YH fP iXB iXP muH KS KOH KNO bH etag etah kh KX muA
KNH bA KOA ka Vol2];
%% Settling parameters
v0prima = 250.0;
v0 = 474;
rh = 0.000576;
rp = 0.00286;
fns = 0.00228;
A = 1500;
zm = 0.4;
Xt = 3000;
paramclarif = [v0prima v0 rh rp fns A zm Xt];
```

```
valinic1 = [30 2.81 1149 82.1 2552 148 449 0.00430 5.37 7.92 1.22 5.28
4.93];
valinic2 = [30 1.46 1149 76.4 2553 148 450 0.0000631 3.66 8.34 0.882
5.03 5.08];
valinic3 = [30 1.15 1149 64.9 2557 149 450 1.72 6.54 5.55 0.829 4.39
4.67];
valinic4 = [30 0.995 1149 55.7 2559 150 451 2.43 9.30 2.97 0.767 3.88
4.29];
valinic5 = [30 0.889 1149 49.3 2559 150 452 0.491 10.4 1.73 0.688 3.53
4.13];
```

B.3.- CÓDIGO INCLUIDO EN LA LEVEL-2 MATLAB S-FUNCTION

B.3.1.- REACTOR BIOLÓGICO

```
function depur_1(block)
% Level-2 MATLAB file S-Function for limited integrator demo.
% Copyright 1990-2009 The MathWorks, Inc.
  setup(block);
function setup(block)
  %% Register number of dialog parameters
  block.NumDialogPrms = 2;
  %% Register number of input and output ports
  block.NumInputPorts  = 2;
  block.NumOutputPorts = 1;
  %% Setup functional port properties to dynamically inherited.
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;
  block.InputPort(2).Dimensions        = 14;
  block.InputPort(2).DirectFeedthrough = true;
  block.InputPort(1).Dimensions        = 1;
  block.InputPort(1).DirectFeedthrough = false;
  block.OutputPort(1).Dimensions       = 14;
  %% Set block sample time to continuous
  block.SampleTimes = [0 0];
  %% Setup Dwork
  block.NumContStates = 13;
  %% Set the block simStateCompliance to default (i.e., same as a
built-in block)
  block.SimStateCompliance = 'DefaultSimState';
  %% Register methods
  block.RegBlockMethod('InitializeConditions',    @InitConditions);
  block.RegBlockMethod('Outputs',                 @Output);
  block.RegBlockMethod('Derivatives',             @Derivative);

function InitConditions(block)
% Initialize Dwork
  block.ContStates.Data = block.DialogPrm(2).Data;

function Output(block)
block.OutputPort(1).Data(1:13) = block.ContStates.Data;
block.OutputPort(1).Data(14) = block.InputPort(2).Data(14);

function Derivative(block)
CP = block.DialogPrm(1).Data;
YA = CP(1);
YH = CP(2);
```

```
fP = CP(3);
iXB = CP(4);
iXP = CP(5);
muH = CP(6);
KS = CP(7);
KOH = CP(8);
KNO = CP(9);
bH = CP(10);
etag = CP(11);
etah = CP(12);
kh = CP(13);
KX = CP(14);
muA = CP(15);
KNH = CP(16);
bA = CP(17);
KOA = CP(18);
ka = CP(19);
Vol = CP(20);

%% Variables de estado
STVR=block.ContStates.Data;
SI=STVR(1);
SS=STVR(2);
XI=STVR(3);
XS=STVR(4);
XBH=STVR(5);
XBA=STVR(6);
XP=STVR(7);
SO=STVR(8);
SNO=STVR(9);
SNH=STVR(10);
SND=STVR(11);
XND=STVR(12);
SALK=STVR(13);

