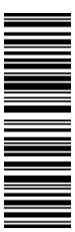

Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Máster en Ingeniería Química

Implementación de un controlador predictivo no lineal y una estrategia de optimización en tiempo real a una planta piloto híbrida

Autor:

Montes López, Daniel Alberto

Tutor:

De Prada, César
Dpto. de Ingeniería de Sistemas y Automática

Valladolid, mayo 2020.

Implementación de un controlador predictivo no lineal y una estrategia de optimización en tiempo real a una planta piloto híbrida

Daniel Montes

Resumen

En el presente Trabajo de Fin de Máster se estudia el desarrollo e implementación inicial de estrategias de control avanzado y optimización de tiempo real en una planta piloto de laboratorio. Se busca poner en práctica a escala de laboratorio capas superiores de la pirámide de automatización que cobran cada vez más relevancia en la industria de procesos.

La planta simula el comportamiento dinámico de un reactor químico sin que tenga lugar la reacción en sí. La reacción se simula a partir de las medidas del proceso para calcular el calor que se generaría y luego se aplica a través de resistencias eléctricas. Aunque es una simplificación del proceso real, este enfoque permite conservar la hidrodinámica original y eliminar la necesidad de compra y disposición de reactivos. El único fluido involucrado en el proceso es agua.

Se desarrolló y ajustó un modelo matemático de la planta usando estimación de parámetros con optimización dinámica. Luego, se programó el controlador predictivo no lineal en EcosimPro y se envolvió en una capa OPC-UA. Fue necesario proveer al controlador de un estimador de estados de horizonte móvil para calcular las variables no medidas. El controlador predictivo calcula valores de caudal de refrigerante y reactivos que se envían como consignas a controladores PID. Para comunicar el controlador con el SCADA se desarrolló una interfaz entre OPC-UA y OPC-DA en Python usando librerías de código abierto.

Por último, se implementó una estrategia de optimización en tiempo real que calcula las consignas de temperatura y concentración del controlador predictivo que maximizan el beneficio económico del proceso. Desde el SCADA es posible cambiar el precio de cada una de las especies involucradas en la reacción y el costo del refrigerante. Los valores de concentración y temperatura son enviados automáticamente al controlador predictivo.

Debido a la situación de estado de alarma decretado por el Estado español no fue posible acceder al laboratorio durante la última fase del TFM, por ello, tanto el controlador predictivo como el optimizador en tiempo real se probaron en una simulación del proceso conectada al SCADA de la planta. Se probaron exitosamente cambios de consigna, de parámetros y perturbaciones y el controlador respondió adecuadamente a cada uno de ellos. Luego, se probaron diferentes situaciones con el optimizador en tiempo real como el desplome del precio de la especie de interés.

Implementation of a non-linear predictive controller and a real-time optimization strategy on a hybrid pilot plant

Daniel Montes

Abstract

In the present Master's thesis, the initial implementation of a non-linear predictive controller and a real-time optimization strategy on a hybrid pilot plant were studied. It aims to put into practice superior layers of the automation pyramid that are gaining relevance in the process industry.

The plant simulates the dynamic behavior of a continuous stirred tank reactor without the reaction actually taking place. The reaction is simulated from real measurements of the process to calculate the heat that would be generated, it is then applied using electrical heating coils. Although it is a simplification of the real process, this approach allows us to preserve the fluid dynamics while deleting the need and handling of reactants and products. The only fluid involved in the process is water.

A mathematical model was developed and adjusted using parameter estimation with dynamic optimization. Later on, a non-linear predictive controller was coded in EcosimPro and wrapped in an OPC-UA interface. It was necessary to provide the controller with a moving horizon state estimator to calculate the value of the unmeasured variables. The predictive controller calculates values for the flowrates of coolant and reactants required to achieve the temperature and concentration set-point. These flowrates are sent as set-points to PID controllers. To establish communication between the controller and the plant's SCADA an interface between OPC-UA and OPC-DA was developed in Python using open-source libraries.

Finally, a real-time optimization strategy that calculates the controller's temperature and concentration set-points that maximize the economics of the process. It is possible to adjust from the SCADA the price of each species and the price of the coolant. The concentration and temperature set-points are automatically sent to the predictive controller.

Both the predictive controller and the real-time optimizer were tested on a simulation of the process connected to the SCADA. Different changes in set-points, tuning parameters, and disturbances were tested and the controller reacted properly to each situation. Later, different situations were tested in the real-time optimizer such as in a sudden drop in the price of the most profitable species.

Dedicatoria

Dedico este trabajo a mis padres y a mi pareja por apoyarme desde el principio en la loca idea de dejar atrás todo mi mundo en busca de nuevos horizontes. Su apoyo y confianza siempre me ha inspirado a seguir adelante.

Agradecimientos

Me gustaría agradecer a cada uno de los profesores del Máster en Ingeniería Química por sus lecciones. Al personal del Laboratorio de Control de Procesos por su ayuda con los múltiples inconvenientes con la planta. A César por su excelente tutoría y permanente disponibilidad para atender mis inquietudes.

Igualmente deseo dar las gracias a la Universidad de Valladolid y al Banco Santander por financiar mis estudios de máster a través del programa de becas Iberoamerica+Asia.

Índice general

Capítulo 1

Introducción

La optimización se define como el uso de métodos específicos de forma sistemática para determinar las soluciones más efectivas a un problema o diseño de un proceso [1]. Esta herramienta es muy importante en las áreas de la ciencia, la ingeniería, la electrónica, los negocios, etc. Además, es una de las principales técnicas cuantitativas para soportar la toma de decisiones en entornos industriales. Los objetivos en cualquier problema de optimización suelen incluir términos como "máximo beneficio", "mínima pérdida", "máxima probabilidad", "mínimo esfuerzo", "máximo aprovechamiento de recursos", etc.

La industria química desempeña un papel importantísimo en la economía de prácticamente cualquier país del mundo. Específicamente, en España representó en 2019 el $13,4\,\%$ del PIB nacional, generando así alrededor de 670000 puestos de trabajo y ocupó el segundo lugar en las exportaciones [2]. El ingreso de países con bajos costes salariales como India o China al comercio mundial es un gran desafío para el sector y supone una feroz competencia. Además, la cada vez más estricta legislación en temas ambientales y energéticos requiere que las industrias estén en un permanente esfuerzo por mejorar y optimizar sus operaciones. Los límites son cada vez más estrechos y el uso de las materias primas ha de ser lo más eficiente posible. Las herramientas de optimización son por tanto clave para cumplir tales objetivos.

Aunque el control tradicional, como los algoritmos PID, permite mantener los procesos en puntos de operación establecidos, se queda corto a la hora de optimizar el uso de los recursos y los bienes de las factorías. Además, los procesos se hacen más complejos, como es el caso con la integración energética y la intensificación. Por estas razones, técnicas avanzadas como el control predictivo basado en modelos y la optimización en tiempo real son cada vez más atractivas. Todo esto se basa en las ventajas de los modelos matemáticos, pues es posible predecir el estado futuro del proceso y calcular las mejores acciones de control que lo lleven a las condiciones deseadas. Además, los mismos modelos pueden ser usados para determinar las condiciones de operación óptimas en términos económicos, máximizando así el beneficio de la factoría mientras se cumplen todas las restricciones y manteniendo, incluso mejorando, la calidad del producto.

La pirámide de automatización, mostrada en la figura 1, es una forma visual de explicar los cinco niveles de un proceso de automatización y de qué forma se integran todas las tecnologías que involucra. En el primer nivel se encuentran los dispositivos de campo como los sensores y actuadores. Los dispositivos de control como los PID y los PLC recogen los datos del primer nivel para regular la planta. El nivel de supervisión está compuesto por los sistemas SCADA, incluyendo las estrategias de control avanzado como el control predictivo. En este nivel, se toman decisiones de procesos. Por encima de este, se encuentra

la capa de planificación local, donde se encuentran soluciones MES y de optimización en tiempo real para tomar decisiones de planta. Por último, en el nivel cinco se gestiona de forma global la producción de la empresa, asignando a cada una de sus plantas cotas de producción, compra de materias primas, etc.

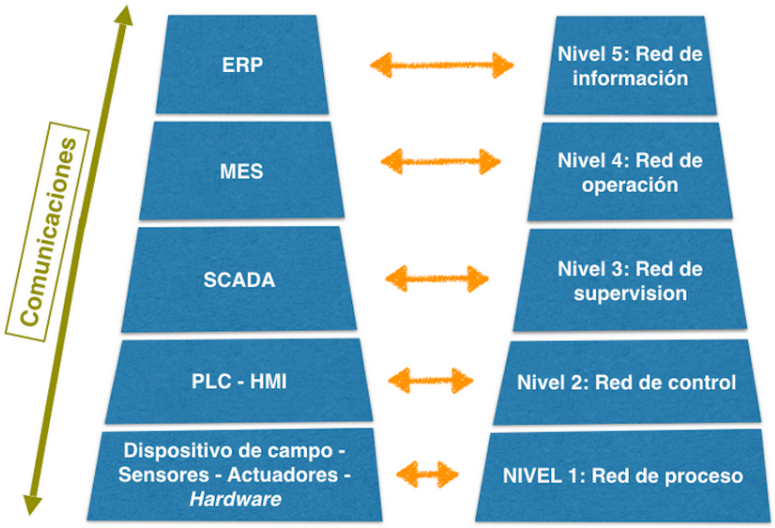

Figura 1: Pirámide de automatización.

Este trabajo se interesa particularmente por la implementación a escala de laboratorio de las capas de control avanzado y optimización en tiempo real a una planta piloto de un reactor químico. Debido al costo de materias primas y la complejidad inherente de almacenar, manipular y desechar sustancias químicas, se ha usado una planta en la que la reacción química está simulada pero toda la hidrodinámica del proceso se mantiene fiel a la realidad. Todos los elementos físicos del reactor están presentes y funcionan, pero la única sustancia involucrada es agua. Se alimenta una simulación de la reacción con medidas reales del proceso, y el calor que generaría la reacción se aplica mediante resistencias eléctricas. La implementación de estas herramientas servirá como demostración práctica de las capas superiores de la pirámide de automatización en una planta de laboratorio relativamente compleja y con dinámicas altamente no lineales. Aunque este estudio se limitará a simulaciones de la planta, se desarrollarán todas las interfaces de comunicación necesarias para la implementación final. Fue imposible obtener datos reales del proceso debido a la situación mundial en la que se enmarca el presente trabajo.

En lo que resta del capítulo 1 se detallarán los objetivos del trabajo así como también se dará un breve repaso a la teoría sobre la que se basa. En el capítulo 2 se presentará una descripción de la planta piloto usada y de los elementos que la conforman. Posteriormente, en el capítulo 3 se desarrollará un modelo matemático del proceso, incluyendo las etapas de estimación de parámetros y validación. En el capítulo 4 se documentará el desarrollo e implementación del controlador predictivo no lineal, mientras que, en el capítulo 5 se presentará el optimizador en tiempo real. El trabajo concluirá con la presentación y discusión de resultados en el capítulo 6 y las conclusiones en el capítulo 7. Todo el código desarrollado en el trabajo y demás herramientas se documentarán al final de la memoria en el capítulo de anexos.

1.1. Objeto

Este trabajo tiene como objeto diseñar, desarrollar e implementar capas superiores de la pirámide de control, control avanzado y optimización en tiempo real, a una plata piloto híbrida de laboratorio que opera en tiempo real. Este conjunto de herramientas servirá como demostración práctica de temas que son cada vez más relevantes en la industria de procesos. Aunque, como ya se ha señalado, dadas las circunstancias asociadas al estado de alarma no fue posible desarrollar todo el trabajo en la planta de laboratorio como era nuestra intención inicial, no obstante, se trabajará sobre una simulación de la planta conectada al SCADA, se desarrollarán todas las interfaces de comunicación y ventanas de configuración necesarias. Para alcanzar el objetivo final se han establecido cuatro etapas principales a ser:

- Puesta a punto de la instalación: Las capas inferiores de la pirámide de control deben funcionar adecuadamente previo a la implementación de capas superiores. Se reconfigurarán los lazos PID de control de temperatura y concentración de la planta a caudales de refrigerante y reactivos.
- Modelado, estimación de parámetros y validación: Para poder diseñar un controlador predictivo, se ha de disponer un modelo dinámico del proceso que permita hacer predicciones fiables. El modelo a desarrollar será uno basado en las leyes de conservación. La etapa de estimación de parámetros y validación se desarrollarán en el entorno de simulación EcosimPro ®.
- Diseño del controlador: Una vez validado el modelo, se procederá con el diseño del controlador. Se implementará en la herramienta EcosimPro ® y luego se embeberá en un servidor OPC-UA. De esta manera su implementación se asemejará a como sería la integración en un sistema de control distribuido. Por otra parte, será necesario desarrollar una interfaz en Python para comunicar el servidor OPC con el SCADA de la planta.
- Optimización: Antes de usar cualquier estrategia de optimización, todas las capas de regulación deben funcionar adecuadamente. Se comprobará el funcionamiento del controlador predictivo, y luego se implementará una estrategia de optimización en tiempo real para determinar las consignas del mismo maximizando el beneficio económico. Funcionará como una capa superior que comanda a los sistemas de control.

1.2. Control de procesos

El objetivo principal del control de procesos es mantener el proceso en unas condiciones de operación deseadas, de forma segura y económicamente aceptable, a la vez que se satisfacen las regulaciones ambientales y los requisitos de calidad [3]. Se basa fundamentalmente en comparar las medidas de las variables con los valores deseados y actuar en consecuencia. En cualquier sistema de control se pueden distinguir al menos los siguientes tres elementos básicos:

- Variables controladas: Lo que se quiere controlar. El valor deseado se conoce como set-point, consigna o referencia.
- Variables manipuladas: Lo que se ajusta para mantener las variables controladas en su set-point. Normalmente son posiciones de válvulas, o flujos.
- Perturbaciones: Lo que afecta a las variables controladas pero que no se puede manipular.

La adecuada especificación de cada uno de estos elementos es clave y para nada trivial en casos en los que hay varias variables presentes. Normalmente se hace basado en el conocimiento del proceso, los objetivos de control y, como no, la experiencia cuando no hay interacción entre ellas. Sin embargo, en los casos que si hay interacción es mejor usar la técnica de la matriz de ganancia relativa de Bristol para encontrar el mejor emparejamiento [3].

En esta sección se explicarán los principios de funcionamiento, las ventajas y desventajas de los controladores PID y de los controladores predictivos.

1.2.1. Control PID

El algoritmo de control Proporcional-Integral-Derivativo (PID) ha sido ampliamente exitoso en la industria de procesos desde los años 40 y sigue siendo hoy en día la estrategia de control más utilizada [4]. Se usa para sistemas en los que hay una entrada y una salida (SISO). Normalmente, las unidades de proceso tienen implementados varios lazos de control de este tipo de forma simultánea. El rendimiento de cada lazo se verá afectado evidentemente por las interacciones con los otros. Una forma de mejorar el rendimiento es usar técnicas de desacoplo.

En un controlador PID la acción de control se calcula de acuerdo a la ecuación 1. Donde K_c, T_i y T_d son los parámetros que deben sintonizarse en cada lazo de control. Existentes multitud de métodos de sintonía como el de Ziegler-Nichols.

$$u(t) = K_c \left(E(t) + \frac{1}{T_i} \int_0^t E(\tau) d\tau + T_d \frac{\mathrm{d}E(t)}{\mathrm{d}t} \right) \tag{1}$$

Conviene conocer las características de cada uno de los modos:

- Proporcional: Proporciona un rápido ajuste de la variable manipulada pero, aunque reduce el error, no es capaz de garantizar el valor del set-point por si mismo. Puede causar inestabilidad si no se ajusta adecuadamente.
- Integral: Ajusta la variable manipulada más lentamente que el modo proporcional pero permite alcanzar exactamente el valor del set-point.
- Derivativo: No tiene influencia en el valor final del error pero provee una rápida respuesta ante la velocidad de cambio del mismo. Puede ser preferible configurarlo con la velocidad de cambio de la variable controlada. Además, puede causar variaciones de alta frecuencia en los elementos finales de control.

Con esta información, es posible deducir algunas de las ventajas del control PID. Es un concepto muy fácil de entender y no se requieren modelos matemáticos complicados del proceso. Son fáciles de implementar y no requieren demasiado esfuerzo. Además, suelen ser suficientes para la mayoría de necesidades de regulación de las plantas de proceso.

Por otra parte, las crecientes exigencias de optimización de costes, mejora de la calidad, productividad, seguridad, legislación ambiental, etc. requiere una mejora de los sistemas de control para poder cumplir las especificaciones. Los controladores PID no incorporan conocimiento del proceso y pueden no funcionar bien en lazos con dinámicas difíciles como retardos o fases no mínimas. Además, la interacción entre variables puede empeorar considerablemente el rendimiento de los sistemas de control.

1.2.2. Control predictivo

El control predictivo (MPC) intenta mejorar la calidad del control incorporando conocimiento del proceso, en forma de un modelo matemático del mismo, en la toma de decisiones. Suponga un proceso de varias entradas y varias salidas que se quiere controlar, satisfaciendo ciertas restricciones. El concepto de control predictivo se resume en el uso de un modelo dinámico y las medidas de los sensores para predecir los valores futuros de las variables controladas. Así, se pueden calcular simultáneamente valores adecuados para todas las variables manipuladas que permitan satisfacer el objetivo y las restricciones. Todas las posibles interacciones entre las variables serán consideradas de antemano por el modelo del proceso. El control predictivo ofrece cuatro principales ventajas:

1. El modelo del proceso captura la dinámica y las interacciones entre las variables manipuladas, controladas y perturbaciones.
2. Se pueden establecer restricciones en las variables manipuladas y controladas de forma sistemática.
3. El sistema de control se puede complementar con el cálculo óptimo de las consignas.
4. Predicciones precisas del modelo pueden advertir de posibles problemas o fallos.

Es importante destacar que el éxito de un controlador predictivo dependerá enormemente de la precisión del modelo. Predicciones imprecisas podrían incluso empeorar el proceso, en lugar de mejorarlo. Por ello la formulación del mismo debe incorporar acción integral para tener en cuenta y corregir las imprecisiones del modelo.

El concepto de MPC se entiende mejor con la ayuda de la figura 2, se presenta de forma simplificada para el caso de un sistema de una entrada y una salida. En el instante de muestreo actual, k, el controlador calcula un conjunto de M valores para la variable manipulada, $\{u(k+i-1), i = 1, 2, ..., M\}$. Luego del horizonte de control, M, las variables manipuladas se mantienen en un valor constante. Estas acciones de control se calculan de tal manera que el conjunto, P, de salidas predichas $\{\hat{y}(k+i), i = 1, 2, ..., P\}$ alcance el set-point de forma óptima. Aunque se calcule un conjunto de acciones de control, solo se aplica el primer valor calculado ya que en cada periodo de muestreo se repiten los cálculos. Esta estrategia de control se basa en minimizar una determinada función de coste como se muestran en la ecuación (2).

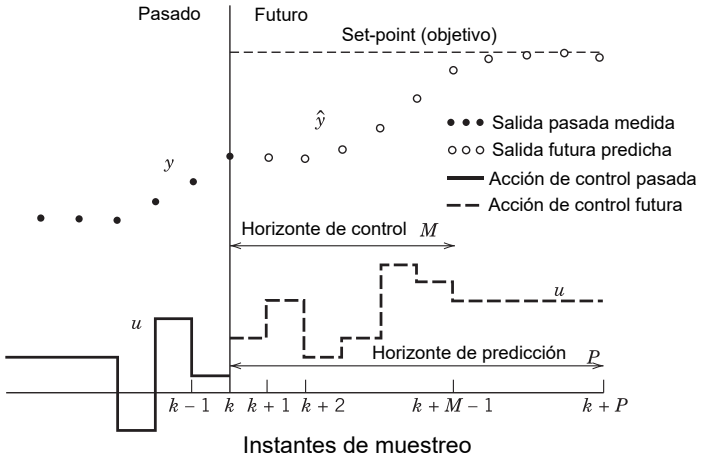

Figura 2: Concepto básico de un MPC [3].

$$\begin{aligned} \min_{u(k),\dots,u(k+M-1)} \quad & \sum_{j=1}^{P} \left[\hat{y}(t+j) - w(t+j)\right]^2 + \sum_{j=0}^{M-1} \left[\beta \Delta u(t+j)\right]^2 \\ \text{s.a.} \quad & \text{ecuaciones del modelo,} \\ & y_{low} \leq \hat{y} \leq y_{upp}, \\ & u_{low} \leq u \leq u_{upp} \end{aligned} \tag{2}$$

Donde $w(t+j)$ representa el set-point, que también podría ser una trayectoria, y β es un coeficiente que penaliza los cambios en la variable manipulada. En sistemas MIMO se añaden parámetros para dar importancia relativa a las diferentes variables. Encontrar valores adecuados para todos estos parámetros no es una tarea trivial y no existen métodos sencillos de sintonía como para los controladores PID. Métodos basados en técnicas de inteligencia artificial son muy comunes. Aunque el significado de cada parámetro es muy claro, cuando hay muchas variables hasta un usuario avanzado podría tener problemas para decidir cual es el mejor conjunto de valores para los parámetros de sintonía. Para aliviar un poco este problema, los fabricantes suelen incluir valores por defecto, los enmascaran en variables más sencillas o proveen métodos de sintonía automática.[5].

El corazón de cualquier controlador predictivo es por supuesto el modelo matemático. La mayoría de MPC comerciales se basan en modelos lineales, como los de respuesta salto o respuesta impulso. Sin embargo, solo son validos alrededor del punto sobre el que se hace la linealización y pueden quedar obsoletos rápidamente si hay algún cambio en el sistema. Por otra parte, los modelos de base fenomenológica pueden comprender condiciones de operación mucho más amplias a expensas de su complejidad, es decir, son no lineales. Se debe tener en cuenta que los modelos lineales dan lugar a óptimos globales del problema de optimización del controlador, contrario a los no lineales que solo pueden proporcionar óptimos locales. En la industria, prácticamente todos los MPC se basan en modelos del primer tipo, aunque es posible encontrar aplicaciones de los no lineales [5].

Como se puede ver en la ecuación (3), los MPC se formulan como problemas de optimización dinámica. En este trabajo se usará EcosimPro ® para resolver el modelo de la planta y calcular la función de coste. Estos valores se enviarán a un optimizador que calculará las acciones de control. La estrategia de resolución se detallará en el capítulo 4.

$$\begin{aligned} \min_{u_j(t)} \quad & \sum_{k=1}^{N_k} \sum_{i=0}^{N} \gamma_k \left[\frac{y_k(t+i) - w_k(t+i)}{y_{k,upp} - y_{k,low}}\right]^2 + \sum_{j=1}^{N_j} \sum_{i=0}^{Nu} \beta_j \left[\frac{\Delta u_j(t+i)}{u_{j,upp} - u_{j,low}}\right]^2 \\ \text{s.a.} \quad & \text{ecuaciones del modelo,} \\ & y_{k,low} \leq y_k \leq y_{k,upp} \quad k = 1, \dots, N_k, \\ & u_{j,low} \leq u_j \leq u_{j,upp} \quad j = 1, \dots, N_j \end{aligned} \tag{3}$$

Por otra parte, en la mayoría de las unidades de proceso, algunas de las variables de estados pueden no estar medidas. Tanto la temperatura como la presión son fáciles y baratas de medir pero, por ejemplo, la concentración suele ser una medida bastante costosa. Los analizadores encargados de hacerlo suelen ser caros y requerir constantes calibraciones. Por otra parte, los sensores casi siempre tienen ruido o pueden no reflejar con exactitud el valor real de la variable [6]. Determinar estimaciones adecuadas para las variables desconocidas es un campo desafiante pero ampliamente estudiado. Un gran ejemplo de estimador de

estados es el filtro de Kalman, usado en prácticamente todos los campos de la tecnología [7].

Una de las técnicas más eficaces para estimar estados de modelos no lineales son los estimadores de horizonte móvil (MHE). Su funcionamiento consiste en estimar los estados iniciales de las variables no medidas y el valor de las perturbaciones en el horizonte de estimación de forma que el proceso evolucione lo más cerca posible de las variables medidas. Con estos valores, simulando el modelo se pueden obtener los valores de las variables no medidas en el instante de tiempo actual. En esencia se formula como un problema de optimización dinámica, de manera idéntica a los problemas de estimación de parámetros con la diferencia de que se resuelven en tiempo real. Los estimadores también proveen de acción integral a los MPC contrarrestando las discrepancias que haya entre el modelo y el proceso. En el capítulo 4 se desarrollará en detalle un MHE para la planta piloto objeto de este trabajo.