%% Lista de procesos
% Aerobic growth of heterotrophs j = 1
ro1 = muH*(SS/(KS+SS))*(SO/(KOH+SO))*XBH;
% Anoxic growth of heterotrophs j = 2
ro2 = muH*(SS/(KS+SS))*(KOH/(KOH+SO))*(SNO/(KNO+SNO))*etag*XBH;
% Aerobic growth of autotrophs j = 3
ro3 = muA*(SNH/(KNH+SNH))*(SO/(KOA+SO))*XBA;
% Decay of heterotrophs j = 4
ro4 = bH*XBH;
% Decay of autotrophs j = 5
ro5 = bA*XBA;
% Ammonification of soluble organic nitrogen j = 6
ro6 = ka*SND*XBH;
% Hydrolysis of entrapped organics j = 7
ro7 =
kh*((XS/XBH)/(KX+(XS/XBH)))*((SO/(KOH+SO))+etah*(KOH/(KOH+SO))*(SNO/(K
NO+SNO)))*XBH;
% Hydrolysis of entrapped organic nitrogen j = 8
ro8 =
kh*((XS/XBH)/(KX+(XS/XBH)))*((SO/(KOH+SO))+etah*(KOH/(KOH+SO))*(SNO/(K
NO+SNO)))*XBH*(XND/XS);
%% Observed conversion rates
%SI (i = 1)
r(1) = 0;
%SS (i = 2)
r(2) = -(1/YH)*ro1-(1/YH)*ro2+ro7;
```

```
%XI (i = 3)
r(3) = 0;
%XS (i = 4)
r(4) = (1-fP)*ro4+(1-fP)*ro5-ro7;
%XBH (i = 5)
r(5) = ro1+ro2-ro4;
%XBA (i = 6)
r(6) = ro3-ro5;
%XP (i = 7)
r(7) = fP*ro4+fP*ro5;
%SO (i = 8)
r(8) = -((1-YH)/YH)*ro1-((4.57-YA)/YA)*ro3;
%SNO (i = 9)
r(9) = -(1-YH)/(2.86*YH)*ro2 + (1/YA)*ro3;
%SNH (i = 10)
r(10) = -iXB*ro1-iXB*ro2-(iXB+(1/YA))*ro3+ro6;
%SND (i = 11)
r(11) = -ro6+ro8;
%XND (i = 12)
r(12) = (iXB-fP*iXP)*ro4+(iXB-fP*iXP)*ro5-ro8;
%SALK (i = 13)
r(13) = -iXB/14*ro1+(((1-YH)/(14*2.86*YH))-iXB/14)*ro2-
(iXB/14+1/(7*YA))*ro3+ro6/14;
%% Balance de masas
KLa = block.InputPort(1).Data;
SOsat=8;
Zant = block.InputPort(2).Data(1:13);
Qant = block.InputPort(2).Data(14);
Qact=Qant;
block.Derivatives.Data = (Qant*Zant+r'*Vol-
Qact*block.ContStates.Data)/Vol;
block.Derivatives.Data(8) = (Qant*Zant(8)+r(8)*Vol+KLa*Vol*(SOsat-
block.ContStates.Data(8))-Qact*block.ContStates.Data(8))/Vol;
```

B.3.2.- DECANTADOR SECUNDARIO

```
function clarif_3(block)
% Level-2 MATLAB file S-Function for limited integrator demo.
% Copyright 1990-2009 The MathWorks, Inc.
  setup(block);
function setup(block)
  %% Register number of dialog parameters
  block.NumDialogPrms = 1;
  %% Register number of input and output ports
  block.NumInputPorts  = 2;
  block.NumOutputPorts = 1;
  %% Setup functional port properties to dynamically inherited.
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;
  block.InputPort(1).Dimensions        = 9;
  block.InputPort(1).DirectFeedthrough = true;
  block.InputPort(2).Dimensions        = 1;
  block.InputPort(2).DirectFeedthrough = true;
  block.OutputPort(1).Dimensions       = 28;
  %% Set block sample time to continuous
  block.SampleTimes = [0 0];
  %% Setup Dwork
  block.NumContStates = 80;
```