1.3. Optimización en tiempo real

Como se mencionó en la sección anterior, el objetivo primordial del control de procesos es mantener ciertas condiciones de operación. La optimización en tiempo real (RTO) permite calcular de forma sistemática, y en periodos regulares de tiempo, las mejores condiciones de operación. Pudiendo ser estas las que lleven al máximo beneficio económico, maximicen la eficiencia en el uso de materias primas, minimicen los costos, etc.

Se formula como un problema de optimización en el que se incluye información económica de la planta. Por ejemplo, los costes de las materias primas, las utilidades, tasas, etc. Además, es preferible que este tipo de sistemas engloben a toda la planta de proceso pues las condiciones óptimas de operación de una unidad de proceso no necesariamente coinciden con el óptimo de toda la planta, es más, podrían incluso afectar negativamente el rendimiento general de la factoría [8]. Por otra parte, es preferible usar modelos de base fenomenológica a los que se obtienen a partir de datos ya que los nuevos set-point podrían alejarse de su rango de validez. Es usual que se resuelvan los modelos en estado estacionario pues el RTO debe proveer información del punto en el que debería mantenerse la operación de la planta. Algunos controladores predictivos del mercado, como el DMC de AspenTech, incluyen capas de RTO.

Los sistemas de optimización en tiempo real son habituales en grandes plantas de proceso como las refinerías de petróleo pero su implementación es muy interesante para otro tipo de industrias. La refinería de Petronor en el norte de España tiene implementado un sistema de este tipo para gestionar una red de hidrógeno de cuatro plantas productoras y catorce consumidoras que ha dado excelentes resultados [8].

1.4. Sistemas SCADA

Los sistemas SCADA son fundamentales en prácticamente cualquier actividad de la industria de procesos. Permiten monitorizar, supervisar, recolectar y procesar datos en tiempo real. Además, son el puente de interacción entre el proceso y los operarios. Su principal objetivo es asegurar el correcto funcionamiento del proceso, incluso generando alarmas o reconfigurando secciones de la planta en situaciones anómalas. Están conformados principalmente por sensores y actuadores, unidades remotas (PLCs), redes de comunicación y terminales de operación. También se caracterizan por ser sistemas modulares, por lo que los fallos pueden solo pueden afectar secciones individuales de la planta y no a todo el conjunto.

1.4.1. Estándar OPC

Por otra parte, una de las tareas más difíciles en cualquier entorno industrial es la integración de todos los sistemas y dispositivos. Antiguamente, cada dispositivo de campo venía con una interfaz propietaria del fabricante. Las aplicaciones de automatización debían tener drivers específicos para cada uno de estos. Así, se requería un driver para cada pareja de elementos que se debían comunicar. Esto generaba altos costes de instalación, poca flexibilidad y una necesidad elevada de mantenimiento y soporte. El estándar OLE for Process Control (OPC) nació para solucionar este problema de interoperabilidad pues establece una interfaz común entre dispositivos. Está basado en un modelo cliente-servidor de forma que los clientes demandan información y los servidores la proveen. La comunicación es varios a varios, es decir, un cliente puede establecer conexión con varios servidores y, a su vez, cada servidor puede establecer comunicación con varios servidores.

El estándar de comunicación OPC es muy relevante para el presente trabajo, pues además de las comunicaciones, la simulación de la reacción química, el controlador predictivo y el optimizador en tiempo real están envueltos en capas OPC. Así se logra facilitar la integración de todos los elementos en el SCADA a la vez que se usan herramientas industriales.

1.4.2. OPC Data Access (OPC-DA)

La especificación OPC-DA se centra en el acceso de datos en línea, es decir, la lectura y escritura entre una aplicación (SCADA) y un dispositivo de control de procesos (tarjetas de adquisición de datos, PLCs). Como pertenece a la primer revisión del estándar OPC, está basada en las tecnologías OLE/COM de Microsoft, por lo que solo son compatibles con sistemas operativos Windows. Los servidores OPC-DA se organizan jerárquicamente en objetos como sigue [9]:

- Servidor: Sirve de interlocutor con los clientes y como contenedor para las objetos grupo.
- Grupo: Mantiene información de algunas propiedades, como el tiempo de actualización o la banda muerta. Ofrece mecanismos para contener y organizar lógicamente los objetos item.
- item: Representan conexiones a fuentes de datos dentro del servidor. Puede ser, por ejemplo, medidas de sensores, señales de actuadores, parámetros de controladores, etc.

Aunque es un estándar antiguo, OPC-DA sigue siendo hoy en día el estándar de comunicación más usado en la industria. Su fiabilidad y gran adopción lo hacen una herramienta difícil de reemplazar.

La tarjeta de adquisición de datos (sección 2.3.3), la simulación de la reacción química (sección 2.4.1) y los controladores PID (sección 2.4.2) de la planta objeto de este trabajo están envueltos en servidores OPC-DA.

1.4.3. OPC Unified Architecture (OPC-UA)

Debido a que el estándar clásico de OPC estaba basado en tecnología propietaria de OPC, la fundación OPC desarrolló el nuevo estándar OPC-UA. Este se basa en tecnología web, por lo que se elimina la dependencia del protocolo COM de Microsoft. Además, se incorporaron múltiples mejoras como sistemas de autenticación, soporte para redundancia, etc. Todo esto manteniendo retrocompatibilidad con las especificaciones anteriores [10].

Al igual que en OPC-DA, OPC-UA tiene una organización jerárquica pero ahora es en carpetas y sub-carpetas. Lo que eran items ahora se conocen como nodos. Añadiendo características interesantes como que ahora hay métodos que pueden ser nodos.

Este estándar es de suma importancia para el presente trabajo pues todas las herramientas que requieran compiladores modernos de C++ no son compatibles con el estándar OPC-DA. Tanto el controlador predictivo como el optimizador en tiempo real están envueltos en capas OPC-UA.

1.5. Optimización dinámica

Es una práctica común en la ingeniería química modelar los procesos a partir de las leyes de conservación de materia, energía y momento. Desde el modelado de la evolución del nivel en un depósito hasta en el diseño de una torre de destilación, aparecen sistemas de ecuaciones que son combinaciones de ecuaciones diferenciales y algebraicas. Especial mención merecen los procesos por lotes, en los que no interesa el estado estacionario sino la trayectoria que siguen las diferentes variables a lo largo del tiempo. Con el creciente uso y aceptación de aplicaciones de simulaciones de procesos dinámicos, nace también la necesidad de optimizarlos y con ello el concepto de la optimización dinámica [11]. Este campo de la ingeniería estudia los problemas de optimización que tienen la particularidad de que algunas de sus restricciones contienen ecuaciones diferenciales y algunas variables o parámetros varían a lo largo del tiempo, o el espacio. Esta forma de optimización es de particular interés para el presente trabajo pues se aplicará en la estimación de parámetros del modelo del reactor y para calcular las acciones del controlador predictivo.

Las aplicaciones son muy variadas pero podrían dividirse en dos grandes categorías según como sea su ejecución: fuera de línea (off-line) o en línea (on-line). La distinción entre estas dos categorías es clara y las aplicaciones específicas son muy diversas. En la ejecución fuera de línea se pueden resolver problemas como el diseño de un reactor o una columna de separación, la estimación de parámetros desconocidos de un modelo a partir de datos experimentales o la optimización de los perfiles de operación de un proceso por lotes. Aunque en este tipo de problemas no se requiere que el problema se resuelva en simultaneo con el proceso que estudia, pueden necesitarse datos del mismo. Por otra parte, la ejecución en tiempo real si está ligada a un proceso y suele usarse para control predictivo, optimización en tiempo real o reconciliación de datos [11]. De esta manera, el problema de optimización se resuelve periódicamente a medida que le llegan nuevos datos del proceso. Más allá de la ingeniería de procesos, este tipo de optimización es recurrente en los campo de la economía, la industria aérea, militar y espacial [12].

De forma general, un problema de optimización dinámica puede formularse como aparece en la ecuación 4. Las variables de decisión pueden ser por ejemplo una trayectoria a lo largo del tiempo, un estado inicial, el tiempo para llegar a una referencia, etc. En esta formulación, $u(t)$ representa un conjunto de parámetros o variables que evolucionan a lo largo del tiempo, $x(t)$ y x_0 son las variables de estado y sus valores iniciales, t_f es el tiempo final de la simulación, $J(u)$ es la función de coste y $h(x, u, z)$ y $g(x, u, z)$ son restricciones de igualdad y desigualdad respectivamente.

$$\begin{aligned} \min_{u(t),x_0,t_f} \quad & J(u) = \int_{t_0}^{t_f} L(x,u)dt \\ \text{s.a.} \quad & \frac{dx}{dt} = f(x,u,p), \\ & x(t_0) = x_0, \\ & h(x,u,z) = 0, \\ & g(x,u,z) \leq 0 \end{aligned} \tag{4}$$

Para resolver este tipo de problemas se deben integrar las ecuaciones diferenciales a lo largo del tiempo, siendo las soluciones analíticas extremedamente raras y solo aplicables para casos muy sencillos. Existen principalmente dos familias de estrategias para resolver problemas de optimización dinámica: métodos indirectos e indirectos. En los primeros, se formulan y resuelven las condiciones necesarias de optimalidad. Esto no siempre es posible y se suelen requerir soluciones analíticas de las ecuaciones diferenciales. Por otra parte, los métodos directos se basan en discretizar las variables que dependen del tiempo y resolverlas numéricamente [11]. A su vez los métodos directos se dividen en dos subfamilias, dependiendo de como se haga la discretización, el enfoque simultáneo y el enfoque secuencial como se muestra en la figura 3.

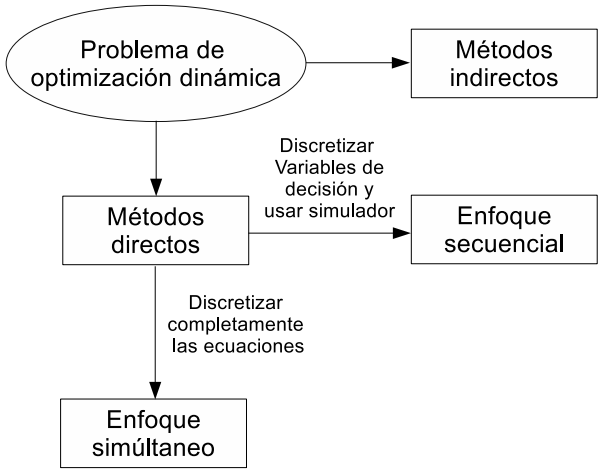

Figura 3: Estrategias de solución de problemas de optimización dinámica

1.5.1. Enfoque secuencial

El primer paso en el enfoque secuencial consiste en discretizar aquellas variables de decisión que varíen en el tiempo. Existen múltiples aproximaciones pero la más común es asignar valores constantes durante ciertos periodos de tiempo como se muestra en la figura 4. Una forma sencilla de implementar esta tipo de discretización es usar una función sigmoide como se muestra en la ecuación 5. Para lograr el mismo efecto de escalones de la figura 4, basta con asignar un valor alto al parámetro σ.

$$u(t) = u_1 + \sum_{i=2}^{N} \frac{u_i - u_{i-1}}{1 + e^{-\sigma(t-t_i)}} \tag{5}$$

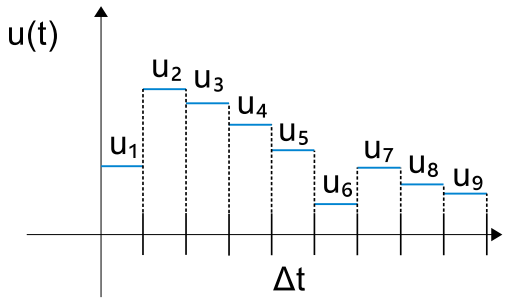

Figura 4: Discretización de variables de decisión.

Se usa un simulador dinámico para resolver las ecuaciones diferenciales y calcular tanto la función de coste como las restricciones. Luego, un optimizador NLP se encarga de suministrar nuevos valores para las variables de decisión. El simulador recalcula y el ciclo se repite hasta que se alcanza cierto criterio de parada, normalmente que no se consiga mejorar significativamente la función de coste. En la figura 5 se muestra un esquema de como funciona esta estrategia. Entornos de simulación dinámica como EcosimPro ® pueden ser utilizados para resolver este tipo de problemas.

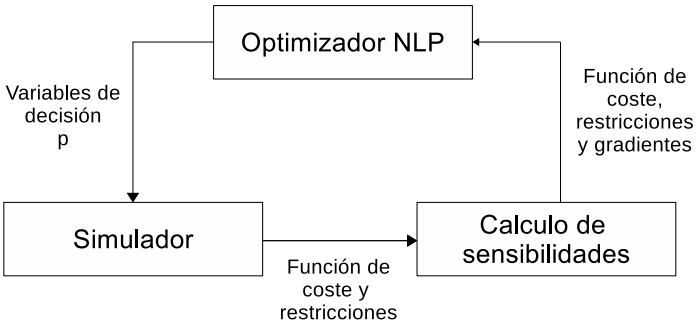

Figura 5: Enfoque secuencial de optimización dinámica.

Por su parte, los algoritmos NLP, como SNOPT, requieren los gradientes de las restricciones activas y la función de coste respecto a cada una de las variables de decisión. En caso de no ser suministrados, el optimizador los calcula con diferencias finitas, lo cuál no suele ser una buena aproximación [13]. Estos gradientes pueden calcularse reusando el Jacobiano que calculan los métodos de integración modernos como DASPK o IDAS [14].

Las principales desventajas de este enfoque son la imposibilidad de resolver problemas en los que hayan sistemas inestables y para tratar las restricciones de camino o calcular gradientes exactos si no se dispone de un algoritmo de cálculo de sensibilidades en el integrador. Técnicas como el "Multiple shooting", donde se divide el horizonte de tiempo en varios intervalos y se agregan los estados iniciales en cada uno de ellos a las variables de decisión, permiten dar una trayectoria inicial a la solución y facilitar el manejo de sistemas inestables [11].

1.5.2. Enfoque simultáneo

Además de los parámetros que varían en el tiempo, también se discretizan las ecuaciones diferenciales, preferiblemente usando un método de colocación ortogonal [15]. Así, se transforma el problema original en un sistema que solo contiene ecuaciones algebraicas. Posteriormente, se resuelve usando un algoritmo NLP. Esto se puede implementar

en entornos como GAMS ® o AMPL ® los cuales incorporan paquetes de diferenciación automática para el cálculo de gradientes.

A diferencia del enfoque secuencial, con el enfoque simultáneo es posible trabajar con sistemas inestables y, además, las restricciones no presentan ninguna dificultad pues basta con añadir las ecuaciones respectivas a cada punto de discretización. Sin embargo, esto tiene como consecuencia que el número de ecuaciones y variables aumenta enormemente. Además, la discretización podría dar problemas pues normalmente, incluso con colocación ortogonal, es menos exacta que los métodos de integración modernos que implementan los entornos de simulación.

1.6. Optimización no lineal

La programación no lineal, NLP por sus siglas en inglés, es el proceso de resolver un problema de optimización en el cual alguna de las restricciones, o la función de coste en si misma, son no lineales. La mayoría de los procesos son no lineales por su propia naturaleza, de hecho, los comportamientos completamente lineales son una rareza [1]. En esta sección se dará una breve introducción sobre como se pueden abordar este tipo de problemas de optimización.

Sea el problema de optimización mostrado en la ecuación (6), donde las ecuaciones $h(x)$ y $g(x)$ son restricciones de igualdad y desigualdad respectivamente. Estas pueden estar asociadas por ejemplo a límites físicos como que la apertura de una válvula no pueda superar el 100 %, a limites de operación como que la temperatura no pueda superar cierto valor, etc.

$$\begin{aligned} \min_{x} \quad & J(x) \\ \text{s.a.} \quad & h(x) = 0, \\ & g(x) \leq 0 \end{aligned} \tag{6}$$

Los multiplicadores de Lagrange proveen las condiciones necesarias que el optimo de un problema con restricciones de igualdad debe cumplir. Las condiciones de optimalidad necesarias de primer orden de los problemas con restricciones de igualdad establecen que la solución $(x*, \lambda*)$ del lagrangiano debe cumplir:

$$\begin{aligned} \nabla J(x^*) + \lambda^* \nabla h(x^*) = 0 \\ h(x^*) = 0 \end{aligned} \tag{7}$$

Sin embargo, lo contrario no es cierto. Si se cumple el criterio (7) no significa que haya un óptimo. Por otra parte, las condiciones Karush-Kuhn-Tucker (KKT) amplían el concepto de multiplicadores de Lagrange para establecer condiciones de optimalidad en problemas con restricciones de desigualdad. Lo que se hace es asignar un multiplicador a $g(x)$ pero estableciendo que $\mu g(x)$ sea igual a cero y que μ no sea negativo. Y así, las condiciones de optimalidad de primer orden KKT para problemas de optimización no lineales son:

$$\nabla J(x^*) + \lambda^* \nabla h(x^*) + \mu^* g(x^*) = 0$$

$$\begin{aligned} h(x^*) &= 0 \\ g(x^*) &\leq 0 \\ \mu^* g(x^*) &= 0 \\ \mu^* &\geq 0 \end{aligned} \tag{8}$$

Otra forma de abordar las restricciones de desigualdad es agregando variables "slack.º de relajación. Las cuales se añaden al vector de decisión como se muestra en (9).

$$\begin{aligned} \min_{x,\,\lambda,\,\mu,\,\epsilon} \quad & L(x,\lambda,\mu,\epsilon) = J(x) + \lambda h(x) + \mu(g(x)+\epsilon) \\ \text{s.a.} \quad & \epsilon \geq 0 \end{aligned} \tag{9}$$

Resolviendo el conjunto de ecuaciones (8) se puede hallar un posible óptimo del problema general (6). Es importante resaltar el hecho que se requieren los gradientes de la función de coste y las restricciones respecto a cada variable de decisión. La forma más intuitiva de calcularlos sería derivando simbólicamente las expresiones originales. Aunque parece sencillo, en problemas de gran magnitud puede ser complicado, e imposible en aquellos que hayan discontinuidades. Además, las expresiones resultantes son de considerable magnitud y, por tanto, su evaluación es muy lenta. Otra forma de calcularlos es mediante diferencias finitas pero la precisión no siempre es buena y en caso de señales ruidosas da problemas. Por último, el método más eficiente y preciso de todos es la diferenciación automática aunque no todos los entornos de optimización lo incluyan.

1.6.1. Programación cuadrática

Mención especial merecen los casos de programación cuadrática (QP), problemas de optimización no lineales en los que la función de coste es cuadrática como se ve en (10). Este tipo de formulaciones son muy comunes y además se suelen utilizar en algoritmos NLP como SNOPT [13].

$$\begin{aligned} \min_{x} \quad & J(x) = c^\mathsf{T} x + \frac{1}{2} x^\mathsf{T} Q x \\ \text{s.a.} \quad & Ax = b, \\ & x \geq 0 \end{aligned} \tag{10}$$

Si la matriz Q es positiva semi-definida, la función de coste $J(x)$ es convexa. La región factible también es convexa pues las restricciones son todas lineales. Por tanto, el problema cuadrático (10) es convexo y cualquier solución local también será global [13]. De esta forma, la solución que cumpla las condiciones de KKT (11) será un optimo global. Por otra parte, los problemas QP tienen la particularidad de que requieren pocas iteraciones para converger.

$$\begin{aligned} c + Qx + A^\mathsf{T}\lambda - \mu &= 0 \\ Ax - b &= 0 \\ x &\geq 0 \end{aligned}$$

$$\mu \geq 0$$
$$\mu^{\mathsf{T}} x = 0 \tag{11}$$

Prácticamente todos los algoritmos NLP de uso general comerciales incorporan paquetes específicos para resolver el sistemas de ecuaciones (11). Por ejemplo, SNOPT incorpora un paquete llamado SQOPT dedicado a resolver problemas QP mediante un algoritmo de conjuntos activos [13].

1.6.2. Programación cuadrática secuencial

Los métodos de programación cuadrática secuencial, o sucesiva, (SQP) resuelven una secuencia de aproximaciones de problemas no lineales a problemas cuadráticos. Aprovechando así las características particulares de los problemas QP y los algoritmos eficientes que existen para ellos [1]. La función de coste es una aproximación cuadrática del Lagrangiano del problema original, y las restricciones son igualmente linealizaciones de las originales. SNOPT particularmente se basa en este tipo de formulación para resolver problemas de optimización no lineales. Explicar en detalle como funciona el algoritmo está fuera del alcance de este trabajo pero de forma muy resumida lo que se hace es formular un Lagrangiano de la función de coste original y obtener una dirección de búsqueda resolviendo una aproximación cuadrática del mismo, con las restricciones linealizadas. Posteriormente, se calcula la longitud del paso en esa dirección con una función de mérito. El procedimiento se repite hasta que se alcancen los criterios de parada. El detalle de como funciona SNOPT se puede ver en [13]. Este algoritmo será el que se usará para resolver los problemas de optimización que se planteen en la etapa de estimación de parámetros y en la formulación del controlador predictivo.

Capítulo 2

Descripción de la planta

La planta piloto que se usó para desarrollar este trabajo pertenece al Laboratorio de Control de Procesos del Departamento de Ingeniería de Sistemas y Automática (DISA) de la Universidad de Valladolid. Su principal característica es que aunque tiene todos los elementos de un reactor químico, la reacción está simulada y la única sustancia involucrada es agua. Una simulación de la reacción química se alimenta con medidas reales del proceso para calcular el calor que generaría la reacción, que luego se aplica mediante resistencias eléctricas. En este capítulo se describirán los elementos físicos que la conforman, las comunicaciones necesarias para su funcionamiento y la forma en la que está implementada la simulación de la reacción. Es importante resaltar que esta planta es el resultado de varios trabajos de investigación y, también, usa varias herramientas desarrolladas dentro por el DISA. Aunque el desarrollo e implementación del controlador predictivo y el optimizador en tiempo real se limitará a una simulación completa de la planta, todas las interfaces de comunicación requeridas con el proceso real fueron tenidas en cuenta de tal forma que la transición en un futuro sea sencilla.

En las figuras 6 y 7 se muestran un diagrama P&ID de la planta y un esquema general de la misma. Mientras que una foto del reactor se muestra en la figura 8

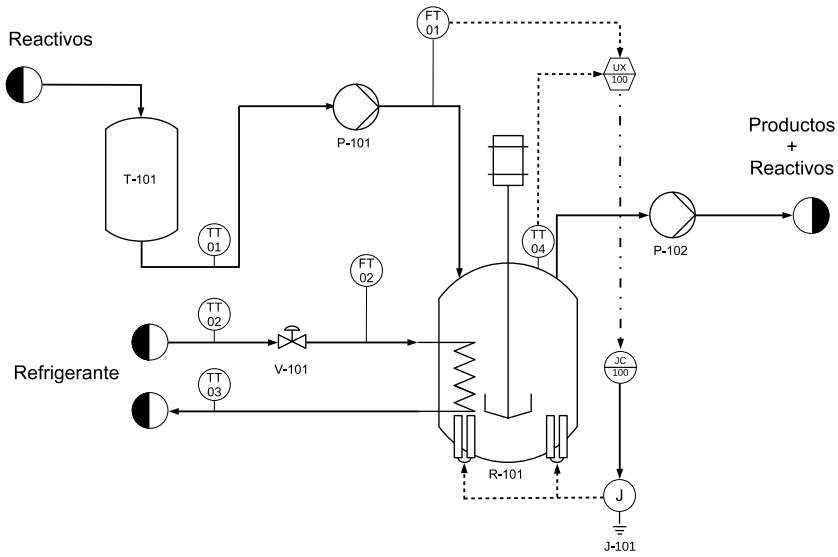

Figura 6: Diagrama de instrumentación de la planta.

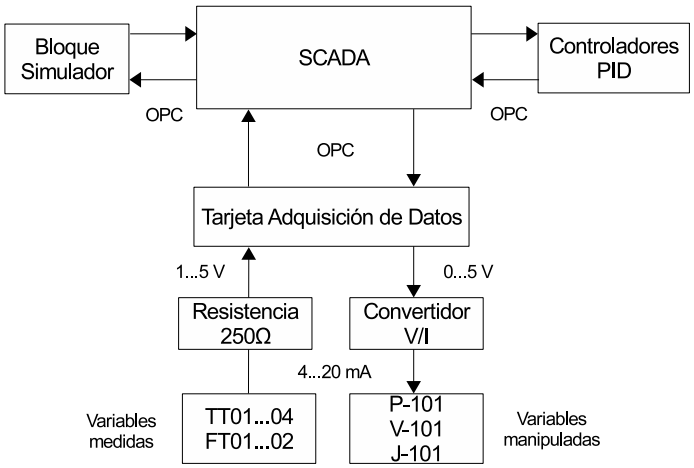

Figura 7: Esquema general de la planta.

2.1. Plantas piloto híbridas

Las ventajas teóricas de técnicas como el control avanzado, la detección de fallos, la reconciliación de datos, etc. deben ser probadas en plantas reales previo a su implementación final. Sin embargo, tales plantas tienen una complejidad inherente que hace que la experimentación pueda resultar extremadamente cara. El riesgo de perder producto siempre está presente y los cambios admisibles en las variables manipuladas podrían ser tan pequeños como para obtener información aportante. En el otro extremo, están las herramientas de simulación que requieren modelos matemáticos, que no siempre están disponibles. Y de estarlo, es frecuente que se requieran simplificaciones significativas que no representan adecuadamente las plantas reales.

Las plantas piloto tienen mucha similitud con las plantas reales pero suelen ser más sencillas y tener menos instrumentación. A pesar de ser simplificaciones de las reales, las plantas pilotos involucran materias primas y productos que podrían llegar a ser caros y peligrosos. Además, requieren de mucha preparación y mantenimiento. En este punto aparece el concepto de plantas pilotos híbridas, plantas en las que una parte del proceso es simulada mientras que algunas características, como la hidrodinámica, se mantienen fieles a realidad [16]. Este enfoque solo es posible si las propiedades de los fluidos no varían considerablemente con la concentración de solutos. Los fluidos se cambian por agua u otros que tengan propiedades similares a los del proceso real.

En el caso de un reactor químico como el objeto de este trabajo, la reacción se simula en tiempo real para calcular variables de salida virtuales como la concentración de las especies, a partir de variables de entrada reales como la temperatura del reactor y los caudales de reactivos y refrigerante.

No solo son útiles las plantas piloto híbridas en investigación, sino que también lo son en la enseñanza. Su implementación reduce significativamente la inversión que deben hacer las universidades para que los estudiantes tengan acceso a procesos con dinámicas relativamente complejas. Y, además, reducen el riesgo inherente a sustancias químicas, al ser el agua el único fluido involucrado.

Figura 8: Foto del reactor.

2.2. Diseño y construcción

La planta fue diseñada en un principio por Sergio Sanz en su trabajo de investigación «Diseño de planta piloto híbrida y control de la misma» [17]. En esta etapa, se seleccionaron tres bombas, una válvula de control y dos caudalímetros. En su trabajo se pueden ver las ofertas económicas de diferentes proveedores así como también las fichas técnicas de cada uno de los equipos. Los elementos, aunque fueron ordenados y comprados, no fueron integrados.

Posteriormente y previo a la construcción de la planta, Juan Manuel Gordo en su TFG «Control de una planta piloto de intercambio de calor» estudió por medio de simulaciones la controlabilidad de la planta y la pertinencia de los elementos considerados en el diseño inicial de Sergio Sanz [18]. También se simularon diferentes estrategias de control, todos ellas basadas en controladores PID. Se demostró que los equipos seleccionados eran adecuados a pesar de la limitación de potencia eléctrica de 9kW de la red eléctrica del laboratorio.

María Marcos en su TFM «Implementación de una planta piloto híbrida, desarrollo de su sistema de control y estudio de su operación óptima» integró todos los elementos que conformarían la planta e implementó estrategias de control digital [19]. Además, desarrolló una aplicación SCADA en el software InTouch ® que permite manipular cada uno de los elementos del sistema y controlar la planta. Su trabajo sirvió como punto de partida para la presente investigación. Se han introducido una serie de cambios tanto en el software como en la instrumentación, el más significativo de los cuales fue la implementación de las resistencias eléctricas directamente en el depósito y no en un tanque aparte. Esto permite aproximarse más a la dinámica de un reactor químico.

2.3. Elementos físicos

El elemento principal de la planta piloto es el depósito R-101 que hace la función de reactor CSTR. Los reactivos se almacenan en el depósito T-101 y se alimentan al reactor mediante la bomba dosificadora de velocidad variable P-101. El refrigerante va por un serpentín instalado en el interior del reactor y su flujo se controla manipulando la apertura de la válvula V-101. La toma de la bomba de productos P-102 es fija para asegurar que el nivel en el reactor es constante. El reactor tiene instalado en su interior dos resistencias eléctricas que son alimentadas por el amplificador J-101. La comunicación de las señales entre el ordenador y los dispositivos de campo está gestionada por una tarjeta de adquisición de datos.

La configuración de la planta se completa con un agitador de velocidad variable, cuatro transmisores de temperatura (TT01-04) y dos caudalímetros magnéticos (FT01-02).

2.3.1. Equipos

La planta está conformada de los siguientes equipos:

- P-101: Bomba peristáltica Masterflex L/S 6-600, con un caudal máximo de 1.5 L/min a 600rpm.
- P-102: Bomba de membrana Membos LB150. Velocidad fija y temperatura máxima de operación de 60ºC.
- R-101: Depósito en acero inoxidable AISI 316 fabricado a medida durante el desarrollo de este proyecto.
- V-101: Válvula neumática de asiento normalmente abierta Fisher NPS 1 GX. Tiene un controlador de posición Fieldve DVC200 que permite ajustar el porcentaje de apertura.
- J-101: Unidad de potencia trifásica Desin Instruments UPTF 380. Alimenta a las dos resistencias trifásicas de 3 kW instaladas en el fondo del reactor R-101.

2.3.2. Instrumentación

Para medir en tiempo real las variables importantes del proceso, la planta cuenta con los siguientes instrumentos:

- Dos caudalímetros electromagnéticos. Para los reactivos un Kobold MIK L443 con un rango de medición de 0.16 a 3.2 L/min. Para el refrigerante un Emerson Rosemount 8732 con un rango de 0.83 a 25.02 L/min.
- Cuatro sondas de temperatura del tipo PT100. Estas tienen un rango de medición de 0 a 100ºC.

Todos los instrumentos de la planta son analógicos por lo que se debe transformar su señal. En el anexo A.2 se resumen las ecuaciones usadas para tal fin.

2.3.3. Tarjeta de adquisición de datos

Para recoger todas las señales de los instrumentos y para enviar las señales de las acciones de control, se usa una tarjeta de adquisición de datos MCC USB-1208HS-4A0. Esta lee y escribe señales en un rango de 0-5 V por lo que se debe hacer una conversión adicional. La señal de los instrumentos de medición se convierte de 4-20 mA a 1-5 V agregando una resistencia de 250 Ω a la conexión de los hilos de los mismos con la tarjeta. Por otra parte,

se usa un convertidor V/I FEMA ISC-P para convertir las señales de salida de 0-5 V a 4-20 mA. En la figura 9 se puede ver una foto de instalación de estos elementos. La tarjeta de adquisición de datos se accede desde el ordenador mediante un servidor OPC desarrollado por Marcos Peréz en su trabajo de fin de grado [20].

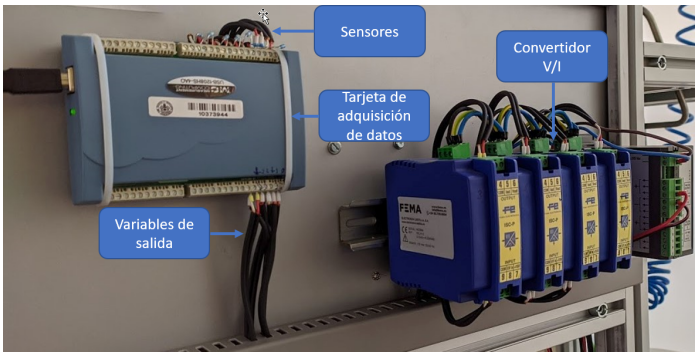

Figura 9: Tarjeta de adquisición de datos y convertidor de señal

2.4. Elementos software

Todos los elementos físicos de la planta deben estar acompañados de software que permiten recoger su información y gobernarlos en el caso de los actuadores. Hay principalmente tres componentes software que son indispensables para el comportamiento de la planta: Un servidor OPC que contiene la simulación de la reacción química, servidores OPC que contienen controladores PID digitales y un sistema SCADA.

2.4.1. Simulación de la reacción química

Originalmente, la planta disponía de un bloque de simulación en el que estaba programada una única reacción de primer orden. Se decidió programar las reacciones de Van de Vusse, conocida por tener dinámicas altamente no lineales [21]. Además, pueden estudiarse los efectos de diferentes estrategias de control en la selectividad y otras aspectos económicos como el coste de separación de las especies.

$$\mathrm{A} \xrightarrow{\mathrm{k_1}} \mathrm{B} \xrightarrow{\mathrm{k_2}} \mathrm{C}$$
$$2\,\mathrm{A} \xrightarrow{\mathrm{k_3}} \mathrm{D}$$

Suponiendo un reactor CSTR perfectamente agitado, la variación de la concentración de cada especie es igual a lo que entra menos lo que sale del sistemas más lo que se genera menos lo que se consume. La especie A se consume a través de las reacciones 1 y 3, mientras que B se genera a través de la reacción 1 y consume en 2. Las especies C y D solo se generan, a través de las reacciones 2 y 3 respectivamente. Esto se resume en las ecuaciones (12)-(15).

$$V\frac{dC_A}{dt} = q(C_{A0} - C_A) - r_1 - 2r_3 \tag{12}$$

$$V\frac{dC_B}{dt} = -qC_B + r_1 - r_2 \tag{13}$$

$$V\frac{dC_C}{dt} = -qC_C + r_2 \tag{14}$$

$$V\frac{dC_D}{dt} = -qC_D + r_3 \tag{15}$$

La velocidad de cada una de las reacciones se formula como fue establecido originalmente por Van de Vusse [21] y se presenta en las ecuaciones (16)-(18). Los valores de los parámetros cinéticos no están establecidos en la literatura por lo que hay bastante libertad para seleccionarlos dependiendo del comportamiento que se desee estudiar con las reacciones.

$$r_1 = k_{10}e^{-\frac{E_1}{RT}}C_A V \tag{16}$$

$$r_2 = k_{20}e^{-\frac{E_2}{RT}}C_B V \tag{17}$$

$$r_3 = k_{30}e^{-\frac{E_3}{RT}}C_A^2 V \tag{18}$$

Por último, el calor desprendido por las reacciones se calcula como la sumatoria del calor aportado por cada reacción individualmente. De nuevo, el valor de los calores de reacción se puede elegir libremente.

$$Q_{rxn} = -V(-r_1\Delta H_{rxn,1} - r_2\Delta H_{rxn,2} - r_3\Delta H_{rxn,3}) \tag{19}$$

Las ecuaciones (12)-(19) se programaron en el entorno de simulación EcosimPro, el código se puede encontrar en el anexo A.3. Luego de compilar la simulación, esta se envuelve en una capa OPC-DA con ayuda del propio software. Así, es posible resolver las ecuaciones en tiempo real utilizando datos medidos del proceso, temperaturas y caudales. De modo que permiten calcular las concentraciones de los productos y el calor desprendido si tuvieran lugar realmente las reacciones. Algunos de estos valores de las concentraciones se usan como medidas en tiempo real y el calor desprendido calculado se aplica realmente al proceso. La potencia es convertida por el SCADA en una señal en voltios que es enviada luego al amplificador.

2.4.2. Controladores PID

La planta tiene implementados controladores PID digitales accesibles como servidores OPC, también desarrollados en EcosimPro, denominados PID ISA [22]. Para el desarrollo del presente trabajo, se reconfiguraron para que controlaran directamente los flujos de reactivos y refrigerante pues originalmente controlaban la temperatura y la concentración. En la figura 10 se muestra la interfaz gráfica de PID ISA.

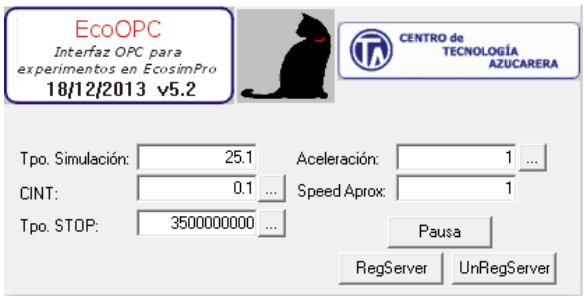

Figura 10: Interfaz del controlador digital PID ISA.

2.4.3. SCADA

El sistema SCADA con el que cuenta la planta permite visualizar la información de todos los sensores así como manipular los diferentes actuadores [19]. Además, es posible gestionar e intervenir los controladores PID y modificar los parámetros de la simulación de la reacción química. Por otra parte, gestiona todas las comunicaciones entre los diferentes servidores OPC-DA de una forma transparente al usuario.

Teniendo en cuenta las diferentes modificaciones realizadas a la planta en el desarrollo de este trabajo, se debió actualizar el SCADA original. De forma resumida, se eliminó todo lo referente al tanque calefactor, se reconfiguraron los controladores PID, se retocó el diagrama del proceso y se añadió una ventana para configurar el controlador predictivo. En la figura 11 se muestra como luce actualmente la ventana principal de la interfaz HMI del SCADA. En un diagrama de flujo del proceso se representan las medidas medidas de cada uno de los sensores. Además, se pueden manipular la bomba de reactivos y la válvula de refrigerante. También es posible encender o apagar los controladores de caudal y la simulación de la reacción química. Con esta última desactivada, el sistema se comporta simplemente como un tanque calefactado. El trabajo de María Marcos contiene una descripción detalla del sistema [19].

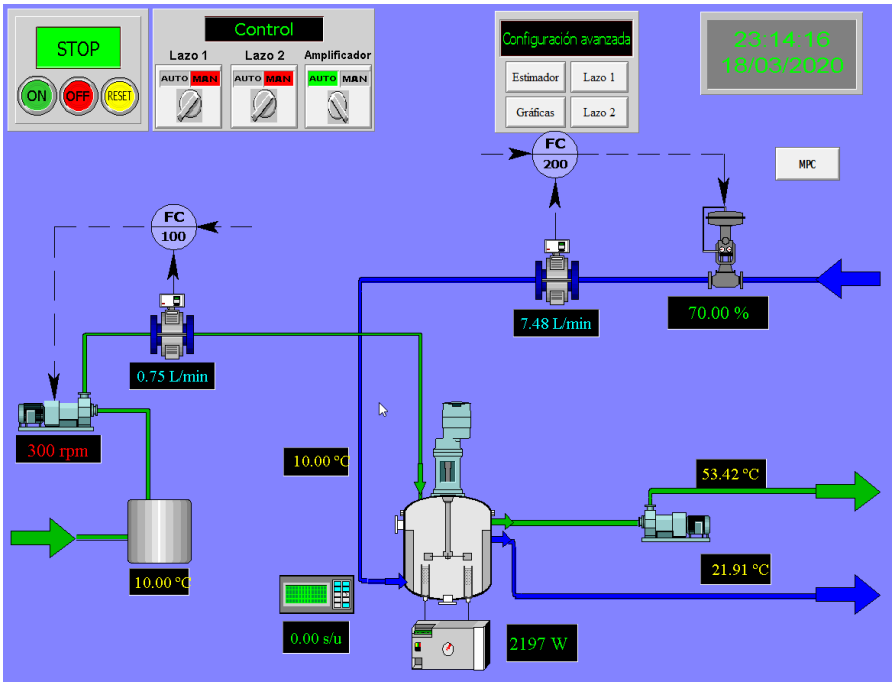

Figura 11: Interfaz HMI del SCADA.

De particular interés es la ventana de configuración del bloque simulador como se muestra en la figura 12. Se pueden modificar individualmente los parámetros cinéticos del conjunto de reacciones químicas simuladas, así como cambiar entre una reacción sencilla y la reacción de Van de Vusse.

2.4.4. Registro y almacenamiento de datos

El sistema SCADA por si mismo genera un servidor OPC que contiene las variables de los otros servidores con los que establece comunicación. Esto es muy útil pues permite

Figura 12: Ventana de configuración del bloque simulador.

recoger en un solo lugar tanto los datos de los sensores como las variables de la simulación. Finalmente, los datos se almacenan estableciendo una conexión con este servidor desde el software ReadOPC v1.6. El fichero de datos resultante se procesa con un script de Python, desarrollado para este proyecto, que lo convierte a un archivo ".csv". El código fuente se puede ver en el anexo A.5.

2.5. Simulación de la planta a través de OPC

Con el fin de poder probar diferentes algoritmos y estrategias sin necesidad de poner en marcha la planta, se preparó una simulación que engaña al SCADA enviándole las señales que recibiría de la tarjeta de adquisición de datos. El esquema que se presentó en la figura 7 se modifica por el de la figura 13. La nueva simulación, embebida en una capa OPC, reemplaza al proceso real y a la tarjeta de adquisición de datos. De hecho, la simulación usa las ecuaciones de conversión para enviar las variables en valores de voltios.

Para representar adecuadamente el comportamiento de la planta real, esta nueva simulación solo contiene los balances de energía del reactor y el serpentín, la reacción se sigue simulando de la misma manera en que se haría con el proceso real. Es decir, simula la parte real de la planta. El código se encuentra en el anexo A.4.

Este enfoque también tiene desventajas. Aunque es posible agregar perturbaciones modificando la temperatura de entrada de los reactivos y el refrigerante, es prácticamente imposible hacerlo de la misma manera en la que sucede en la realidad. Además, los modelos nunca son cien por ciento exactos. Pueden existir diferencias más o menos significativas dependiendo del ajuste que se haga con datos experimentales.

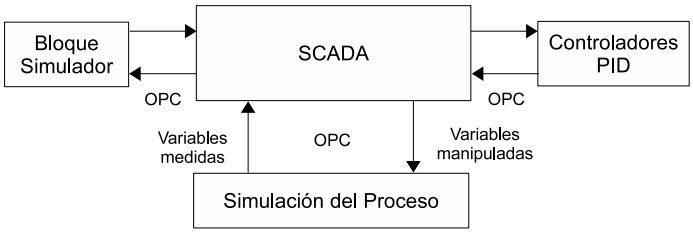

Figura 13: Esquema general de la planta en simulación.

Capítulo 3

Construcción de un modelo

Se requiere un modelo matemático de la planta que describa su comportamiento dinámico. Debe ser útil para predecir el comportamiento de cada una de las variables del proceso ante cambios en el flujo y temperatura de reactivos y refrigerante. Además, debe ser válido en prácticamente todo el rango de operación de la planta.

Se decidió construir un modelo de base fenomenológica. Se supondrá que las propiedades físicas del agua, tanto del lado del reactor como del serpentín, permanecen constantes y no cambian con la temperatura. De la misma forma, se supone que el volumen del reactor es constante y que está perfectamente agitado.

Luego de ser formulado, se ajustará el modelo estimando los parámetros desconocidos a partir de datos experimentales del proceso. Para esto será necesario desarrollar y ejecutar un experimento en el que se manipulen los caudales de reactivos y refrigerante. Por último, se validará el modelo con un conjunto de datos diferentes a los usados para la estimación.

3.1. Formulación matemática

Se modelará la planta como un reactor de tanque continuamente agitado. Además, se supondrá que las reacciones químicas son conocidas, pero no sus parámetros. En caso de que las reacciones fueran desconocidas, se debería usar otro enfoque. Por ejemplo, estimar directamente el valor de la velocidad de la reacción a lo largo del tiempo y luego ajustar alguna ecuación a la misma. En todo caso, la reacción de van der Vusse se describe con las siguientes ecuaciones:

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \xrightarrow{k_2} \mathrm{C}$$
$$2\,\mathrm{A} \xrightarrow{k_3} \mathrm{D}$$

3.1.1. Balances de materia

Se usarán las mismas ecuaciones que ya están programas en el bloque simulador, aunque podría usarse cualquier otra metodología adecuada. El balance de materia de cada especie consiste en calcular el cambio de la concentración como la cantidad que entra menos la que sale, más lo que se genera y menos lo que se consume. La especie A se consume mediante las reacciones 2 y 3. Mientras que B se genera a través de la reacción 1 pero se consume en la 2. A su vez, las especies C y D se generan con las ecuaciones 2 y 3 respectivamente. Los balances de cada una de las ecuaciones se ven en las ecuaciones (20)-(23).

$$V\frac{dC_A}{dt} = q(C_{A0} - C_A) - r_1 - 2r_3 \tag{20}$$

$$V\frac{dC_B}{dt} = -qC_B + r_1 - r_2 \tag{21}$$

$$V\frac{dC_C}{dt} = -qC_C + r_2 \tag{22}$$

$$V\frac{dC_D}{dt} = -qC_D + r_3 \tag{23}$$

3.1.2. Balances de energía

Las ecuaciones de los balances de energías representan la parte del proceso que no está siendo simulada. La temperatura del reactor se ha supuesto homogénea, y se modela como tal. Se incluye el término de generación por la reacción química y el de transferencia de calor con el serpentín. La temperatura del serpentín también se modela como si fuese homogénea pues el flujo de refrigerante es lo suficientemente grande como para que los gradientes no sean significativos. Los balances de energía del reactor y del serpentín se muestran en las ecuaciones (24) y (25).

$$\rho C_p V\frac{dT}{dt} = q\rho C_p(T_0 - T) - Q + Q_{rxn} \tag{24}$$

$$\rho C_p V_c\frac{dT_c}{dt} = q_c\rho C_p(T_{c0} - T_c) + Q \tag{25}$$

3.1.3. Cinética de la reacción

Como se ha supuesto que las reacciones químicas son conocidas, también lo es la estructura de la cinética. Se toman para el modelo las mismas ecuaciones de la ley de Arrhenius programadas en el bloque simulador como se muestra en (26)-(29). Sin embargo, los factores pre-exponenciales, las energías de activación y las entalpías de reacción se suponen desconocidos.

$$r_1 = k_{10}e^{-\frac{E_1}{RT}}C_A V \tag{26}$$

$$r_2 = k_{20}e^{-\frac{E_2}{RT}}C_B V \tag{27}$$

$$r_3 = k_{30}e^{-\frac{E_3}{RT}}C_A^2 V \tag{28}$$

$$Q_{rxn} = -V\sum_{i=1}^{3} r_i \Delta H_{rxn,i} \tag{29}$$

3.1.4. Transferencia de calor

La transferencia de calor entre el reactor y el serpentín se modela usando un coeficiente de transferencia de calor multiplicado por el gradiente de temperatura como se ve en la ecuación (30).

$$Q = UA(T - T_c) \tag{30}$$

Como es sabido, el coeficiente global de transferencia de calor se calcula como la resistencia térmica total por convección a ambos lados de la interfaz de transferencia y por la conducción a través de la pared. Si se desprecian efectos de ensuciamiento, el coeficiente de transferencia tiene la forma:

$$\frac{1}{U \cdot A} = \frac{1}{h_{reactor} \cdot A} + \frac{L}{k \cdot A} + \frac{1}{h_{serpentin} \cdot A} \tag{31}$$

Donde U representa el coeficiente global de transferencia de calor, A el área del serpentín, $h_{reactor}$ y $h_{serpentin}$ los coeficientes de transferencia por convección en el lado del reactor y el serpentín respectivamente, L el espesor de la pared del serpentín y k la conductividad del serpentín.

Los coeficientes de convección pueden ser calculados con correlaciones empíricas disponibles en la literatura como sigue [23, 24]:

$$h_{reactor} = 0.87 \left(\frac{L_{agitador}^2 N \rho}{\mu} \right)^{2/3} \left(\frac{C_p \mu}{k_{agua}} \right)^{1/3} \tag{32}$$

$$h_{serpentin} = 0.023 \frac{k_{agua}}{L_{serpentin}} \left(\frac{\rho v D}{\mu} \right)^{4/5} \left(\frac{C_p \mu}{k_{agua}} \right)^{0.3} \tag{33}$$

Donde $L_{agitador}$ y N representan la longitud y la velocidad del agitador respectivamente, ρ, μ, C_p, k_{agua} son la densidad, la viscosidad, la capacidad calorífica y la conductividad térmica del agua. Por otra parte, D es el diámetro interior del serpentín y v la velocidad del refrigerante.

Sería necesario conocer cada uno de los parámetros que aparecen en las ecuaciones (32) y (33) para poder calcular el coeficiente de transferencia. Cómo la mayoría de ellos son desconocidos, es más conveniente estudiar la dependencia con cada una de las variables. La longitud y la velocidad del agitador, la densidad, la conductividad térmica y la capacidad calorífica del agua, el diámetro y la longitud del serpentín permanecen aproximadamente constantes durante los experimentos. Las únicas variables que cambian en gran medida son la velocidad del agua en el serpentín y la viscosidad de la misma. De 20ºC a 60ºC, la viscosidad del agua disminuye alrededor de un 50 %. Mientras que en un experimento, el caudal de agua por el serpentín puede variar de 3 a 15 L/min. Por tanto, se considera mucho más significativo el efecto del cambio de la velocidad.

Si se agrupan todos los términos asumidos como constantes en uno solo, incluyendo los factores de conversión, se puede definir la ecuación (34) para calcular el coeficiente global de transferencia de calor. El parámetro α deberá ajustarse con datos experimentales para que se puedan obtener estimaciones que representen adecuadamente la realidad. A partir de los datos generados por el pre-experimento presentado en el anexo A.1, se toma $1.22\ kJ/min^{0.2}/L^{0.8}/K$ como valor inicial para este parámetro.

$$UA = \alpha q_c^{4/5} \tag{34}$$

Aunque es una simplificación, y no refleja el cambio de las propiedades con la temperatura, la ecuación (34) tiene en cuenta como varia el coeficiente global de transferencia de calor con la variable más importante, el caudal de refrigerante.