```
  %% Set the block simStateCompliance to default (i.e., same as a
built-in block)
  block.SimStateCompliance = 'DefaultSimState';
  %% Register methods
  block.RegBlockMethod('InitializeConditions',    @InitConditions);
  block.RegBlockMethod('Outputs',                 @Output);
  block.RegBlockMethod('Derivatives',             @Derivative);
function InitConditions(block)
InitMat = zeros(8,10);
InitMat(1,:) = ones(1,10)*30;
InitMat(2,:) = ones(1,10)*0.889;
InitMat(3,:) = ones(1,10)*0.491;
InitMat(4,:) = ones(1,10)*10.4;
InitMat(5,:) = ones(1,10)*1.73;
InitMat(6,:) = ones(1,10)*0.688;
InitMat(7,:) = ones(1,10)*4.13;
InitMat(8,:) = [6394 356 356 356 356 356 69.0 29.5 18.1 12.5];
block.ContStates.Data = InitMat(:);

function Output(block)
caudentr = block.InputPort(1).Data(9);
propcaud = block.InputPort(2).Data(1);
Qeff = caudentr*propcaud;
Qundf = caudentr*(1-propcaud);
effstates = block.ContStates.Data(73:80);
undfstates = block.ContStates.Data(1:8);
StateMat = (reshape(block.ContStates.Data, 8, 10));
Xf = StateMat(8, (1:10))';
vectorsalida = [effstates; Qeff; undfstates; Qundf; Xf];
block.OutputPort(1).Data = vectorsalida;

function Derivative(block)
DP = block.DialogPrm(1).Data;
v0prima = DP(1);
v0 = DP(2);
rh = DP(3);
rp = DP(4);
fns = DP(5);
A = DP(6);
zm = DP(7);
Xt = DP(8);
Zf = (block.InputPort(1).Data(1:7));

%% Determinación de los caudales de salida
propcaud = block.InputPort(2).Data(1);
Qf = block.InputPort(1).Data(9);
Qe = Qf*propcaud;
Qu = Qf*(1-propcaud);
vdn = Qu/A;
vup = Qe/A;

%% Cálculos internos
StateMat = (reshape(block.ContStates.Data, 8, 10));
DerivMat = zeros(8,10);

%% Cálculo de los componentes solubles
% Layer 6
DerivMat((1:7),6) = ((((Qf*Zf)/A)-(vdn+vup)*StateMat((1:7),6))/zm); %
¿Por qué es necesario el " ' "?
% Layers 1:5
```

```
for m = 1:5
   DerivMat((1:7),m) = vdn*((StateMat((1:7),m+1) -
StateMat((1:7),m)))/zm;
end

% Layers 7:10
for m = 7:10
   DerivMat((1:7),m) = vup*((StateMat((1:7),m-1) -
StateMat((1:7),m)))/zm;
end

%% Cálculo del fango
% Cálculos previos
Xf = block.InputPort(1).Data(8);
Xmin = fns*Xf;
XSludge = StateMat(8,(1:10)); % Conocido en el Benchmark como X
vsArray = zeros(1,10);
for n=1:10
    vsArray(n) = max(0,min(v0prima, v0*(exp(-rh*(XSludge(n)-Xmin))-
exp(-rp*(XSludge(n)-Xmin)))));
end
Js=vsArray.*XSludge;

% Cálculo de Jclar
Jclar = zeros(10,1);
for m = 7:10
   if XSludge(m-1)>Xt
       Jclar(m) = min(vsArray(m)*XSludge(m), vsArray(m-1)*XSludge(m-
1));
   else
       Jclar(m) = vsArray(m)*XSludge(m);
   end
end

% Cálculo por capas
% Layer 6
DerivMat(8,6) = (((Qf*Xf)/A)+Jclar(7)-(vup+vdn)*XSludge(6)-
min(Js(6),Js(5)))/zm;

% Layers 2:5
for m = 2:5
    DerivMat(8,m) = (vdn*(XSludge(m+1)-XSludge(m))+min(Js(m),Js(m+1))-
min(Js(m),Js(m-1)))/zm;
end

% Layer 1
DerivMat(8,1) = (vdn*(XSludge(2)-XSludge(1))+min(Js(2),Js(1)))/zm;

% Layers 7:9
for m = 7:9
   DerivMat(8,m) = (vup*(XSludge(m-1)-XSludge(m))+Jclar(m+1)-
Jclar(m))/zm;
end

% Layer 10
DerivMat(8,10) = (vup*(XSludge(9)-XSludge(10))-Jclar(10))/zm;

%% Reasignación de la matriz de derivadas
block.Derivatives.Data = DerivMat(:);
```