3.2. Estimación de parámetros

Una de las etapas más importantes, y tal vez la más difícil, en la construcción de un modelo matemático, es ajustar y obtener los valores de los parámetros desconocidos de forma tal que la dinámica del modelo se asemeje lo más posible a la del proceso que representa. Aunque existen diversas técnicas para tal fin, se decidió usar la técnica de estimación de parámetros por optimización dinámica. Dicha técnica consiste en minimizar una función de coste que depende de la diferencia entre la respuesta del modelo y datos experimentales ante ciertas variables de entrada, teniendo como variables de decisión los parámetros del modelo.

Todos los parámetros del modelo desarrollado en las secciones anteriores se presentan en la tabla 1. Aunque los parámetros relacionados con la cinética de las reacciones se preestablecen en el SCADA, se supondrán desconocidos como sería el caso si la reacción no estuviera simulada. Así, todos los parámetros relacionados con las reacciones han de ser estimados. Además, el factor α del coeficiente de transferencia de calor también es desconocido y su valor deberá estimarse para ajustar el modelo al proceso.

Parámetro	¿Conocido?	Valor
V	✓	$11.5\,\mathrm{L}$
V_c	✓	$1\,\mathrm{L}$
ρ	✓	$1000\,\frac{\mathrm{kg}}{\mathrm{m}^3}$
C_p	✓	$4.186\,\frac{\mathrm{kg}}{\mathrm{kJ\,K}}$
k_{10}	✗	-
k_{20}	✗	-
k_{30}	✗	-
$\Delta H_{rxn,1}$	✗	-
$\Delta H_{rxn,2}$	✗	-
$\Delta H_{rxn,3}$	✗	-
E_1	✗	-
E_2	✗	-
E_3	✗	-
α	✗	-

Tabla 1: Parámetros del modelo.

3.2.1. Diseño del experimento

Para capturar la información dinámica del proceso es necesario diseñar de forma adecuada un experimento. En este se debe estudiar la respuesta del sistema en distintas situaciones de funcionamiento, por ejemplo, a altas y bajas temperaturas, con flujos de refrigerante muy grandes y muy pequeños, etc. Además, el proceso debe ser operado en lazo abierto, pues en caso contrario, los controladores intentarán mantener el proceso lo más estable posible, sesgando los datos. Teniendo todo esto en cuenta, se planea el experimento cambiando los caudales de reactivos y de refrigerantes durante cinco horas como se muestra en la figura 14. Es importante recordar que la planta cuenta con sensores para todas las temperaturas involucradas en el modelo y, se supondrá, que se tiene acceso a las medidas de concentración de las especies B y D.

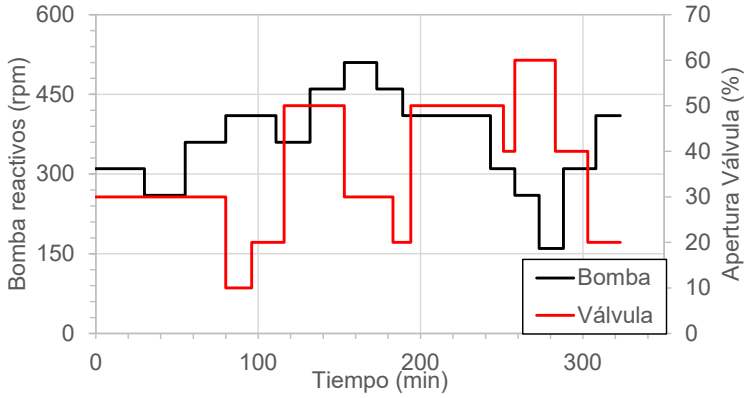

Figura 14: Diseño del experimento de ajuste del modelo.

Se desarrolló un script en Python que ejecuta los cambios del experimento de forma automática, evitando así retroalimentaciones del ejecutor y facilitando el trabajo. Su funcionamiento consiste en generar un cliente OPC que envía a la tarjeta de adquisición de datos señales para la bomba de reactivos y la válvula de refrigerante de acuerdo al planteamiento del experimento. El código fuente se muestra en el anexo A.6. Los resultados del experimento se muestran en línea discontinua en las figuras 15 y 16.

3.2.2. Problema de optimización

Como se mencionó anteriormente, se formuló la estimación de parámetros como un problema de optimización dinámica. En la ecuación (35) se minimiza la diferencia entre las variables de estado del modelo matemático del proceso y las medidas reales ajustando el valor de los parámetros.

$$
\begin{aligned}
\min_{\substack{k_i,E_i,\\ \Delta H_{rxn,i},\alpha}} \quad & J = c^2 \sum_{i=1}^{4} \left[\gamma_i \frac{|y_{m,i} - y_{p,i}|}{c\hat{y}_{p,i}} - \ln\left(1 + \gamma_i \frac{|y_{m,i} - y_{p,i}|}{c\hat{y}_{p,i}}\right)\right] \\
\text{s.a.} \quad & V\frac{dC_A}{dt} = q(C_{A0} - C_A) - k_{10}e^{-\frac{E_1}{RT}}C_A V - 2k_{30}e^{-\frac{E_3}{RT}}C_A^2 V, \\
& V\frac{dC_B}{dt} = -qC_B + k_{10}e^{-\frac{E_1}{RT}}C_A V - k_{20}e^{-\frac{E_2}{RT}}C_B V, \\
& V\frac{dC_C}{dt} = -qC_C + k_{20}e^{-\frac{E_2}{RT}}C_B V, \\
& V\frac{dC_D}{dt} = -qC_D + k_{30}e^{-\frac{E_3}{RT}}C_A^2 V, \\
& \rho C_p V\frac{dT}{dt} = q\rho C_p(T_0 - T) - \alpha q_c^{4/5}(T - T_c) - V\sum_{i=1}^{3} r_i \Delta H_{rxn,i}, \\
& \rho C_p V_c\frac{dT_c}{dt} = q_c\rho C_p(T_{c0} - T_c) + \alpha q_c^{4/5}(T - T_c), \\
& 10 \leq T, T_c \leq 60,
\end{aligned}
\tag{35}
$$

$$
\begin{aligned}
&0 \leq C_A \leq 5, && \\
&0 \leq k_i \leq 20 && i = 1, 2, 3, \\
&0 \leq E_i \leq 20 && i = 1, 2, 3, \\
&-40 \leq \Delta H_{rxn,i} \leq 0 && i = 1, 2, 3, \\
&0 \leq \alpha \leq 5 &&
\end{aligned}
$$

Se resolvió el problema en EcosimPro con un enfoque secuencial, el código se muestra en el anexo A.7. Se calcula la función de coste en el simulador y luego se envía al optimizador externo, SNOPT. Este devuelve nuevos valores de los parámetros para recalcular la función de coste. El ciclo se repite hasta que se cumple cierta tolerancia o hasta que no se encuentran valores de los parámetros que mejoren la función de coste. En la sección 1.5 se ha explicado en detalle este procedimiento.

En las figuras 15 y 16 se muestra gráficamente el ajuste de las variables del modelo a los datos experimentales con los parámetros estimados. Se puede apreciar que el ajuste de las temperaturas y las concentraciones es muy bueno en todo el rango de operación estudiado. Aunque este resultado es muy prometedor, hay que tener en cuenta que los datos fueron obtenidos por simulación y el ajuste con el proceso real no será perfecto. En la tabla 2 se presentan los valores obtenidos de los parámetros. Es evidente que son muy diferentes de lo que deberían ser. Esto no es un error, como se mencionó en la sección 1.6 los problemas de optimización no-lineales suelen ser no convexos y, por tanto, no está garantizado que sus soluciones sean globales. El conjunto de valores obtenidos es un mínimo local de la función de coste, pero gráficamente es evidente que con ellos el modelo tiene una respuesta prácticamente idéntica a los datos experimentales.

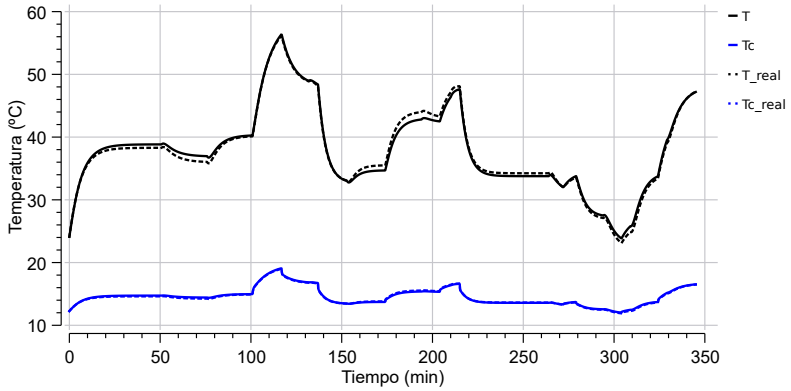

Figura 15: Ajuste de temperatura de reactivos y serpentín.

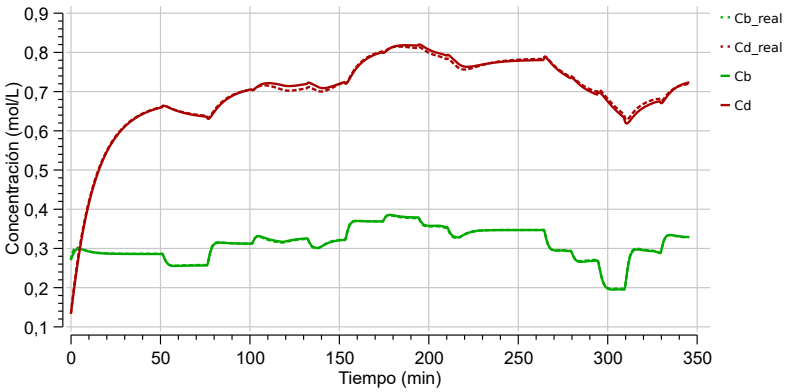

Figura 16: Ajuste de concentración de B y D.

	Real	Estimado
k_{10} $[\frac{1}{\mathrm{L\,min}}]$	15.60	20
k_{20} $[\frac{1}{\mathrm{L\,min}}]$	13.40	9.67
k_{30} $[\frac{1}{\mathrm{mol\,min}}]$	3.20	10.93
$\Delta H_{rxn,1}$ $[\frac{\mathrm{kJ}}{\mathrm{mol}}]$	-2	-4.633
$\Delta H_{rxn,2}$ $[\frac{\mathrm{kJ}}{\mathrm{mol}}]$	-5	-3.670
$\Delta H_{rxn,3}$ $[\frac{\mathrm{kJ}}{\mathrm{mol}}]$	-7	-1.147
E_1 $[\frac{\mathrm{kJ}}{\mathrm{mol}}]$	8.09	8.920
E_2 $[\frac{\mathrm{kJ}}{\mathrm{mol}}]$	7.59	6.767
E_3 $[\frac{\mathrm{kJ}}{\mathrm{mol}}]$	5.21	8.721
α $[\frac{\mathrm{kJ}}{\mathrm{min\,°C}}]$	1.220	1.229

Tabla 2: Resultados de la estimación de parámetros.

3.3. Validación

Aunque existen multitud de técnicas para validar modelos, no hay una prueba definitiva que permita decir con seguridad que un modelo es válido. Lo que si es posible es establecer un cierto grado de confianza dependiendo de las pruebas realizadas. Para validar el modelo construido en la sección anterior, se ejecutará un nuevo experimento donde se aplicarán cambios de las variables manipuladas diferentes a los usados para la estimación de parámetros, al proceso y al modelo y se compararán sus respuestas tanto cualitativa como cuantitativamente. El diseño de este nuevo experimento se muestra en la figura 17. Los valores de caudales de refrigerante y reactivos se generaron de forma aleatoria para evitar retroalimentaciones del experimento original. Los resultados se muestran en las figuras 18 y 19 donde las líneas discontinuas representan las variables medidas de la planta.

A partir de las figuras 18 y 19 se puede ver que el ajuste del modelo es muy bueno a los datos experimentales, incluso en condiciones diferentes a las que se usaron para estimar los parámetros. En consecuencia, se da el modelo por válido para los rangos normales de operación de la planta.

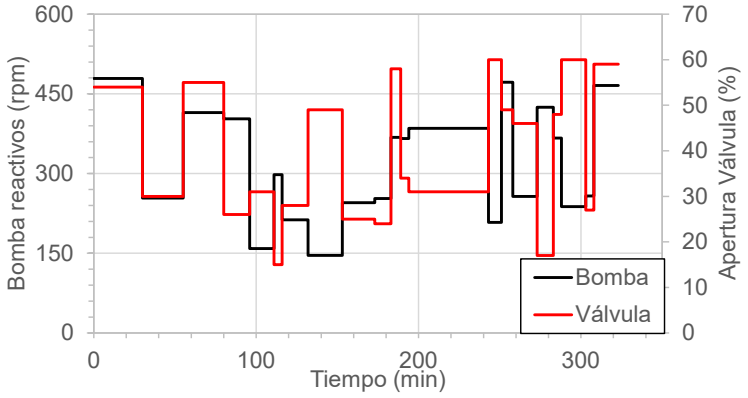

Figura 17: Diseño del experimento de validación.

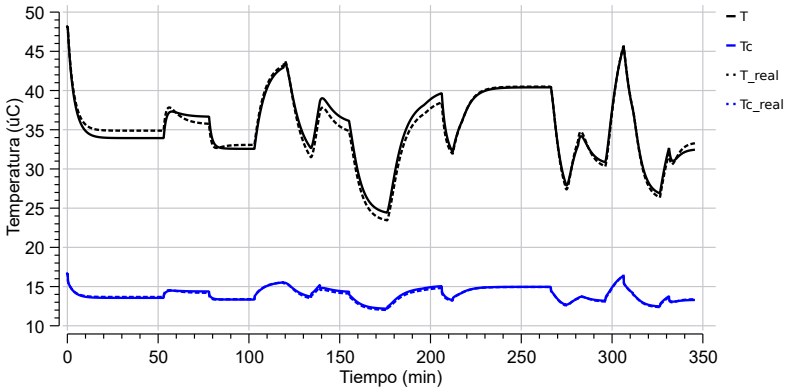

Figura 18: Comparación de temperaturas entre el modelo y el proceso.

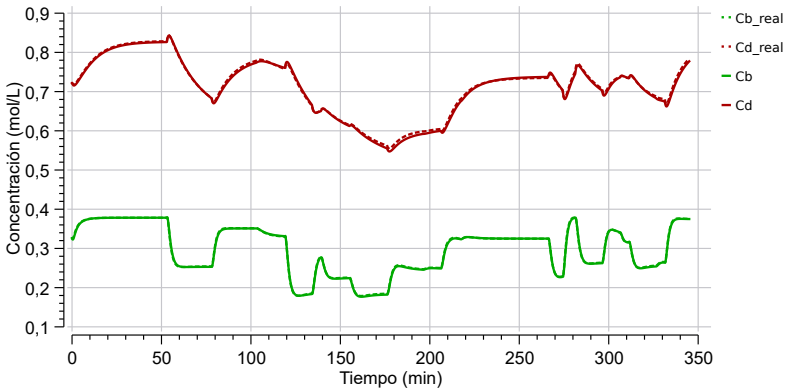

Figura 19: Comparación de concentraciones de B y D entre el modelo y el proceso

Capítulo 4

Formulación del controlador predictivo

El modelo matemático desarrollado en la sección anterior será el corazón del controlador predictivo no lineal (MPC). Además, será necesario formular un estimador de estados que permita conocer el valor de las variables no medidas y dotar de acción integral al controlador. Debido a que pueden haber perturbaciones en los caudales de reactivos y refrigerante, el controlador no calculará las acciones finales de control, es decir, la velocidad de la bomba de reactivos y la apertura de la válvula de reactivos. Se calcularán los set-points de los controladores de dichos caudales. El esquema general de la planta actualizado con el estimador de estados y el controlador se muestra en la figura 20.

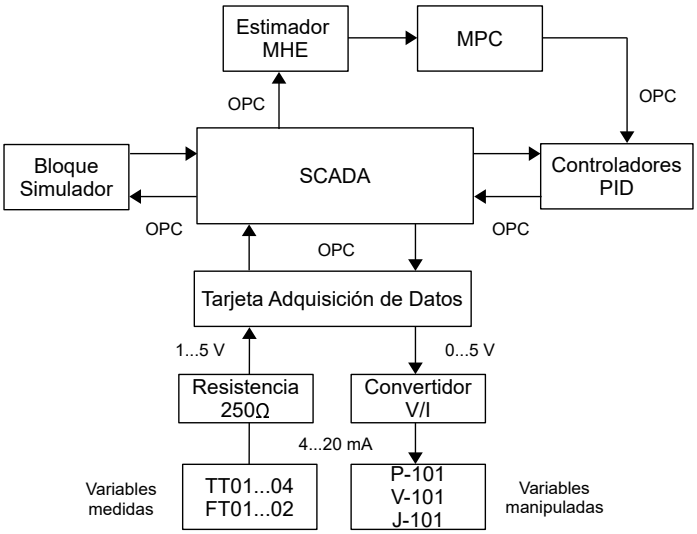

Figura 20: Esquema general de la planta con MHE, MPC y RTO.

Tanto el MPC como el estimador de estados se envolverán en un servidor OPC-UA para facilitar la comunicación con el SCADA de la planta y los controladores PID de los caudales de flujo de reactivos y refrigerante. También se desarrollará una interfaz de comunicación en Python pues el SCADA actual no es compatible con OPC-UA, solo con OPC-DA.

4.1. Controlador predictivo no lineal

Como se indicó en el capítulo 1, un controlador predictivo consiste en realizar predicciones a futuro del proceso para calcular las acciones de control que llevarían las variables controladas a sus referencias respetando las restricciones establecidas. En el capítulo anterior se construyó un modelo que permitirá realizar tales predicciones. El controlador se formula por tanto como aparece en la ecuación (36).

$$
\begin{aligned}
\min_{\substack{u_1(t),\\ u_2(t)}} \quad & \sum_{k=1}^{2}\sum_{i=0}^{N}\gamma_k\left[\frac{y_k(t+i)-w_k(t+i)}{y_{k,upp}-y_{k,low}}\right]^2+\sum_{j=1}^{2}\sum_{i=0}^{Nu}\beta_j\left[\frac{\Delta u_j(t+i)}{u_{j,upp}-u_{j,low}}\right]^2 \\
\text{s.a.} \quad & V\frac{dC_A}{dt}=q(C_{A0}-C_A)-k_{10}e^{-\frac{E_1}{RT}}C_AV-2k_{30}e^{-\frac{E_3}{RT}}C_A^2V+d_1, \\
& V\frac{dC_B}{dt}=-qC_B+k_{10}e^{-\frac{E_1}{RT}}-k_{20}e^{-\frac{E_2}{RT}}C_BV, \\
& V\frac{dC_C}{dt}=-qC_C+k_{20}e^{-\frac{E_2}{RT}}C_BV, \\
& V\frac{dC_D}{dt}=-qC_D+k_{30}e^{-\frac{E_3}{RT}}C_A^2V, \\
& \rho C_pV\frac{dT}{dt}=q\rho C_p(T_0-T)-\alpha q_c^{4/5}(T-T_c)-V\sum_{i=1}^{3}r_i\Delta H_{rxn,i}+d_2, \\
& \rho C_pV_c\frac{dT_c}{dt}=q_c\rho C_p(T_{c0}-T_c)+\alpha q_c^{4/5}(T-T_c)+d_3, \\
& u_k(t)=u_{k,1}+\sum_{i=2}^{Nu}\frac{u_{k,i}-u_{k,i-1}}{1+e^{-\sigma(t-t_i)}}, \\
& y_{k,low}\le y_k\le y_{k,upp} \qquad k=1,2,3, \\
& u_{j,low}\le u_j\le u_{j,upp} \qquad j=1,2
\end{aligned}
\tag{36}
$$

Donde el subíndice $k=1,2,3$ representa la temperatura de reactivos, del serpentín y la concentración de A. Mientras que $j=1,2$ representa el flujo de refrigerantes y el de reactivos. Los parámetros γ_k y β_j son parámetros de peso de cada una de las variables de controladas y manipuladas respectivamente. Las cuales a su vez se normalizan dividiéndolas por su rango $(y_{k,upp}-y_{k,low})$ y $(u_{j,upp}-u_{j,low})$. Los parámetros N y Nu representan los horizontes de predicción y de control. Por último, las variables d_k son términos de perturbaciones. Estas se estimarán al igual que las concentraciones no medidas para dotar al controlador de acción integral. La tabla 3 muestra los parámetros de sintonía base del controlador.

El primer término de la función de coste del controlador representa el seguimiento de la referencia. Solo se establecen consignas para la concentración de B y para la temperatura del reactor, en este caso no hay interés en controlar la temperatura del serpentín. El segundo término corresponde a la penalización de cambios de las variables manipuladas, de forma que se pueda ajustar la agresividad del controlador. Las restricciones son las ecuaciones del modelo, los límites superiores e inferiores de las variables controladas y las manipuladas y la discretización con función sigmoide de los caudales de reactivos y de

Variable	Descripción	Valor
N	Horizonte de predicción	30 min
Nu	Horizonte de control	3
γ_1	Peso de T	100
γ_2	Peso de C_B	100
β_1	Penalización cambios reactivos	0.5
β_1	Penalización cambios refrigerante	0.5

Tabla 3: Parámetros de sintonía del controlador

refrigerante.

Resolviendo el problema de optimización (36), se obtienen las acciones de control necesarias para llevar el proceso a su consigna. Esto se resuelve en EcosimPro® con un enfoque secuencial, usando IDAS como integrador y SNOPT como optimizador. La solución es un vector de seis elementos, considerando un horizonte de control de tres periodos de muestreo, tres para el caudal de reactivos y tres para el de refrigerante. Solo se envía al SCADA el primer elemento de cada uno, ya que se recalculan en cada llamada al controlador. En la figura 21 se muestra gráficamente la secuencia de calculo del MPC.

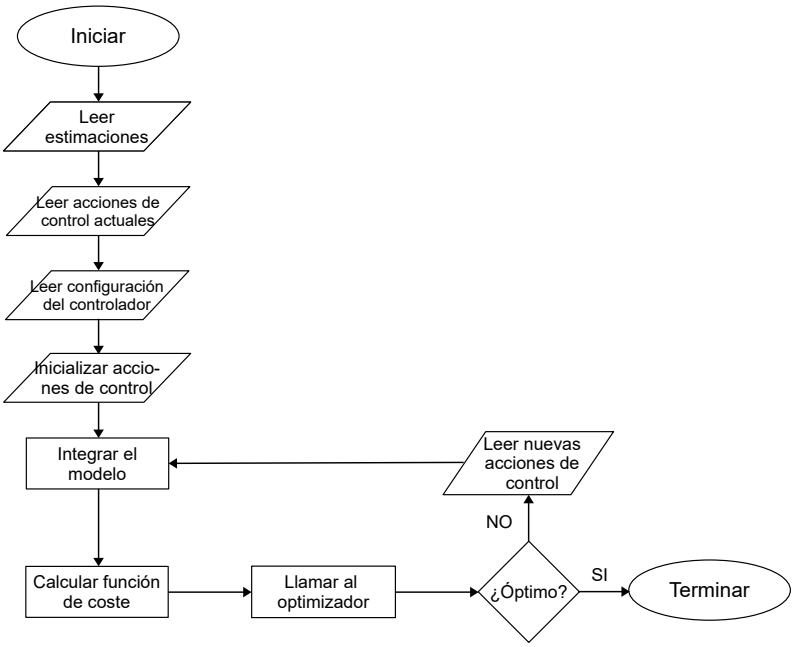

Figura 21: Esquema de funcionamiento del MPC.

4.2. Estimador de estados

La planta cuenta con sensores de flujo y de temperatura pero se estableció que solo uno de concentración, como cabría esperar en una aplicación real. De esta forma, será necesario estimar continuamente las concentraciones de A, C y D. El estimador MHE se formula como se muestra en la ecuación (37). Se agrega un termino de perturbaciones $d_i(t)$ que

permite ajustar las pequeñas discrepancias entre el modelo y el proceso real. Así, el controlador tendrá una acción integral y se reducirá su error en estado estacionario. Este término se agrega a las ecuaciones diferenciales de la temperatura del reactor, del serpentín y de la concentración de B, C_B.

Resolviendo el problema de optimización (37), se obtienen las estimaciones de los valores iniciales de de C_A, C_C, C_B, C_D y de las perturbaciones. Posteriormente pueden obtenerse los estados actuales estimados por integración a partir de estos valores y las acciones de control pasadas. Se usó EcosimPro® con un enfoque secuencial al igual que para el MPC. La ejecución del estimador de estados será previa a la del MPC, pues este último requiere las estimaciones. En la figura 22 se muestra gráficamente la secuencia de cálculo del estimador de estados.

$$
\begin{aligned}
\min_{\substack{y_k(t-N),\\ d_k(t-i)}} \quad & \sum_{k=1}^{3}\sum_{i=0}^{N_E}\left[\gamma_k\left(y_k(t-i)-y_{k,real}(t-i)\right)^2+\alpha_j d_k(t-i)^2\right] \\
\text{s.a.} \quad & V\frac{dC_A}{dt} = q(C_{A0}-C_A) - k_{10}e^{-\frac{E_1}{RT}}C_A V - 2k_{30}e^{-\frac{E_3}{RT}}C_A^2 V + d_1, \\
& V\frac{dC_B}{dt} = -qC_B + k_{10}e^{-\frac{E_1}{RT}} - k_{20}e^{-\frac{E_2}{RT}}C_B V, \\
& V\frac{dC_C}{dt} = -qC_C + k_{20}e^{-\frac{E_2}{RT}}C_B V, \\
& V\frac{dC_D}{dt} = -qC_D + k_{30}e^{-\frac{E_3}{RT}}C_A^2 V, \\
& \rho C_p V\frac{dT}{dt} = q\rho C_p(T_0-T) - \alpha q_c^{4/5}(T-T_c) - V\sum_{i=1}^{3} r_i \Delta H_{rxn,i} + d_2, \\
& \rho C_p V_c\frac{dT_c}{dt} = q_c\rho C_p(T_{c0}-T_c) + \alpha q_c^{4/5}(T-T_c) + d_3, \\
& u_k(t) = u_{k,1} + \sum_{i=2}^{Nu}\frac{u_{k,i}-u_{k,i-1}}{1+e^{-\sigma(t-t_i)}}, \\
& y_{k,low} \le y_k \le y_{k,upp} \qquad k=1,\dots,3, \\
& d_{k,low} \le d_k \le d_{k,upp} \qquad k=1,\dots,3
\end{aligned}
\tag{37}
$$

Donde N_E representa el horizonte de estimación, es decir, el número de periodos de muestreos usados. y_1, y_2, y_3 representan la temperatura de reactivos, del serpentín y la concentración de A. $gamma_k$ son factores de peso de las variables de estado. Y α_j son factores de penalización de las perturbaciones. Como punto de partida, a todas las variables de estado y perturbaciones se les asigna el mismo peso, 1 para las primeras y 0.1 para las segundas. Es necesario proveer al estimador de estados la información de las variables medidas en los N_E periodos de muestreo anteriores.

4.3. Implementación

Tanto el controlador como el estimador de estados se programaron en EcosimPro ®. La idea fundamental es generar una clase a partir del modelo matemático de la planta y la

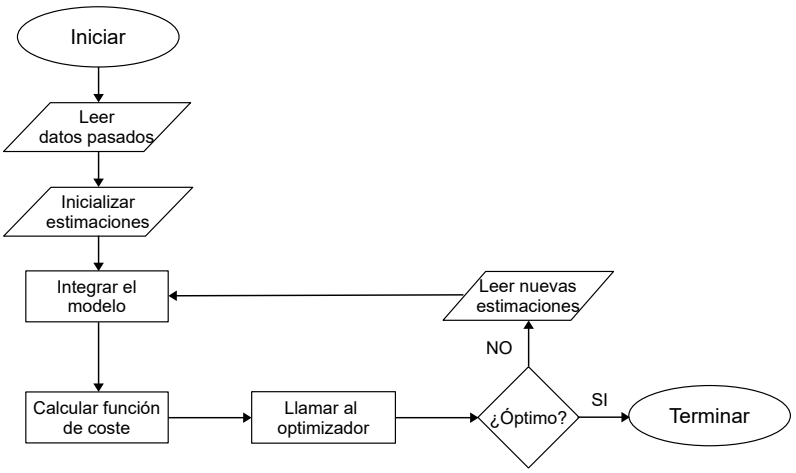

Figura 22: Esquema de funcionamiento del estimador de estados.

función de coste respectiva y, luego, crear un método haga las llamadas al optimizador. Así se facilita la programación y el fichero principal solo contiene la definición de las variables y las llamadas a los métodos de las respectivas clases como sigue:

```
estimador.ejecutar(t_Sam,q,qc,T0,Tc0,T,Tc,Ca,estim)
controlador.control(t_Sam,Pred_h,sta,man_low,man_up,con_low,con_up,MV,u_new)
```

Los parámetros que se pasan a el estimador son el tiempo de muestreo y la información de todas las variables de la planta. Y devuelve el vector `estim` que contiene las estimaciones de C_A, C_C, C_D y de las perturbaciones. Para ejecutar el controlador, es necesario proveerle los límites de las variables controladas y manipuladas, así como también las variables estimadas, entre otras. Las acciones de control calculadas están contenidas en el vector `u_new`. Todo esto se envuelve en una capa OPC-UA. Toda comunicación entre el controlador (de ahora en adelante se refiere tanto al MPC como al MHE) y el SCADA se hará a través de InterfazOPC, detallada en la próxima sección. En la figura 23 se muestra un resumen del procedimiento de calculo seguido en cada llamada al controlador. La variable binaria *Flag* se modifica desde el SCADA y determina si el controlador predictivo está activado o desactivado.

Se probó el comportamiento del controlador sobre una simulación de la planta. Tanto el estimador de estados como el MPC usan los parámetros obtenidos en la sección 3.2 para calcular sus respectivas funciones de coste. La simulación, por otra parte, usa los parámetros originales. Desde la simulación se hacen llamadas al controlador y luego se aplican las acciones de control. Empezando desde una temperatura de 30°C y una concentración de B de 0 mol/L, se activa el controlador para que lleve la planta a 50°C y 0.7 mol/L. Los resultados obtenidos se presentan en la figura 24.

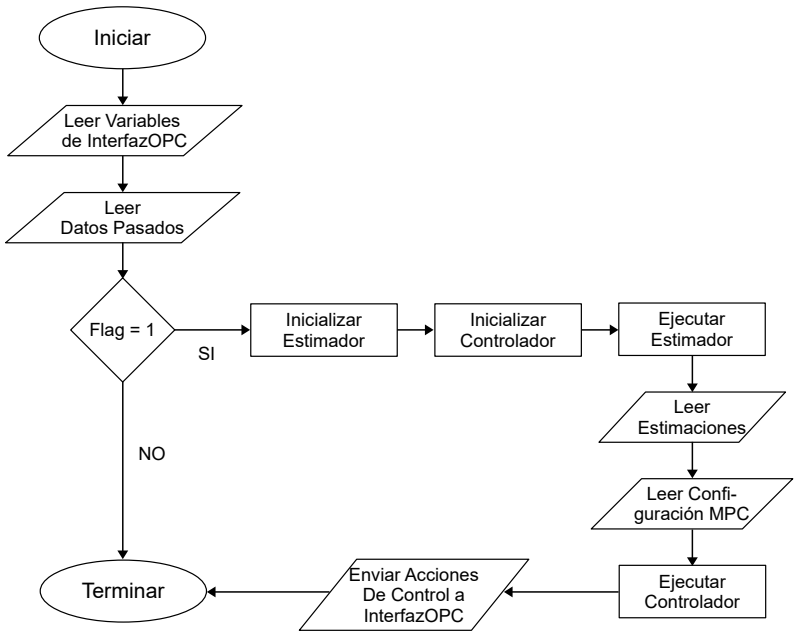

Figura 23: Esquema de funcionamiento del servidor del controlador.

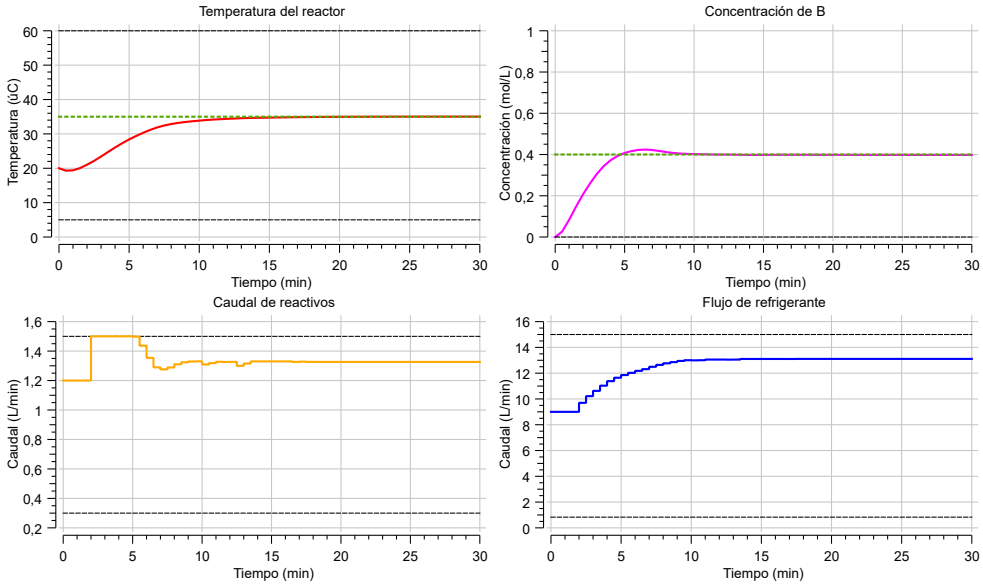

Figura 24: Prueba del controlador predictivo.

Se puede ver que el controlador es capaz de llevar el proceso al set-point requerido, incluso sin error de estado estacionario. Los primeros instantes de tiempo donde no hay cambios de las variables manipuladas corresponden a la recolección de datos para que el MHE pueda empezar la estimación. En la concentración de A se ve un pequeño sobre salto alrededor del set-point pero no es significativo. Por lo demás, las variables manipuladas en ningún momento superan los límites impuestos como es debido.

Para permitir la comunicación entre el controlador y el SCADA, es necesario dotarlo de una capa OPC. EcosimPro ofrece de forma nativa esta posibilidad a través de decks. Estos funcionan como cajas negras que encapsulan el código y todo lo necesario para ejecutarlo de forma independiente de EcosimPro. Para crear un deck se ha de seleccionar en primera instancia su tipo. Se seleccionó como OPC-UA, pues el estándar OPC-DA no es compatible con el compilador requerido por el optimizador SNOPT, Visual Studio 2015 en adelante. Luego, se han de seleccionar las variables de entrada, configuración del controlador y datos medidos de la planta, y las variables de salida, las acciones de control. Por último, solo es necesario darle al botón de generar y se crea un ejecutable que contiene al controlador y cada que es llamado sigue la secuencia de calculo mostrada en la figura 23. La aplicación carece de interfaz gráfica, solo muestra una ventana de la consola donde va reportando información sobre las operaciones que se ejecutan como se muestra en la figura 25.

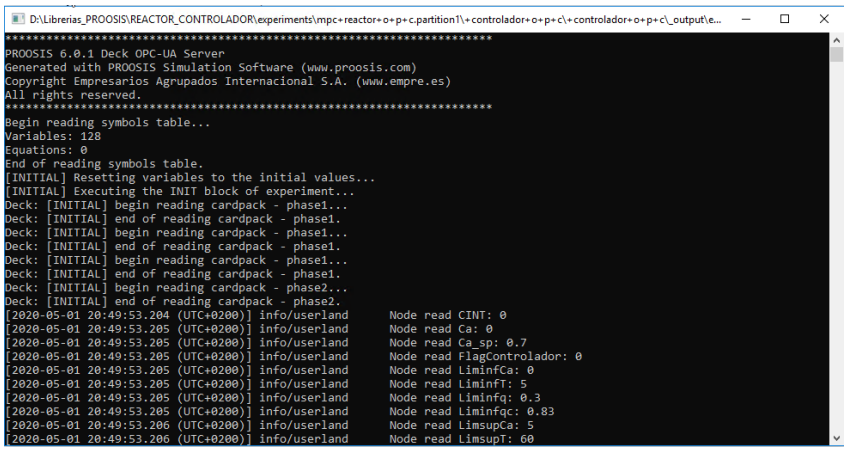

Figura 25: Consola del controlador.

4.4. Interfaz de comunicación

El controlador se envolvió en una capa OPC-UA. No es posible hacerlo con OPC-DA pues aunque es el estándar más usado, no es compatible con compiladores de C++ actuales requeridos por el optimizador. Infortunadamente, el SCADA de la planta está programado en InTouch 2012, versión solo compatible con OPC-DA. Para solventar este problema, es necesario desarrollar un programa intermedio que sea compatible con ambos estándares, permitiendo la comunicación entre el SCADA y el controlador. El controlador deberá recibir las variables medidas desde el SCADA, así como su configuración, para calcular la consigna de los controladores PID de los caudales de reactivos y refrigerante. Esta interfaz se desarolló en el lenguaje de programación Python y se denomina InterfazOPC. Se usaron las librerías de código abierto OpenOPC y FreeOpcUa pues facilitan enormemente la conexión con los servidores OPC. En la figura 26 se muestra un esquema simplificado del funcionamiento de InterfazOPC.

4.5. Interfaz HMI del MPC

Se agregó una pestaña del MPC a la interfaz HMI del SCADA de la planta. Allí es posible activar o desactivar el MPC o los controladores PID. También se pueden cambiar de forma sencilla los parámetros de todos los controladores y las consignas. En la figura 27 se muestra como luce esta interfaz.

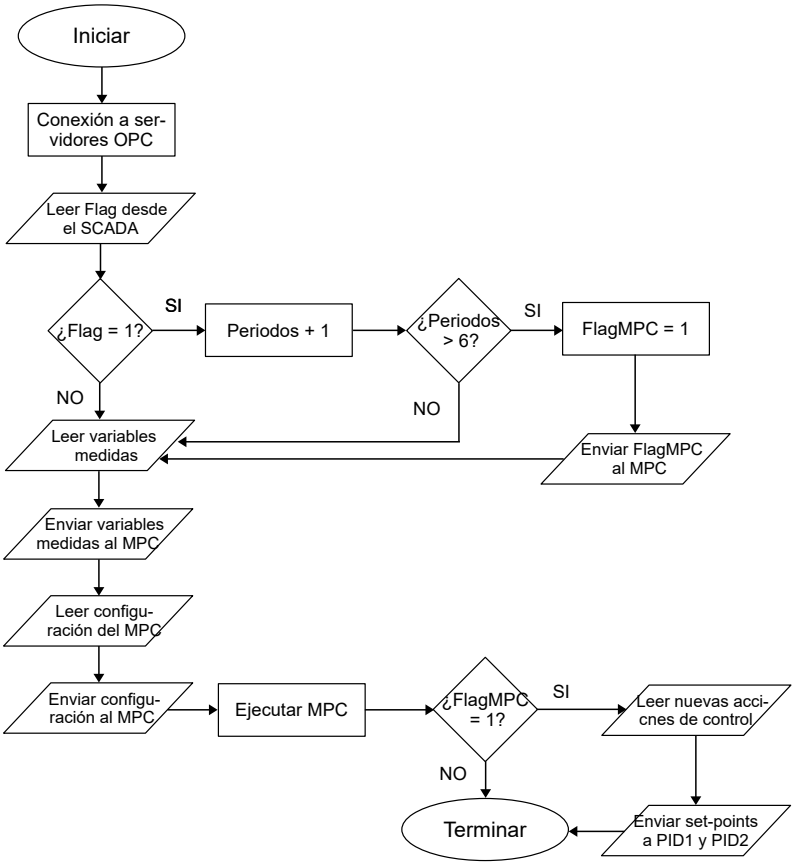

Figura 26: Esquema de InterfazOPC.

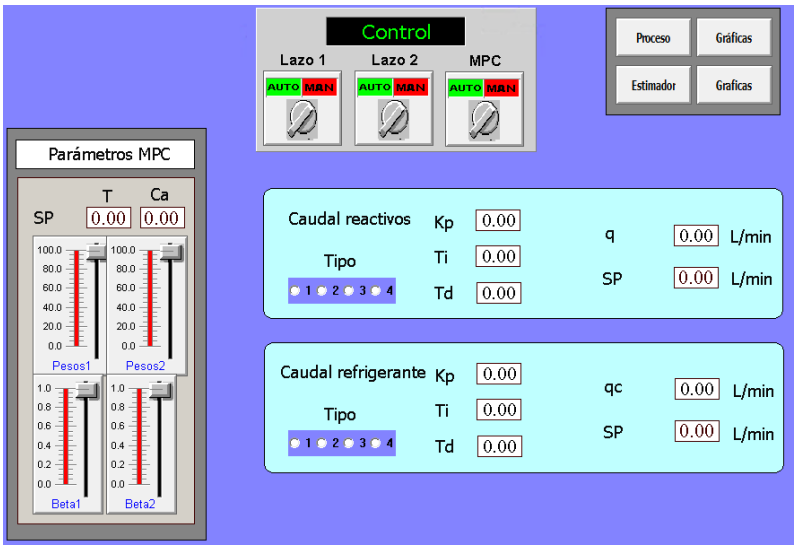

Figura 27: Interfaz del MPC en el SCADA.

Capítulo 5

Optimización en tiempo real

El controlador predictivo no lineal desarrollado en el capítulo anterior permite regular la temperatura y concentración en el reactor modificando las consignas de los controladores PID de flujo de reactivos y refrigerante. Sin embargo, el MPC no tiene consideración de ningún parámetro económico por lo que su funcionamiento se limita estrictamente a la regulación de consignas.

En este capítulo se desarrollará un optimizador en tiempo real para calcular las consignas de temperatura y concentración del MPC que generen el mayor beneficio económico posible. Debido a que las especies de la reacción no son reales sino simuladas, los precios son imaginarios. El funcionamiento de la RTO, sin embargo, servirá como demostración de como funcionaría esta estrategia de optimización en un caso real como una capa superior de la pirámide de automatización.

5.1. Formulación

Se usará el modelo desarrollado en el capítulo 3 para formular el optimizador en tiempo real. Se simplificará a estado estacionario pues los transitorios del reactor son previsiblemente cortos y, por tanto, tienen poco impacto en la economía del proceso. El problema de optimización a resolver consiste en maximizar una función de coste del tipo económico, donde la cantidad consumida/producida de cada especie se multiplica por su respectivo precio. Se añade de la misma forma un término para el refrigerante. Por otra parte, las ecuaciones del modelo se formulan como restricciones de igualdad y se imponen límites al caudal de reactivos, refrigerante, a la temperatura de operación y la concentración. En la ecuación (38) se muestra la formulación del optimizador en tiempo real.

$$
\begin{aligned}
\max_{T,C_A} \quad & J = \sum c_i q C_i - c_A * q * C_{A0} - c_{ref} q_c \\
\text{s.a.} \quad & 0 = q(C_{A0} - C_A) - k_{10} e^{-\frac{E_1}{RT}} C_A V - 2k_{30} e^{-\frac{E_3}{RT}} C_A^2 V, \\
& 0 = -qC_B + k_{10} e^{-\frac{E_1}{RT}} C_A V - k_{20} e^{-\frac{E_2}{RT}} C_B V, \\
& 0 = -qC_C + k_{20} e^{-\frac{E_2}{RT}} C_B V, \\
& 0 = -qC_D + k_{30} e^{-\frac{E_3}{RT}} C_A^2 V,
\end{aligned}
\tag{38}
$$

$$
\begin{aligned}
&0 = q\rho C_p(T_0 - T) - \alpha q_c^{4/5}(T - T_c) - V\sum_{i=1}^{3} r_i \Delta H_{rxn,i},\\
&0 = q_c\rho C_p(T_{c0} - T_c) + \alpha q_c^{4/5}(T - T_c),\\
&10 \leq T, T_c \leq 60,\\
&0 \leq C_i \leq 5 \qquad\qquad i = A, \ldots, D,\\
&0 \leq q \leq 1.5,\\
&0 \leq q_c \leq 15
\end{aligned}
$$

El problema de optimización (38) es del tipo no lineal pues las constantes cinéticas dependen de la temperatura, que a su vez depende de la concentración. Además, hay varios términos en los que se multiplica de forma implícita concentración y temperatura. Para resolver este problema se usó EcosimPro® pues, aunque no es un entorno propiamente de optimización, se usó anteriormente para la estimación de parámetros y de estados y para el controlador predictivo. Además, tiene la gran ventaja de que la simulación se puede envolver fácilmente en un servidor OPC-UA para facilitar las comunicación con el SCADA de la planta. El optimizador usado fue igualmente SNOPT y el código se puede encontrar en el anexo A.10. Aunque los precios de cada especie y del refrigerante se podrán modificar desde el SCADA, la tabla 4 muestra los precios de partida considerados. Es importante resaltar que estos precios son arbitrarios pero reflejan que la sustancia de interés es B.

Variable	Descripción	Valor
c_A	Precio de A	0.5 €/mol
c_B	Precio de B	5 €/mol
c_C	Precio de C	0.3 €/mol
c_D	Precio de D	0.2 €/mol
c_{ref}	Costo del refrigerante	0.01 €/L

Tabla 4: Precios de partida de las especies y el refrigerante

Al igual que para el controlador predictivo, se envolvió el optimizador en tiempo real en un servidor OPC-UA y luego se modificó InterfazOPC para poder establecer los precios de las especies desde el SCADA. En la ventana de control avanzado del SCADA se agregó la opción de establecer precios para las especies y la posibilidad de activar o desactivar la optimización en tiempo real. Cuando este está activado, las consignas de temperatura y concentración del controlador predictivo son modificadas automáticamente y no es posible intervenirlas manualmente. La optimización se resuelve previamente a cada llamada del controlador predictivo.

Capítulo 6

Resultados y discusión

En esta sección se probará el comportamiento del controlador predictivo y el optimizador en tiempo real. Para el primero, se estudiará el comportamiento del sistema ante cambios en las consignas de concentración y temperatura. Se probarán diferentes rangos de operación y el rechazo de perturbaciones. Para el segundo, se realizarán cambios en los precios de las especies y el refrigerante y luego se analizará como se adapta el controlador predictivo para llevar al proceso a las nuevas consignas. Todas las pruebas se hicieron sobre la simulación conectada al SCADA de la planta. Tal simulación usa los parámetros presentados como conocidos en la tabla 2, mientras que el modelo matemático del controlador y el optimizador usa los parámetros estimados.

6.1. Pruebas controlador predictivo

6.1.1. Seguimiento de consignas

Se empieza la prueba llevando el reactor a una temperatura de 40 °C y una concentración de B de 0.4 mol/L. Luego se ejecutan varios cambios en las consignas como se muestra en las figuras 28 y 29. De forma general, las figuras en donde se presente la evolución de las variables manipuladas estarán escaladas para que reflejen los límites superiores e inferiores de dichas variables. En la primera gráfica se muestran las variables controladas mientras que en la segunda las manipuladas. Se presentan la velocidad de la bomba de reactivos y la apertura de la válvula de refrigerante aunque realmente el controlador predictivo calcula consignas de caudales. Los controladores PID son los encargados de calcular la acción final de control.

En el primer cambio se ajusta la consigna de temperatura a 45 °C, a lo que el controlador aumenta la velocidad de la bomba de reactivos y disminuye la apertura de la válvula del refrigerante. Se logra llegar a la consigna en alrededor de 15 minutos, teniendo solo un pequeño sobrepico en la concentración de B. En el siguiente cambio se modifica la consigna de temperatura a 35 °C, a lo que el controlador responde disminuyendo temporalmente el caudal de reactivos y llevando la apertura de refrigerante a prácticamente el máximo (60 %). En este caso, aunque se alcanza rápidamente la consigna de temperatura, la de concentración sufre una perturbación más pronunciada que en el primer ejemplo. En el tercer cambio ejecutado, se modificaron simultáneamente ambas consignas a 40 °C y 0.3 mol/L respectivamente. El controlador disminuye ambos caudales, pues debe aumentar la temperatura y disminuir la concentración de B. Alrededor del minuto 40, se cambió la consigna de concentración a 0.5 mol/L y aunque el controlador lleva el flujo de reactivos al máximo no es posible llegar a la consigna fijada. El controlador decide mantener la

consigna de la temperatura y aumentar al máximo la concentración de B. En el último cambio, se modificó la consigna de temperatura a 50 °C, lo que implica que el controlador disminuye la apertura del refrigerante. Como aún tiene una consigna de concentración que no puede alcanzar, mantiene al máximo el caudal de reactivos.

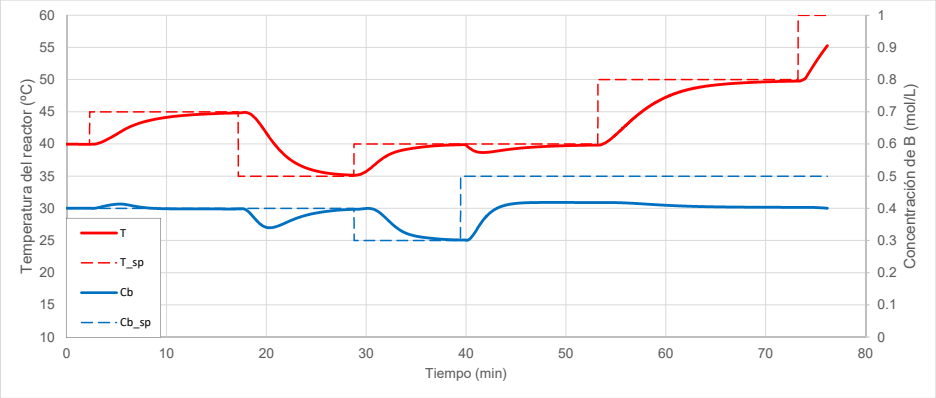

Figura 28: Seguimiento de consignas del MPC.

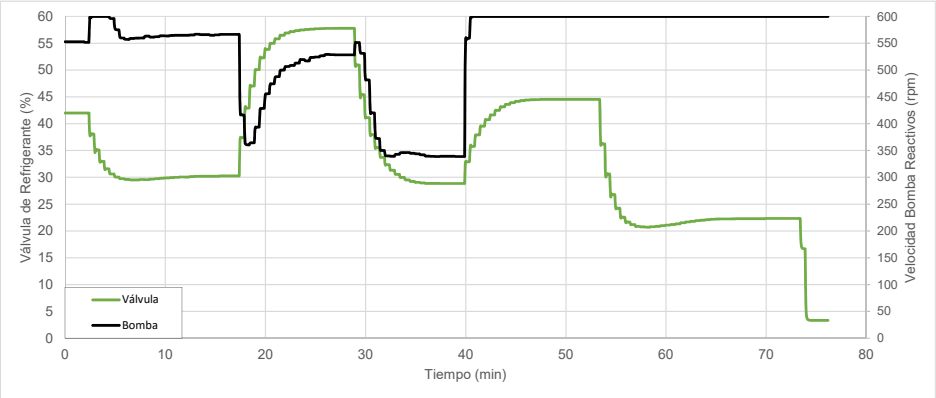

Figura 29: Variables manipuladas del MPC.

6.1.2. Rechazo de perturbaciones

Luego de la prueba de seguimiento de consignas, se ejecutó un experimento para probar como rechaza el controlador tanto perturbaciones medidas como no medidas. Como el controlador está implementado en una simulación de la planta, es fácil modificar intencionalmente los valores de las perturbaciones. Se cuenta con medidas de la temperatura de entrada de reactivos y del refrigerante mientras. Por otra parte, en el modelo se asume que la concentración de A es constante e igual a 5 mol/L y el controlador no tiene acceso a tales medidas, aunque se puede modificar desde el SCADA. Se hicieron pruebas escalón hacia arriba y hacia abajo modificando la temperatura de entrada de reactivos y refrigerante y la concentración de entrada de A como se muestra en las figuras 30, 31 y 32. Las consignas de temperatura y concentración se establecieron en 40 °C y 0.36 mol/L respectivamente y no se modificaron durante la prueba.

En primera instancia, se hace un escalón de +5 °C en la temperatura de entrada de reactivos. Hay un pequeño sobrepico en la temperatura del reactor pero el controlador

rápidamente aumenta el caudal de refrigerante para compensarlo. Lo mismo sucede, pero en sentido, contrario cuando se hace un escalón de -5 °C en la misma variable. La desviación observada de la consigna de temperatura es muy pequeña, no supera 0.5 °C, y la de concentración es inapreciable. Posteriormente, se hace un escalón hacia abajo y luego hacia arriba de 5 °C en la temperatura de entrada de refrigerante. Para el primer caso, el controlador consigue volver a la consigna de temperatura disminuyendo un poco el caudal de refrigerante. La modificación necesaria para mantener tal consigna es mucho más significativa cuando se eleva la temperatura de entrada de refrigerante, aunque luego de una desviación máxima de casi 1 °C es posible volver a la consigna de 40 °C. Hasta este punto, se hicieron modificaciones en dos perturbaciones medidas por lo que se probó la acción feedforward del controlador. Sin embargo, las perturbaciones no medidas también son muy importantes. Para probar este aspecto, se hicieron escalones hacia arriba y hacia abajo en la concentración de entrada del reactivo A. Se observa que ambas variables controladas, la temperatura del reactor y la concentración de B, se alejan de sus consignas. Aunque el controlador intenta manipular el caudal de reactivos y de refrigerante, no es capaz de mantener el proceso en el punto deseado de operación. Esto da a entender que probablemente el estimador de estados de horizonte móvil no esté funcionando correctamente y, así, el controlador carece de acción integral y previsiblemente no puede garantizar que no haya error de estado estacionario en presencia de perturbaciones. Esto se ve reforzado por el hecho de que las variables manipuladas siguen estando dentro de sus límites y, intuitivamente, se puede pensar que si que sería posible llegar a la consigna con una pequeña modificación.

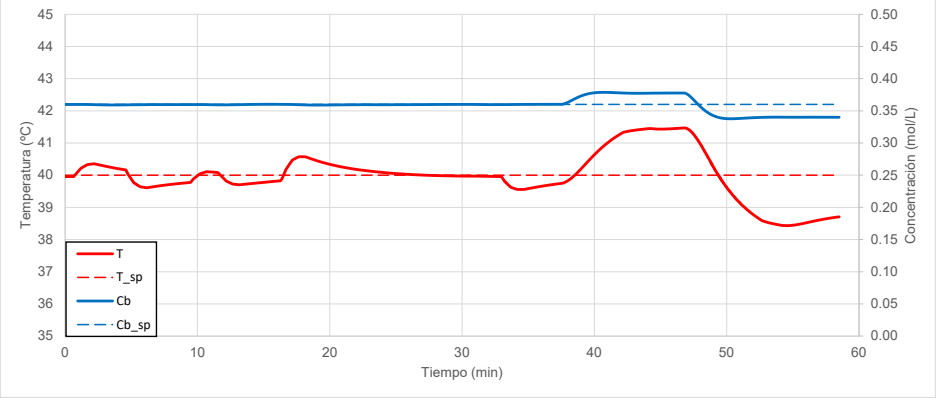

Figura 30: Rechazo de perturbaciones del MPC.

6.2. Pruebas del optimizador en tiempo real

Cuando se activa el optimizador en tiempo real en el SCADA, se desactiva la opción de modificar manualmente las consignas del controlador predictivo. Estas ahora son calculadas por el RTO de forma previa a las llamadas al controlador. Los precios de cada una de las especies y del refrigerante pueden ser modificados desde el SCADA. Es importante recordar, que los precios son arbitrarios pues la reacción química es simulada. En la prueba realizada solo se cambió el costo de refrigerante y el precio del producto de interés, B.

Se inicia la prueba del RTO estableciendo los precios base mostrados en la tabla 4. Se ve que el RTO determina que el máximo beneficio económico se alcanza a una temperatura de 42 °C y una concentración de B de 0.4 mol/L, para lo cual el controlador aumenta al máxima el caudal de reactivos. Luego de unos minutos, el proceso llega a la consigna

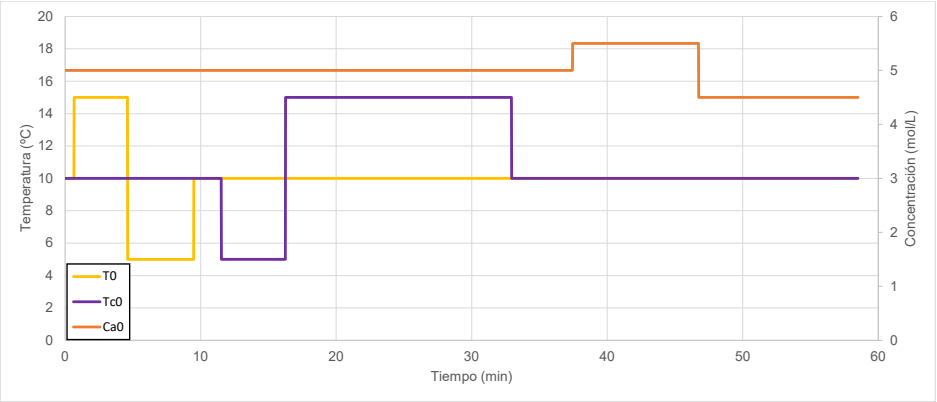

Figura 31: Escalones en las perturbaciones.

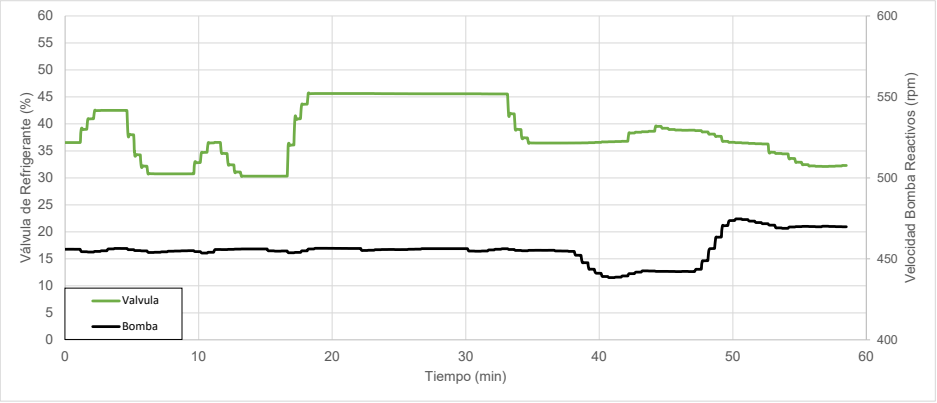

Figura 32: Variables manipuladas del MPC.

establecida por el RTO. Posteriormente, se duplicó el precio de la especie B, manteniendo los demás constantes, y se puede apreciar que aunque se mantiene aproximadamente la misma consigna de concentración, la de temperatura disminuye por debajo de los 35 °C. En este punto se ve que el controlador no es capaz de llevar al proceso al punto que le pide el optimizador en tiempo real, lo cuál probablemente se deba a que se está en una condición de operación fuera de la que se ajustó el modelo del proceso. Manteniendo el precio de B en 10 €/mol y duplicando el costo del refrigerante, el RTO aumenta un poco la consigna de temperatura, que ahora si puede ser alcanzada. Luego, se prueba una situación adversa y es una caída súbita y pronunciada de precio de la sustancia de interés. Al llevar el precio de B a 2 €/mol, se puede ver que el RTO determina en llevar al límite de 60 °C la temperatura del reactor y disminuir en gran medida la de concentración. Para esto el controlador determina que se deben llevar al límite inferior los caudales de reactivos y refrigerante. Los precios son tan desfavorables, que prácticamente sería mejorar para el proceso desde el punto de vista económico. Por último, se eleva poco a poco el precio de B hasta el punto en el que el optimizador determina que rentable aumentar los caudales de reactivos y refrigerante.

Es interesante resaltar que en cualquier situación, el optimizador en tiempo real siempre llevó a alguna variable a su límite. Del minuto 0 al 10, el caudal de reactivos está al máximo, mientras que del 10 al 20 lo está el de refrigerante. Del 20 al 25, de nuevo está en

su máximo el caudal de reactivos. Mientras que cuando cae súbitamente el precio de B de los minutos 25 a 37, ambos caudales son llevados al mínimo y la consigna de temperatura al máximo.

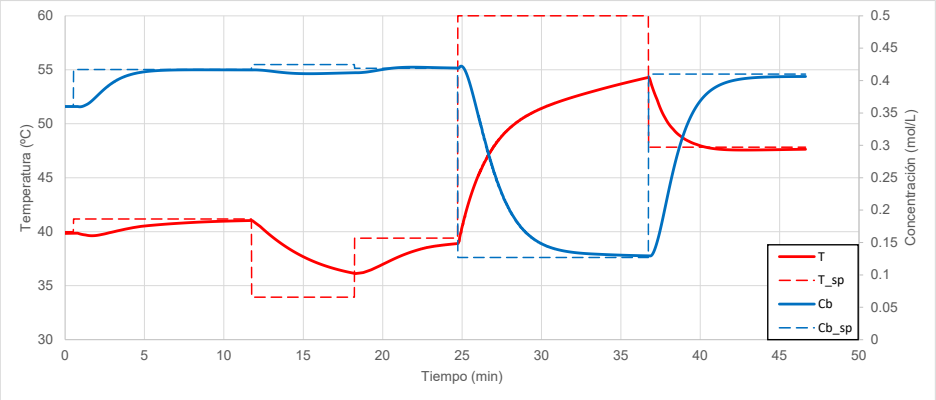

Figura 33: Variables controladas con el RTO+MPC

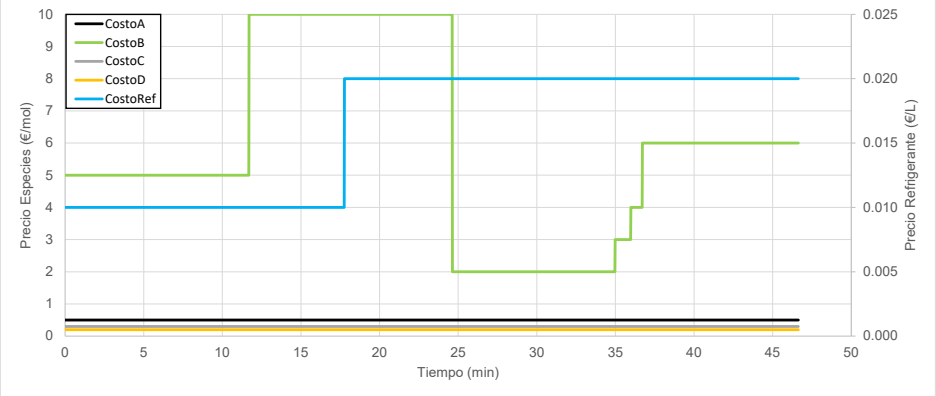

Figura 34: Cambios en los costos de las especies y el refrigerante.

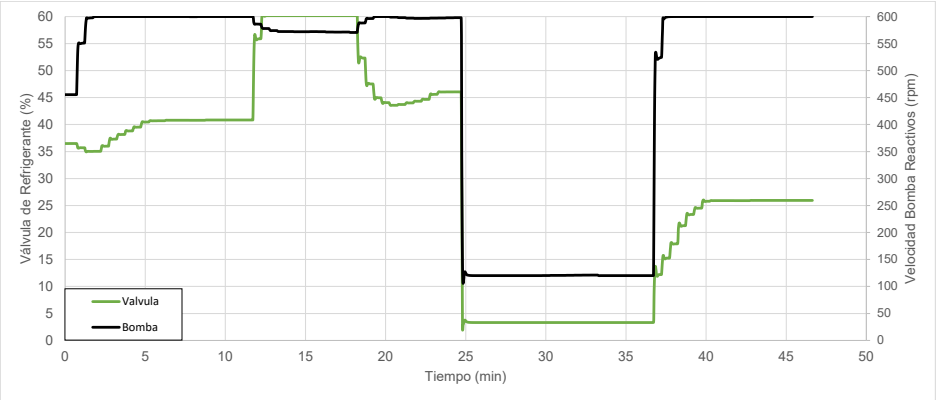

Figura 35: Variables manipuladas del RTO+MPC.

Capítulo 7

Conclusiones

Se desarrolló un modelo matemático de la planta piloto híbrida de un reactor continuamente agitado. Se ajustó dicho modelo a datos del proceso mediante una técnica de estimación de parámetros usando optimización dinámica. Se observó que el ajuste era bueno y luego se corroboró con un conjunto de datos diferentes, a los usados para el ajuste original, que el modelo seguía representando correctamente al proceso.

Usando el modelo matemático como base, se desarrolló un controlador predictivo no lineal para controlar la temperatura del reactor y la concentración de la especie B manipulando las consignas de controladores PID de caudal. Tales controladores se encargan de calcular las acciones finales de control, la velocidad de la bomba de reactivos y la apertura de la válvula de refrigerante. Adicionalmente, se dotó al controlador de un estimador de estados de horizonte móvil para estimar en línea los estados de las variables no medidas y las posibles perturbaciones en cada una de las variables de estado. El controlador fue envuelto en una capa OPC-UA.

Para conectar el controlador con el SCADA de la planta fue necesario programar una interfaz de comunicación en Python que sirve como puente entre ambos. La interfaz funcionó correctamente en cada una de las pruebas realizadas al controlador y no presentó ningún problema.

Por encima del controlador, se implementó una estrategia de optimización en tiempo real que calcula las consignas de temperatura y concentración de B que llevan al máximo beneficio económico, teniendo en cuenta restricciones en las diferentes variables y el precio de cada una de las especies y el refrigerante.

El controlador dio resultados satisfactorios en la prueba de escalones de consignas pero no funcionó igual de bien para rechazar perturbaciones. Aunque si era posible mantener las consignas especificadas al variar la temperatura de entrada de reactivos y de refrigerante, hubo grandes desviaciones cuando se modificó la concentración de entrada de A. Esto se debe probablemente a que el estimador de estados de horizonte móvil no está funcionando adecuadamente y no calcula los términos de perturbaciones como debe. En cualquier caso, no hubo en ningún momento inestabilidad del controlador.

Por otra parte, el optimizador en tiempo real funcionó correctamente. Al modificar los precios de las especies desde el SCADA, se calculaban automáticamente nuevas consignas a las que luego el controlador intentaba llevar al proceso. Se notó que el controlador no era capaz de alcanzar lo impuesto por el optimizador cuando ambos caudales de reactivos y de refrigerante debían ser llevados al mínimo. Esta situación no fue estudiada a la hora de obtener el modelo del proceso, por lo que está fuera de su rango de validez.

Con todo esto, se puede concluir que el desarrollo e implementación de capas superiores de la pirámide de automatización a los procesos no es una tarea sencilla y requiere de conocimientos multidisciplinares. Se deben pasar por múltiples etapas, desde la comprensión de como funciona el proceso, su modelado, la formulación del controlador, etc. hasta la integración de todos los elementos con el SCADA. Aunque en el presente trabajo se estudia la implementación a una simulación de la planta conectada al SCADA, es de preveer que la implementación en el proceso real será más compleja y se requerirá estudiar más a fondo la formulación del controlador y agregar factores como el filtrado de las señales y la calibración de los sensores de la planta. La estimación de estados también deberá ser revisada para que el controlador pueda rechazar efectivamente perturbaciones no medidas.

Bibliografía

[1] Thomas Edgar, David Himmelblau y Ladson Leon. *Optimization of Chemical Processes*. 2da edición. McGraw Hill, 2001.

[2] Fundación Empresarial de la Industría Química Española. *Radiografía del sector químico español 2019*. URL: https://www.feique.org/el-sector-en-cifras/ (visitado 07-05-2020).

[3] Seborg Dale, Thomas Edgar, Duncan Mellichamp y Fancis J. Doyle. *Process Dynamics and Control*. 4ta edición. Wiley, 2016. ISBN: 978-1-119-28595-3.

[4] Thomas Marlin. *Process Control: Designing Processes and Control Systems for Dynamic Performance*. 2da edición. McGraw Hill, 200.

[5] Michael G. Forbes, Rohit S. Patwardhan, Hamza Hamadah y R. Bhushan Gopaluni. «Model Predictive Control in Industry: Challenges and Opportunities». En: *IFAC-PapersOnLine* 48.8 (ene. de 2015), págs. 531-538. ISSN: 2405-8963. DOI: 10.1016/J.IFACOL.2015.09.022. URL: https://www.sciencedirect.com/science/article/pii/S2405896315011039.

[6] J Rawlings, D Q Mayne y Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Ene. de 2017.

[7] R E Kalman. «A New Approach to Linear Filtering and Prediction Problems». En: *Journal of Basic Engineering* 82.1 (mar. de 1960), págs. 35-45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. URL: https://doi.org/10.1115/1.3662552.

[8] Cesar de Prada, Daniel Sarabia, Gloria Gutierrez, Elena Gomez, Sergio Marmol, Mikel Sola, Carlos Pascual y Rafael Gonzalez. «Integration of RTO and MPC in the Hydrogen Network of a Petrol Refinery». En: *Processes* 5.4 (ene. de 2017), pág. 3. ISSN: 2227-9717. DOI: 10.3390/pr5010003. URL: http://www.mdpi.com/2227-9717/5/1/3.

[9] Jesus María Zamarreño Cosme. *Acceso a datos mediante OPC*. 2010. URL: http://www.andavira.com/catalogo/?idlibro=310%7B%5C&%7Dcat=5%7B%5C&%7Dcat2=1%7B%5C&%7Dcat3=2%7B%5C&%7Dlibros=check%7B%5C&%7Dlang=esp.

[10] Wolfgang Mahnke, Stefan-Helmut Leitner y Matthias Damm. *OPC Unified Architecture*. 1st. Springer Publishing Company, Incorporated, 2009. ISBN: 3540688986.

[11] Lorenz T Biegler. *Nonlinear Programming: Concepts, Algorithms and Applications to Chemical Processes*. Society for Industrial y Applied Mathematics, ene. de 2010, pág. 410. ISBN: 978-0-89871-702-0. DOI: doi:10.1137/1.9780898719383. URL: https://doi.org/10.1137/1.9780898719383.

[12] Guido Lorenzoni. *Dynamic Optimization Methods with Applications*. 2009. URL: https://ocw.mit.edu.

[13] Philip E Gill, Walter Murray y Michael A Saunders. «SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization». En: *SIAM Review* 47.1 (ene. de 2005), págs. 99-131. ISSN: 0036-1445. DOI: 10.1137/S0036144504446096. URL: https://doi.org/10.1137/S0036144504446096.

[14] Daniel López, Gloria Rodríguez, Pedro Herrero y César Moraga. «New features in dynamic optimization with EcosimPro(TM)». En: *ITEGAM-JETIA* 4.13 SE -

Articles (mar. de 2018). DOI: 10.5935/2447-0228.201838. URL: https://itegam-jetia.org/journal/index.php/jetia/article/view/38.

[15] Lorenz T Biegler. «Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation». En: *Computers & Chemical Engineering* 8.3 (1984), págs. 243-247. ISSN: 0098-1354. DOI: https://doi.org/10.1016/0098-1354(84)87012-X. URL: http://www.sciencedirect.com/science/article/pii/009813548487012X.

[16] Luis Bergh. «A Hybrid Approach to Empirically Test Process Monitoring, Diagnosis and Control Strategies». En: *Advances in Intelligent and Soft Computing* 126 (ene. de 2012), págs. 215-222. DOI: 10.1007/978-3-642-25908-1_28.

[17] Sergio Sanz. «Diseño de una planta piloto híbrida y control de la misma». Trabajo de Investigación. Universidad de Valladolid, 2015.

[18] Juan Manuel Gordo Martín. «Control de una planta piloto de intercambio de calor». Trabajo de Fin de Grado. Universidad de Valladolid, 2016. URL: https://bit.ly/2wAsfPP.

[19] María Paloma Marcos Núñez. «Implementación de una planta piloto, desarrollo de su sistema de control y estudio de su operación óptima». Trabajo de Fin de Máster. Universidad de Valladolid, 2017. URL: https://bit.ly/38w6EVR.

[20] Marcos Pérez González. «Desarrollo de un servidor OPC para la tarjeta de adquisición de datos externa PMD-1208FS». Trabajo de Fin de Grado. Universidad de Valladolid, 2007.

[21] J G van de Vusse. «Plug-flow type reactor versus tank reactor». En: *Chemical Engineering Science* 19.12 (1964), págs. 994-996. ISSN: 0009-2509. DOI: https://doi.org/10.1016/0009-2509(64)85109-5. URL: http://www.sciencedirect.com/science/article/pii/0009250964851095.

[22] Jesús M. Zamarreño, María J. Fuente y Luis F. Acebes. «Desarrollo de un controlador PID accesible como servidor OPC». En: *XXXVI Jornadas de Automática*. Bilbao: Comité Español de Automática (CEA-IFAC), 2015, págs. 997-1000. URL: https://bit.ly/38LOvDI.

[23] T H Chilton, T B Drew y R H Jebens. «Heat Transfer Coefficients in Agitated Vessels». En: *Industrial & Engineering Chemistry* 36.6 (jun. de 1944), págs. 510-516. ISSN: 0019-7866. DOI: 10.1021/ie50414a006. URL: https://doi.org/10.1021/ie50414a006.

[24] T L Bergman, F P Incropera, D P DeWitt y A S Lavine. *Fundamentals of Heat and Mass Transfer*. 7ma edición. Wiley, 2011, pág. 544. ISBN: 9780470501979. URL: https://books.google.es/books?id=vvyIoXEywMoC.

Apéndice A

Anexos

A.1. Pre-experimento: calentamiento e enfriamiento

Se hizo un pre-experimento para tener una idea de que tan rapido se calienta el reactor y que tan efectivo es su enfriamiento. Esto permitio conocer mejor la planta y aporto informacion util para preparar tanto la simulacion de la planta como la planeacion del experimento para ajustar los parametros del modelo. Se siguieron los siguientes pasos:

- Desconectar el bloque que simula la reaccion quimica. Esto permite operar con una potencia constante las resistencias electricas.
- Llenar el reactor con agua hasta el nivel normal de operacion.
- Encender el agitador y se ajusto su velocidad de giro hasta un valor del 30 %.
- Cerrar la valvula de refrigerante y apagar la bomba de reactivos.
- Encender las resistencias a su maxima potencia, enviando una señal de 3.3 V al amplificador.
- Apagar las resistencias una vez el reactor alcance 60°C.
- Esperar alrededor de quince minutos para ver como eran las perdidas al ambiente.
- Finalmente, abrir la valvula de refrigerante hasta el 100 %.

Los resultados de los experimentos se muestran en las figuras 36 y 37. Se puede ver que la temperatura del reactor solo disminuye dos grados durante los quince minutos en los que no hay calentamiento ni enfriamiento. Se puede concluir que las perdidas al ambiente son despreciables. Ademas, la capacidad de enfriamiento es muy similar a la de calentamiento, lo que es bueno para la regulacion de temperatura. Los datos de este experimento se pueden usar para dar una estimacion inicial del coeficiente de transferencia de calor y conocer la velocidad maxima a la que pueda cambiar la temperatura, siendo esto de importancia para el filtrado de las señales.

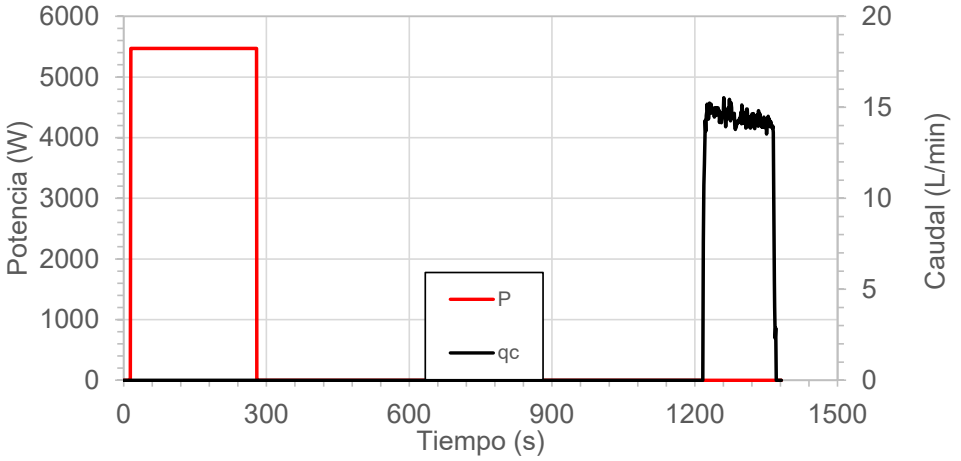

Figura 36: Potencia de las resistencias y caudal de refrigerante.

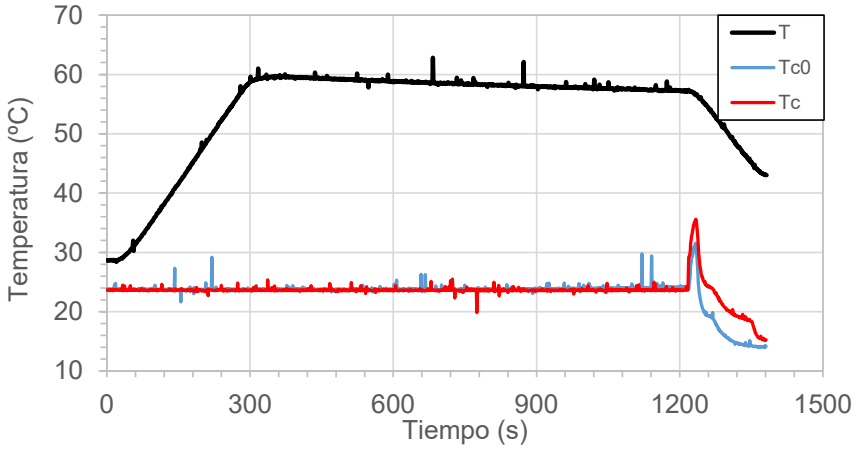

Figura 37: Temperatura del reactor (T), de entrada (T_{c0}) y de salida del serpentin (T_c).

A.2. Conversion de señales

La tarjeta de adquisicion de datos tiene seis variables de entrada y tres de salida. A continuacion se recopilan todas las ecuaciones de conversion, de voltios (x) a unidades de ingenieria (y), utilizadas.

Señales de entrada:

- Temperatura de reactivos (ºC)

$$y = 25.323 \cdot x - 26.728 \tag{39}$$

- Temperatura entrada de refrigerante (ºC)

$$y = 25.323 \cdot x - 26.728 \tag{40}$$

- Temperatura del reactor (ºC)

$$y = 25.323 \cdot x - 26.728 \tag{41}$$

- Temperatura de salida de refrigerante (ºC)

$$y = 25.323 \cdot x - 26.728 \tag{42}$$

- Caudal de reactivos (L/min)

$$y = 0.760 \cdot x - 0.600 \tag{43}$$

- Caudal de refrigerante (L/min)

$$y = 6.046 \cdot x - 5.212 \tag{44}$$

Señales de salida:

- Velocidad de la bomba de reactivos (rpm)

$$y = 120 \cdot x \tag{45}$$

- Apertura de la valvula de refrigerante (%)

$$y = 100 - 20 \cdot x \tag{46}$$

- Potencia del amplificador (W)

$$y = 166.49 \cdot x^5 - 1401.4 \cdot x^4 + 3654.6 \cdot x^3 - 2480.8 \cdot x^2 + 700.02 \cdot x - 122.73 \tag{47}$$

A.3. Bloque simulador

La simulacion de la reaccion se programo en el lenguaje EL de EcosimPro ®.

- Componente

```
COMPONENT Bloque_CalcuLectura

DATA

    REAL V = 11.5           "Volumen del reactor(L)"
    REAL R = 0.00831        "Constante de los gases ideales (kJ/mol/K)"
    REAL cA0 = 5            "Concentracion entrada A (mol/L)"
    REAL Ea1 = 81.09        "Energia de activacion (kJ/mol)"
    REAL Ea2 = 81.09        "Energia de activacion (kJ/mol)"
    REAL Ea3 = 71.13        "Energia de activacion (kJ/mol)"
    REAL k10 = 2.145e10     "Factor preexponencial (1/min)"
    REAL k20 = 2.145e10     "Factor preexponencial (1/min)"
    REAL k30 = 1.507e8      "Factor preexponencial (1/min/mol)"
    REAL dHrxn1 = 4.2       "Calor de reaccion (kJ/mol)"
    REAL dHrxn2 = -11       "Calor de reaccion (kJ/mol)"
    REAL dHrxn3 = -41.85    "Calor de reaccion (kJ/mol)"

DECLS

    REAL Operacion      "Modo de operacion (s/u)"
    -- Mediciones
    REAL q              "Caudal volumetrico en (L/min)"
    REAL T              "Temperatura del reactor (C)"
    REAL T0             "Temperatura de entrada al reactor (C)"
    -- Calculos
    REAL cA, cB, cC, cD     "Concentracion A (mol/L)"
    REAL r1, r2, r3         "Velocidad de la reaccion 1 (mol/L/min)"

    -- Salidas
    REAL x      "Conversion (s/u)"
    REAL P      "Calor de reaccion (W)"

INIT

    cA = 0
    cB = 0
    cC = 0
    cD = 0
    T  = T0

CONTINUOUS
    -- Las reacciones son
    --       A  -1->  B  -2->  C
    --       2A -3->  D
    -- Calculo de la conversion
    x = (cA0 - cA)/cA0
    -- Calculo de la velocidad de reaccion
    r1 = k10*exp(-Ea1/(R*(T+273.15)))*cA*V
    r2 = Operacion*k20*exp(-Ea2/(R*(T+273.15)))*cB*V
    r3 = Operacion*k30*exp(-Ea3/(R*(T+273.15)))*cA**2*V
    -- Balances de materia
    V*cA' = q*(cA0 - cA) + (-r1 - 2*r3)/60
    V*cB' = -q*cB + ( r1 -   r2)/60
    V*cC' = -q*cC + ( r2       )/60
    V*cD' = -q*cD + (        r3)/60
    -- Potencia del amplificador
    P = (1000/60)*V*( - dHrxn1*r1 - dHrxn2*r2 - dHrxn3*r3 )

```

```
58 END COMPONENT
```

- Experimento

```
1  EXPERIMENT OPC_CalcuLectura ON Bloque_CalcuLectura.partition1
2     DECLS
3     OBJECTS
4     INIT
5        -- initial values for state variables
6
7     BOUNDS
8        -- Set equations for boundaries: boundVar = f(TIME;...)
9          -- Valores tomados de https://www.ncbi.nlm.nih.gov/pmc/articles/
       PMC4539502/
10         -- Se han corregido los valores de la energia de activacion para
       que la reaccion pueda tener lugar a 40C
11
12         -- Estos valores sirven como referencia pero deben eliminarse en
       el servidor OPC.
13
14         /*
15       Ea1 = 81.09
16       Ea2 = 81.09
17       Ea3 = 71.13
18       Operacion = 1
19       T = 15
20       T0 = 15
21       cA0 = 5
22       dHrxn1 = 4.2
23       dHrxn2 = -11
24       dHrxn3 = -41.85
25       k10 = 2.145e10
26       k20 = 2.145e10
27       k30 = 1.507e8
28       q = 1
29         */
30
31    BODY
32       -- creates an ASCII file with the results in table format
33       -- REPORT_TABLE("results.rpt", "*",TRUE)
34
35       -- set the debug level (valid range [0,4])
36       DEBUG_LEVEL= 1
37       -- select default integration solver. Valid methods are IDAS (
       _SPARSE), DASSL(_SPARSE), CVODE_BDF(_SPARSE), CVODE_AM, RK4, EULER,
       AM1, AM2 and AM4
38       IMETHOD= IDAS -- default is DASSL, recommended is either IDAS or
       IDAS_SPARSE
39       -- set tolerances and other important inputs
40       REL_ERROR = 1e-06 -- transient solver relative tolerance
41       ABS_ERROR = 1e-06 -- transient solver absolute tolerance
42       TOLERANCE = 1e-06 -- steady solver relative tolerance
43       INIT_INTEG_STEP = -1 -- initial integration step size (-1 means use
        the solver estimation)
44       MAX_INTEG_STEP = -1 -- maximum integration step size (-1 means use
       the solver estimation)
45       NSTEPS = 1 -- Only for explicit solvers use CINT/NSTEPS as
       integration step size
46       REPORT_MODE = IS_EVENT -- by default it reports results at every
       CINT and event detection. Other valid options are IS_STEP, IS_CINT and
        IS_MANUAL_REFRESH
47
48       -- simulate a transient in range[TIME,TSTOP] reporting every CINT
49       TIME = 0
```

```
50        TSTOP = 15
51        CINT = 1
52        INTEG()
53 END EXPERIMENT
```

A.4. Simulacion de la planta

La simulacion de la planta se programo en el lenguaje EL de Ecosimpro ®.

- Componente

```
COMPONENT SimulacionReactor

    DATA

        REAL V = 11.5           "Volumen del reactor (L)"
        REAL Vc = 1             "Volumen de la camisa (L)"
        REAL rho = 1            "Densidad del agua (kg/L)"
        REAL Cp = 4.184         "Capacidad Calorifica del agua (kJ/kg/C)"

        -- Parametros de la reaccion
        REAL UA = 7.182         "Coeficiente global de transferencia de
    calor"

        -- Variables medidas
        REAL T0 = 10            "Temperatura reactivos (C)"
        REAL Tc0 = 10           "Temperatura entrada refrigerante (C)"

        -- Variables manipuladas
        REAL Out_Volts00 = 2.5  "Senal de la valvula (V)"
        REAL Out_Volts01 = 1    "Senal del amplificador (V)"
        REAL Out_Volts02 = 2.5  "Senal de la bomba (V)"

    DECLS

        REAL q              "Caudal reactivos (L/min)"
        REAL qc             "Caudal refrigerante (L/min)"

        -- Salidas medidas del proceso
        REAL T              "Temperatura reactor (C)"
        REAL Tc             "Temperatura refrigerante (C)"

        -- Otras variables del modelo
        REAL Q                  "Calor transferido (kJ/mol)"
        REAL Heat_rxn           "Calor generado (kJ/min)"
        -- Variables de la DAQ
        REAL In_Volts00, In_Volts01, In_Volts02, In_Volts03, In_Volts04,
    In_Volts06     "Senal de 1 a 5 V"

    INIT

        T = T0+10
        Tc = Tc0+10

    CONTINUOUS

    -- Variables de salida
        q = 0.3*Out_Volts02 -- qmax = 1.5
        qc = 5*Out_Volts00  -- qcmax = 15 (La valvula se abre hasta el
    60%
        -- Se convierte el calor de la reaccion de W a kJ/min
        Heat_rxn = 2*(60/1000)*(63.007*Out_Volts01**5 - 537.74*
    Out_Volts01**4 + 1340.7*Out_Volts01**3 - 563.35*Out_Volts01**2 -
    34.826*Out_Volts01 + 12.836)

    -- Balances de energia
        -- Reactor
        rho*Cp*V*T' = (q*rho*Cp*(T0 - T) - Q + Heat_rxn)/60

```

```
54          -- Camisa
55          rho*Cp*Vc*Tc' = (qc*rho*Cp*(Tc0 - Tc) + Q)/60
56
57          -- Transferencia de calor
58          Q = UA*(T - Tc)
59
60      -- Variables medidas
61          In_Volts00 = 1.316*q + 0.789          -- Caudal de reactivos
62          In_Volts01 = 0.165*qc + 0.863         -- Caudal de regriferante
63          In_Volts02 = 0.04*T + 1               -- Temperatura del reactor
64          In_Volts03 = 0.04*Tc0 + 1             -- Temperatura entrada
        refrigerante
65          In_Volts04 = 0.04*Tc + 1              -- Temperatura serpentin
66          In_Volts06 = 0.04*T0 + 1              -- Temperatura reactivos
67
68  END COMPONENT
```

- Experimento

```
1  EXPERIMENT SimulacionVandeVusse ON SimulacionReactor.partition1
2     DECLS
3     OBJECTS
4
5     BOUNDS
6        -- Set equations for boundaries: boundVar = f(TIME;...)
7
8
9     BODY
10       -- creates an ASCII file with the results in table format
11       --REPORT_TABLE("results.csv", "T Tc Ca Cb Cc Cd q qc T0 Tc0")
12       -- set the debug level (valid range [0,4])
13       DEBUG_LEVEL= 1
14       -- select default integration solver. Valid methods are IDAS (
        _SPARSE), DASSL(_SPARSE), CVODE_BDF(_SPARSE), CVODE_AM, RK4, EULER,
        AM1, AM2 and AM4
15       IMETHOD= IDAS -- default is DASSL, recommended is either IDAS or
        IDAS_SPARSE
16       -- set tolerances and other important inputs
17       REL_ERROR = 1e-06 -- transient solver relative tolerance
18       ABS_ERROR = 1e-06 -- transient solver absolute tolerance
19       TOLERANCE = 1e-06 -- steady solver relative tolerance
20       INIT_INTEG_STEP = -1 -- initial integration step size (-1 means use
         the solver estimation)
21       MAX_INTEG_STEP = -1 -- maximum integration step size (-1 means use
        the solver estimation)
22       NSTEPS = 1 -- Only for explicit solvers use CINT/NSTEPS as
        integration step size
23       REPORT_MODE = IS_EVENT -- by default it reports results at every
        CINT and event detection. Other valid options are IS_STEP, IS_CINT and
         IS_MANUAL_REFRESH
24
25       -- simulate a transient in range[TIME,TSTOP] reporting every CINT
26       TIME = 0
27       TSTOP = 120
28       CINT = 1
29       INTEG()
30  END EXPERIMENT
```

A.5. Procesador de ficheros de ReadOPC

Este script se programo en Python. Su funcion es organizar los datos recolectados con ReadOPC en un fichero que se pueda leer facilmente desde Excel ® o EcosimPro ®.

```
# El fichero original debe llamarse Datos_Crudos.txt
import os
import csv

# Lectura del directorio actual
current_directory = script_dir = os.path.dirname(__file__)

# Ruta del archivo de datos
Datos_Crudos = os.path.join(current_directory, "Datos_Crudos.txt")
Datos_Tratados = os.path.join(current_directory, "Datos_Tratados.txt")

# Lectura del fichero de datos
with  open(Datos_Crudos, "r") as f:
    Datos = f.readlines()

# Captura del numero de tags y periodo de muestreo
# Se convierte el SamplingTime de segundos a minutos
NumTags =  int(Datos[0][8:10])
SamplingTime =  float(Datos[0][-2])/60

# Eliminacion de columnas basura
Datos.pop(0)
Datos.pop(1)
Datos.pop(1)
Datos.pop(1)

# Extraccion de datos
Datos_Extraidos = []
for n in  range(1,  len(Datos)):
    a = 0
    b = 20
    Tags = [None]*(NumTags + 1)
    Tags[0] =  str((n-1)*SamplingTime)
    for k in  range(1, NumTags + 1):
        a = b + 1
        b = a + 8
        Tags[k] = Datos[n][a:b].replace(",", ".")
    Datos_Extraidos.append(Tags)

# Escritura del fichero nuevo
with  open(Datos_Tratados, 'w') as f:
    wr = csv.writer(f, delimiter='\t', lineterminator='\n')
    for row in Datos_Extraidos:
        wr.writerow(row)
```

A.6. Ejecutor de experimentos

El script de ejecucion de experimentos se programo en Python. Hace uso de las librerias OpenOPC, time, sys y datetime.

```
import OpenOPC
import time
import sys
from datetime import datetime

# Valores de las variables manipuladas.
# [Velocidad Bomba reactivos, Apertura valvula de refrigerante]
u = [[310, 30], [260, 30], [360, 30], [410, 10], [410, 20], [360, 20],
    [360, 50], [460, 50], [510, 30], [460, 30], [460, 20], [410, 20],
    [410, 50], [310, 50], [310, 40], [260, 60], [160, 60], [160, 40],
    [310, 40], [310, 20], [410, 20], ]

# Tiempo de ejecucion
tiempo = [50, 25, 25, 16, 15, 5, 16, 21, 20,
          10, 6, 5, 49, 8, 7, 15, 10, 5, 15, 5, 15]

# Aceleracion: Use 1 para tiempo real
factor_aceleracion = 1

# Conexion al servidor OPC
opc = OpenOPC.client()
opc.connect('ArchestrA.FSGateway.3')

# Inicializacion del contador de cambios
n = 0

for inter in tiempo:
    now = datetime.now()
    current_time = now.strftime("%H:%M:%S")
    # Escritura de la velocidad de la bomba de reactivos
    opc.write(('Simulacion.Grupo1.Out_Volts02', u[n][0]/120))
    print(" Bomba de reactivos a " +  str(u[n][0]) + ' rpm')

    # Escritura de la apertura de la valvula de refrigerante
    opc.write(('Simulacion.Grupo1.Out_Volts00', 5-u[n][1]/20))
    print("Apertura valvula de refrigerante a " +  str(u[n][1]) + ' %')

    # Agregar el cambio al contador
    n += 1

    # Imprimir en pantalla cuantos cambios se han ejecutado
    print("\n")
    if n == 1:
        print('Se ha realizado ' +  str(n) + ' cambio')
    elif n >= 1:
        print('Se han realizado ' +  str(n) + ' cambios\n')

    # Dormir hasta el proximo cambio
    time.sleep(inter*60/factor_aceleracion)

# Reporte de finalizacion del experimento
now = datetime.now()
current_time = now.strftime("%H:%M:%S")
print(current_time + ': El experimento ha terminado. Exitosamente?')

opc.close()
```

A.7. Codigo estimacion de parametros

El problema de optimizacion dinamica de estimacion de parametros se resolvio en EcosimPro ®. El componente contiene la simulacion dinamica y, por tanto, el calculo de la funcion de coste. El experimento hace llamadas al optimizador externo y devuelve nuevos valores de los parametros a la simulacion.

A.7.1. Componente

```
1  USE MATH
2  COMPONENT SimReactor (INTEGER nsal = 3)
3
4      DATA
5
6          REAL V = 11.5            "Volumen del reactor (L)"
7          REAL Vc = 1              "Volumen de la camisa (L)"
8          REAL R = 0.00831         "Constante de los gases (kJ/mol/K)"
9          REAL rho = 1             "Densidad del agua (kg/L)"
10         REAL Cp = 4.184      "Capacidad Calorifica del agua (kJ/kg/C)"
11       REAL Ca0 = 5           "Concentracion de entrada de A (mol/L)"
12         REAL T0 = 10             "Temperatura de entrada de los reactivos (C)"
13         REAL Tc0 = 10            "Temperatura de entrada de los reactivos (C)"
14
15         -- Parametros a estimar
16         REAL k10 = 3.2           "Factor pre exponencial"
17       REAL k20 = 15.6
18       REAL k30 = 3.2
19         REAL dHrxn1 = -3
20       REAL dHrxn2 = -7
21       REAL dHrxn3 = -12
22         REAL Ea1 = 18.09
23       REAL Ea2 = 18.09
24       REAL Ea3 = 15.21
25         REAL alpha = 1.2197
26
27         -- Optimizador
28         REAL pesos[3] = {1, 1, 1}
29         REAL media[3] = {26.31, 12.33, 2.94}
30         REAL LiminfT = 5
31         REAL LimsupT = 60
32         REAL LiminfTc = 5
33         REAL LimsupTc = 60
34         REAL LiminfCa = 0
35         REAL LimsupCa = 5
36         REAL C = 5
37
38
39     DECLS
40         -- Caudales
41         REAL q                "Caudal reactivos (L/min)"
42         REAL qc               "Caudal refrigerante (L/min)"
43
44         -- Salidas medidas del proceso
45         REAL T                 "Temperatura reactor (C)"
46         --REAL T0
47         REAL Tc                "Temperatura refrigerante (C)"
48         --REAL Tc0
49
50         -- Otras variables del modelo
51         REAL Q                 "Calor transferido (kJ/mol)"
52         REAL Heat_rxn          "Calor generado (kJ/min)"
53         REAL Ca, Cb, Cc, Cd
```

```
54          REAL r1, r2, r3
55          REAL UA
56
57          -- Lectura de datos
58          TABLE_1D tab1, tab2, tab3, tab4, tab5, tab6, tab7, tab8, tab9, tab10
59
60          -- Optimizador
61          REAL F_optim[7]
62          REAL J[nsal]
63          REAL J_costo
64          REAL y_real[nsal], y_modelo[nsal]
65          REAL coef = 0
66
67
68      INIT
69          -- Coeficiente que activa el indice de coste
70        coef = 1
71
72          -- Lectura de variables
73          -- Variables de entrada
74          q = 0.760*linearInterp1D(tab1, TIME) - 0.6
75          qc = 6.046*linearInterp1D(tab2, TIME) - 5.212
76          --T0 = 25.323*linearInterp1D(tab3, TIME) - 26.728
77          --Tc0 = 25.323*linearInterp1D(tab4, TIME) - 26.728
78          -- Variables de salida
79          y_real[1] = 25*linearInterp1D(tab5, TIME) - 25
80          y_real[2] = 25*linearInterp1D(tab6, TIME) - 25
81          y_real[3] = linearInterp1D(tab7, TIME)
82
83          y_modelo[1] = y_real[1]
84          y_modelo[2] = y_real[2]
85          y_modelo[3] = y_real[3]
86
87          T = y_real[1]
88          Tc = y_real[2]
89          Ca = y_real[3]
90
91          Cb = linearInterp1D(tab8, TIME)
92          Cc = linearInterp1D(tab9, TIME)
93          Cd = linearInterp1D(tab10, TIME)
94      CONTINUOUS
95
96          -- Lectura de variables
97          -- Variables de entrada
98          q = 0.760*linearInterp1D(tab1, TIME) - 0.6
99          qc = 6.046*linearInterp1D(tab2, TIME) - 5.212
100         --T0 = 25.323*linearInterp1D(tab3, TIME) - 26.728
101         --Tc0 = 25.323*linearInterp1D(tab4, TIME) - 26.728
102         -- Variables de salida
103         y_real[1] = 25*linearInterp1D(tab5, TIME) - 25
104         y_real[2] = 25*linearInterp1D(tab6, TIME) - 25
105         y_real[3] = linearInterp1D(tab7, TIME)
106
107         -- Balances de materia
108         V*Ca' = q*(Ca0 - Ca) + (-r1 - 2*r3)
109         V*Cb' = -q*Cb + ( r1 -   r2)
110         V*Cc' = -q*Cc + ( r2       )
111         V*Cd' = -q*Cd + (        r3)
112         r1 = k10*exp(-Ea1/(R*(T+273.15)))*Ca*V
113         r2 = k20*exp(-Ea2/(R*(T+273.15)))*Cb*V
114         r3 = k30*exp(-Ea3/(R*(T+273.15)))*Ca**2*V
115
116         -- Balances de energia
```

```
117         -- Reactor
118         rho*Cp*V*T' = q*rho*Cp*(T0 - T) - Q + Heat_rxn
119         -- Camisa
120         rho*Cp*Vc*Tc' = qc*rho*Cp*(Tc0 - Tc) + Q
121         -- Transferencia de calor
122         Q = UA*(T - Tc)
123         UA = alpha*qc**0.8
124         -- Calor generado por la reaccion
125         Heat_rxn = V*( - dHrxn1*r1 - dHrxn2*r2 - dHrxn3*r3 )
126
127         -- Optimizador
128         -- Subtotales del indice de coste
129         EXPAND(i IN 1,3) J[i] = (abs(y_modelo[i] - y_real[i])/(media[i]*C) -
        log(1+abs(y_modelo[i] - y_real[i])/(media[i]*C)))
130         -- Funcion de coste
131       J_costo' = C**2*SUM(i IN 1,3; pesos[i]*J[i])
132
133         -- Restricciones no lineales (expresiones  c(x) <= 0)
134         F_optim[1] = J_costo
135         F_optim[2] = MATH.min(T,LiminfT) - LiminfT
136        F_optim[3] = MATH.max(T,LimsupT) - LimsupT
137        F_optim[4] = MATH.min(Tc,LiminfTc) - LiminfTc
138        F_optim[5] = MATH.max(Tc,LimsupTc) - LimsupTc
139        F_optim[6] = MATH.min(Ca,LiminfCa) - LiminfCa
140        F_optim[7] = MATH.max(Ca,LimsupCa) - LimsupCa
141
142         -- Asignacion de valores del modelo
143         y_modelo[1] = T
144         y_modelo[2] = Tc
145         y_modelo[3] = Ca
146
147 END COMPONENT
```

A.7.2. Experimento

```
1  USE OPTIM_METHODS
2
3  EIDAS calculoSens
4
5  CONST INTEGER numC = 6              -- numero de restricciones del problema de
       optimizacion
6  CONST INTEGER numU = 10             -- numero de variables de decision
7  --CONST INTEGER numX = 4                -- numero de variables de estado + 1 (
       costo)
8
9  FUNCTION INTEGER coste_y_restricciones (IN REAL esnopt_x[], IN INTEGER needF,
       OUT REAL esnopt_F[], IN INTEGER explicit_derivatives, IN INTEGER needG, OUT
        REAL esnopt_G[])
10
11     DECLS
12         REAL arr_quad[ numC + 1 ]                        -- array de integrales
       temporales
13         REAL arr_quadsen[ (numC + 1) * numU ]       -- array de sensibilidad de
        las integrales
14         REAL arr_quadsen_p[ (numC + 1) * numU ] -- array de la derivada
       temporal de la sensibilidad de las integrales
15 --      REAL arr_sen[ numX * numU ]                 -- array de sensibilidad de
        los estados
16 --      REAL arr_sen_p[ numX * numU ]                    -- array de la derivada
        temporal de la sensibilidad de los estados
17
18     BODY
19         FOR( j IN 1,numC + 1 )
```

```
20              arr_quad[ j ] = 0
21          END FOR
22          FOR( j IN 1, (numC + 1) * numU )
23              arr_quadsen[ j ] = 0
24              arr_quadsen_p[ j ] = 0
25          END FOR
26  --      FOR( j IN 1, numX * numU )
27  --          arr_sen[ j ] = 0
28  --          arr_sen_p[ j ] = 0
29  --      END FOR
30
31          -- Actualizar las variables de decision a los valores que propone el
        optimizador
32          calculoSens.setU( "k10", esnopt_x[1] )
33          calculoSens.setU( "k20", esnopt_x[2] )
34          calculoSens.setU( "k30", esnopt_x[3] )
35          calculoSens.setU( "dHrxn1", esnopt_x[4] )
36          calculoSens.setU( "dHrxn2", esnopt_x[5] )
37          calculoSens.setU( "dHrxn3", esnopt_x[6] )
38          calculoSens.setU( "Ea1", esnopt_x[7] )
39          calculoSens.setU( "Ea2", esnopt_x[8] )
40          calculoSens.setU( "Ea3", esnopt_x[9] )
41          calculoSens.setU( "alpha", esnopt_x[10] )
42          SET_INIT_ACTIVE(TRUE)
43
44          -- Reiniciar el calculo de sensibilidades con los nuevos valores de los
         parametros
45
46          calculoSens.reiniciar()
47          calculoSens.calculateSens( TSTOP )                  -- calculoSens.
        getSens( TSTOP, arr_sen, arr_sen_p )
48          calculoSens.getQuad( arr_quad, arr_quadsen )        -- calculo de las
        cuadraturas y sus sensibilidades
49          calculoSens.getQuadN( arr_quadsen_p, 1 )            -- calculo de la
        derivada de las sensibilidades de las cuadraturas
50
51          --  Introduccion de los gradientes de la funcion de costo y
        restricciones a SNOPT
52          --  si la funcion es de camino, hay que introducir la derivada parcial
        de la cuadratura ( arr_quadsen )
53          -- en caso contrario, si la restriccion es de punto final, al usar
        arr_quadsen_p se usa la derivada parcial del valor de la funcion.
54          IF ( explicit_derivatives == 1) THEN
55              FOR( i IN 1,numU )
56                  esnopt_G[i] = arr_quadsen_p[ i ]        -- derivadas de las 7
        F_optim respecto a las 4 variables de decision
57              END FOR
58
59              FOR( i IN numU+1,(numC+1)*numU )
60                  esnopt_G[i] = arr_quadsen[ i ]
61              END FOR
62
63          END IF
64
65          -- Introduccion de los valores de las funcines a SNOPT
66          --  si la funcion es de camino, hay que introducir la cuadratura (
        arr_quad )
67          -- en caso contrario, si la restriccion es de punto final, al usar
        F_optim se usa el valor de la funcion.
68          esnopt_F[1] = F_optim[1]
69          FOR( i IN 2,numC+1 )
70              esnopt_F[i] = arr_quad[ i ]
71          END FOR
```

```
72
73          RETURN 0
74  END FUNCTION
75
76  -- llamada a la funcion de residuos por parte de IDAS. No se debe modificar
77  -- IDAS propone unos valores de las variables y de las derivadas y pide
        calcular los residuos
78  FUNCTION NO_TYPE funcionResiduos( IN REAL tiempo, IN REAL var[], IN REAL der[],
         IN INTEGER numPar, IN STRING nomPar[], IN REAL par[], OUT REAL res[] )
79      DECLS
80      OBJECTS
81      BODY
82          FOR( i IN 1,numPar )
83              setValueReal( nomPar[i], par[i] )
84          END FOR
85
86          FRES_ARGS( tiempo, var, der, res, FALSE )
87
88  END FUNCTION
89
90  -- llamada a las funciones de cuadraturas por parte de IDAS, para el calculo de
         sensibilidades
91  -- utilizar el vector F_optim que se actualiza al llamar a los residuos
92  FUNCTION NO_TYPE funcionQuadraturas( IN REAL tiempo, IN REAL var[], IN REAL der
        [], IN INTEGER numPar, IN STRING nomPar[], IN REAL par[], OUT REAL quad[] )
93      DECLS
94      OBJECTS
95      BODY
96          FOR( i IN 1,numPar )
97              setValueReal( nomPar[i], par[i] )
98          END FOR
99
100         FRES_ARGS( tiempo, var, der, der, FALSE )
101
102         -- funciones de cuadraturas
103         FOR( i IN 1,7 )
104             quad[i] = F_optim[i]
105         END FOR
106
107 END FUNCTION
108
109 FUNC_PTR<ptrFunRes> ptrRes = funcionResiduos
110 FUNC_PTR<ptrFunRes> ptrQuad = funcionQuadraturas
111
112 EXPERIMENT Estim_vandeVusse ON SimReactor.partition1
113
114    DECLS
115
116         REAL dec_var[numU]          -- valor inicial de las variables de
       decision, size numU. Necesario para las llamadas a ESnopt
117
118         REAL xlow[numU]             -- valor inferior de las variables de
       decision, size numU. Necesario para las llamadas a ESnopt
119     REAL xupp[numU]             -- valor superior de las variables de decision,
         size numU. Necesario para las llamadas a ESnopt
120
121     REAL Flow[numC + 1]         -- valor inferior de la funcion objetivo y las
       restricciones, size (numC + funcion de coste). Necesario para las llamadas
       a ESnopt
122     REAL Fupp[numC + 1]         -- valor superior de la funcion objetivo y las
       restricciones, size (numC + funcion de coste). Necesario para las llamadas
       a ESnopt
123
```

```
124         INTEGER calcularSens = 1
125         INTEGER infoESnopt = 0
126
127     OBJECTS
128
129         VECTOR_STRING nombresX      -- variable auxiliar para la inicializacion
        de las sensibilidades
130 --      VECTOR_STRING nombresU      -- variable auxiliar para la inicializacion
        de las sensibilidades
131
132     INIT
133       -- initial values for state variables
134         k10 = 3.2
135       k20 = 15.6
136       k30 = 3.2
137         dHrxn1 = -3
138       dHrxn2 = -7
139       dHrxn3 = -12
140         Ea1 = 18.09
141       Ea2 = 18.09
142       Ea3 = 15.21
143         alpha = 1.2197
144
145     BOUNDS
146       -- Set equations for boundaries: boundVar = f(TIME;...)
147       coef = 1
148
149     BODY
150
151         readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,2,tab1)   -- q
152       readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,3,tab2)   -- qc
153         readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,7,tab3)   -- T0
154         readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,5,tab4)   -- Tc0
155         readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,4,tab5)   -- T
156         readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,6,tab6)   -- Tc
157         readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,11,tab7)  -- Ca
158         readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,12,tab8)  -- Cb
159         readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,13,tab9)  -- Cc
160         readTableCols1D ("03-28-2020_2_Datos_Tratados.txt",1,14,tab10) -- Cd
161
162         EXEC_INIT()
163       -- creates an ASCII file with the results in table format
164       -- REPORT_TABLE("results.rpt", "*",TRUE)
165
166       -- set the debug level (valid range [0,4])
167       DEBUG_LEVEL = 1
168       -- select default integration solver. Valid methods are IDAS (_SPARSE),
        DASSL(_SPARSE), CVODE_BDF(_SPARSE), CVODE_AM, RK4, EULER, AM1, AM2 and AM4
169       IMETHOD = IDAS_SPARSE -- default is DASSL, recommended is either IDAS or
        IDAS_SPARSE
170       -- Settings for different actions. Valid actions are: SEV_DISABLE,
        SEV_NONE, SEV_WARNING, SEV_ERROR, SEV_KILLPOINT, SEV_FATAL
171       eSetErrorAction(ERR_BAD_OPER,SEV_KILLPOINT) -- Detect bad numerical
        operations (eg division by zero), default is SEV_NONE, recommended is
        SEV_KILLPOINT
172       eSetErrorAction(ERR_NAN_INF, SEV_KILLPOINT) -- Detect NaN or Inf values,
        default is SEV_NONE, recommended is SEV_KILLPOINT
173       eSetConfig(CFG_FORCE_STOP_CINT,TRUE) -- Force or not to stop the solver
        each CINT. Sometimes you can speed up the simulation if you select FALSE
174       -- set tolerances and other important inputs
175       REL_ERROR = 1e-06 -- transient solver relative tolerance
176       ABS_ERROR = 1e-06 -- transient solver absolute tolerance
177       TOLERANCE = 1e-06 -- steady solver relative tolerance
```

```
178       INIT_INTEG_STEP = -1 -- initial integration step size (-1 means use the
      solver estimation)
179       MAX_INTEG_STEP = -1 -- maximum integration step size (-1 means use the
      solver estimation)
180       NSTEPS = 1 -- Only for explicit solvers use CINT/NSTEPS as integration
      step size
181
182          -- Anadir las variables del jacobiano para el calculo de sensibilidades
183          -- NO MODIFICAR, LO REALIZA AUTOMATICAMENTE
184          -- SINTAXIS:
185          -- addX( nombre de la variable, valor inicial de la misma, booleano que
       indica si la variable es T_DYNAMIC )
186          nombresX = getVariablesWithCategory( T_JACOBIAN_VARS )
187          FOR( i IN 1,nombresX.size() )
188              calculoSens.addX( nombresX.at(i), getValueReal( nombresX.at(i) ), (
       getVarCategory (nombresX.at(i)) == 2 ) )
189          END FOR
190
191          -- Iniciliazacion del algoritmos de optimizacion
192          -- inicializaciom de las variables de decision, y los limites
193          -- SINTAXIS:
194          -- addU( nombre de la variable, valor inicial de la misma, booleano que
       indica si el parametro es un valor inicial )
195          calculoSens.addU( "k10", k10, FALSE )
196          calculoSens.addU( "k20", k20, FALSE )
197          calculoSens.addU( "k30", k30, FALSE )
198          calculoSens.addU( "dHrxn1", dHrxn1, FALSE )
199          calculoSens.addU( "dHrxn2", dHrxn2, FALSE )
200          calculoSens.addU( "dHrxn3", dHrxn3, FALSE )
201          calculoSens.addU( "Ea1", Ea1, FALSE )
202          calculoSens.addU( "Ea2", Ea2, FALSE )
203          calculoSens.addU( "Ea3", Ea3, FALSE )
204          calculoSens.addU( "alpha", alpha, FALSE )
205
206          dec_var[1] = k10
207          dec_var[2] = k20
208          dec_var[3] = k30
209          dec_var[4] = dHrxn1
210          dec_var[5] = dHrxn2
211          dec_var[6] = dHrxn3
212          dec_var[7] = Ea1
213          dec_var[8] = Ea2
214          dec_var[9] = Ea3
215          dec_var[10] = alpha
216
217        --  Limites de las variables de decision
218              --  k10
219          xlow[1] = 0.001
220          xupp[1] = 20
221              -- k20
222          xlow[2] = 0.001
223          xupp[2] = 20
224          -- k30
225            xlow[3] = 0.001
226            xupp[3] = 20
227          -- dHrx1
228            xlow[4] = -20
229            xupp[4] = -0.001
230          -- dHrxn2
231            xlow[5] = -20
232            xupp[5] = -0.001
233              --  dHrxn3
234          xlow[6] = -20
```

```
235         xupp[6] = -0.001
236             --  Ea1
237         xlow[7] = 0.001
238         xupp[7] = 20
239         -- Ea2
240           xlow[8] = 0.001
241           xupp[8] = 20
242         -- Ea3
243           xlow[9] = 0.001
244           xupp[9] = 20
245         -- alpha
246           xlow[10] = 0.001
247           xupp[10] = 10
248         TIME = 0
249       TSTOP = 345
250       CINT = 0.0833
251
252         -- Inicializacion del algoritmo de integracion y calculo de
    sensibilidades
253         -- Inicializacion del algoritmos de optimizacion
254         -- SINTAXIS:
255         -- iniciarRes( tiempo inicial desde el que calcular las sensibilidades,
     puntero a la funcion de residuos definida previamente )
256         -- iniciarQuad( numero de cuadraturas a calcular, puntero al calculo de
     las funciones de cuadratura )
257         calculoSens.iniciarRes(TIME, ptrRes)
258         calculoSens.iniciarQuad(numC + 1, ptrQuad)
259
260         -- Configurar la tolerancia en el calculo de sensibilidades (opcional)
261         calculoSens.setTol( 1e-5 )
262
263         -- inicializacion de los limites de las restricciones y la funcion de
    coste
264         Flow[1] = -1.0e10
265         Fupp[1] =  1.0e10
266         FOR (i IN 2,7)
267         Flow[i] = 0
268         Fupp[i] = 0
269         END FOR
270
271         -- Representacion en el monitor de los valores previos a la
    optimizacion
272         WRITE ("\n\n\t\t------------------- limits before optimization \n")
273         FOR ( i IN 1,numU )
274                 WRITE ("\t\t var_dec[%d]:\t %g \t %g \t %g \n", i, xlow[i],
    dec_var[i], xupp[i])
275         END FOR
276         FOR ( i IN 1,numC + 1 )
277                 WRITE ("\t\t F_optim[%d]:\t %g \t %g \t %g \n", i, Flow[i],
    F_optim[i], Fupp[i])
278         END FOR
279         WRITE ("\n\t\t------------------- END of limits  \n\n")
280
281     --Optimization extern routine call
282         setSilentMode(TRUE)
283         SET_REPORT_ACTIVE("#MONITOR",FALSE)
284
285         esnopt_init (numU, numC)
286         esnopt_set_variables_bounds_and_initial_values (xlow, xupp, dec_var)
287         esnopt_set_constraints_bounds_and_initial_values (Flow, Fupp, F_optim)
288         esnopt_set_cost_function_and_constraints (coste_y_restricciones)
289         esnopt_set_explicit_derivatives ( calcularSens )
290         esnopt_set_function_precision ( 1.0e-5 )
```

```
291         esnopt_set_iterations_limit (200)
292         infoESnopt = esnopt ()
293 
294         -- Final de la optimizacion, obtencion de los resultados para la
    simulacion.
295         setSilentMode(FALSE)
296         SET_REPORT_ACTIVE("#MONITOR",TRUE)
297 
298         esnopt_print_data ()
299         esnopt_get_variables_values(dec_var)
300         esnopt_free ()
301         RESET()
302         --  SET_INIT_ACTIVE(TRUE)
303 
304         -- Modificacion de los parametros con los nuevos valores obtenidos

305         k10    = dec_var[1]
306         k20  = dec_var[2]
307         k30  = dec_var[3]
308         dHrxn1 = dec_var[4]
309         dHrxn2 = dec_var[5]
310         dHrxn3 = dec_var[6]
311         Ea1      = dec_var[7]
312         Ea2  = dec_var[8]
313         Ea3  = dec_var[9]
314         alpha  = dec_var[10]
315         -- Llamada al integrador
316       TIME = 0
317         CINT = 0.0833
318        INTEG()
319 
320         -- Representacion en el monitor de los valores posteriores a la
    optimizacion
321         WRITE ("\n\n\t\t------------------- The optimiser succeed, returned the
     value: %d. \n", infoESnopt)
322         WRITE ("\n\n\t\t------------------- constraints and decision variables
    after optimization \n")
323         FOR ( i IN 1,numU )
324             WRITE ("\t\t var_dec[%d]:\t %g \t %g \t %g \n", i, xlow[i], dec_var
    [i], xupp[i])
325         END FOR
326         WRITE ("\n\n\t\t------------------- cost function and constraints after
     optimization \n")
327         FOR ( i IN 1,numC + 1 )
328             WRITE ("\t\t F_optim[%d]:\t %g \t %g \t %g \n", i, Flow[i], F_optim
    [i], Fupp[i])
329         END FOR
330         WRITE ("\n\t\t------------------- end   \n\n")
331 
332 END EXPERIMENT
```

A.8. Codigo del MPC

El controlador predictivo se programo en EcosimPro ®Se generaron clases asociadas al estimador de estados de horizonte movil y al controlador predictivo en si. Las llamas se hacen desde un fichero principal que se comunica mediante OPC con el SCADA de la planta.

A.8.1. Componente fichero principal

```
1  USE MATH
2
3  COMPONENT mpcReactorOPC (INTEGER Nu = 3,INTEGER MV = 2, INTEGER PV = 2)
4
5  DATA
6
7      --Iniciacion de variables de proceso
8      REAL T_iniciacion = 30
9      REAL Tc_iniciacion = 10
10     REAL Cb_iniciacion = 0.1
11
12     -- Parametros del controlador
13     REAL t_Sample = 0.5             "Tiempo de muestro (min)"
14     REAL norma[4] = {60, 5, 2, 15}  "Normalizacion de las variables"
15     REAL uq[Nu] = 1.2
16     REAL uqc[Nu] = 9
17     REAL Pred_h = 30
18     REAL Liminfq = 0.3
19     --REAL Limsupq = 3.2
20     REAL Limsupq = 1.5
21     REAL Liminfqc = 0.83
22     REAL Limsupqc = 15
23     REAL LiminfT = 5
24     REAL LimsupT = 60
25     REAL LiminfCb = 0
26     REAL LimsupCb = 5
27     -- Consignas
28     REAL T_sp = 30
29     REAL Cb_sp = 0.4
30     REAL gamma[2] = {100, 100}              "Importancia relativa de los
       setpoint"
31     -- Variables manipuladas
32     REAL beta[2] =  {0.1, 0.1}              "Penalizacion de cambios"
33
34 DECLS
35     -- Variables del proceso
36     DISCR REAL q          "Caudal reactivos (L/min)"
37     DISCR REAL qc         "Caudal reqcigerante (L/min)"
38     DISCR REAL T0, Tc0  "Temperaturas de entrada (C)"
39     DISCR REAL T, Tc        "Temperaturas (C)"
40     DISCR REAL Ca           "Concentracion de A (mol/L)"
41     DISCR REAL Cb           "Concentracion de B (mol/L)"
42
43     -- Codigo adicional para el controlador predictivo
44     BOOLEAN sample = TRUE
45     BOOLEAN ControlFlag = FALSE
46     DISCR REAL state[9]                  -- state variables + algebraic
47     DISCR REAL u_new[MV*Nu]     -- Vector of decision variables computed by the
        MPC
48     -- Upper and lower bounds of the manipulated and controlled variables
49     DISCR REAL lim_manip_up[MV], lim_manip_low[MV], lim_con_up[PV], lim_con_low
       [PV]
50     DISCR REAL uqant               "Accion de control anterior"
```

```
51     DISCR REAL uqcant               "Accion de control anterior"
52     DISCR REAL tab_q[5], tab_qc[5], tab_T0[5], tab_Tc0[5], tab_T[5], tab_Tc[5],
        tab_Cb[5]
53     DISCR REAL estim[4]          "Estados estimados"
54     DISCR REAL ini_estim[16]     "Iniciacion del MHE"
55     DISCR REAL config[6]             "Configuracion del controlador"
56
57 OBJECTS
58 -- Declaracion de objetos
59 optimiz controlador
60 optimiz_MHE estimador
61
62 END COMPONENT
```

A.8.2. Experimento componente principal

```
1  EXPERIMENT ControladorOPC ON mpcReactorOPC.partition1
2     DECLS
3          REAL q_sp
4          REAL qc_sp
5          INTEGER FlagControlador = 0
6     OBJECTS
7     INIT
8     BOUNDS
9     BODY
10
11
12 -- constraints MVs
13    lim_manip_low[1] = Liminfq        -- limites  q
14     lim_manip_up[1] = Limsupq
15
16     lim_manip_low[2] = Liminfqc      -- limites  Fr
17    lim_manip_up[2] = Limsupqc
18
19 -- Constraints process variables
20     lim_con_up[1]  = LimsupT          -- limites temperatura reactor
21     lim_con_low[1] = LiminfT
22
23     lim_con_up[2] = LimsupCb           -- limites conversion
24     lim_con_low[2] = LiminfCb
25
26 -- Initial values MVs
27
28     FOR (i IN 1,Nu)
29         u_new[MV*(i-1)+1] = uq[i]
30         u_new[MV*(i-1)+2] = uqc[i]
31     END FOR
32
33     FOR (i IN 1,16)
34         ini_estim[i] = 0
35     END FOR
36
37 -- Initialization of the sensitivities
38
39    controlador.iniciar()
40
41     -- Recoleccion acciones de control pasadas
42     --uqant = q
43     --uqcant = qc
44
45     -- Actualizacion de datos del MHE
46     FOR (i IN 2,5)
47         tab_q[i-1] = tab_q[i]
```

```
48          tab_qc[i-1] = tab_qc[i]
49          tab_T0[i-1] = tab_T0[i]
50          tab_Tc0[i-1] = tab_Tc0[i]
51          tab_T[i-1] = tab_T[i]
52          tab_Tc[i-1] = tab_Tc[i]
53          tab_Cb[i-1] = tab_Cb[i]
54      END FOR
55
56      tab_q[5] = q
57      tab_qc[5] = qc
58      tab_T0[5] = T0
59      tab_Tc0[5] = Tc0
60      tab_T[5] = T
61      tab_Tc[5] = Tc
62      tab_Cb[5] = Cb
63
64      -- Llamada al MHE
65      IF FlagControlador == 1 THEN
66
67          estimador.ejecutar(t_Sample, tab_q, tab_qc, tab_T0, tab_Tc0, tab_T,
    tab_Tc, tab_Cb, estim, ini_estim)
68
69          -- Vector de informacion para el controlador
70          state[1] = T
71          state[2] = Tc
72          state[3] = Cb
73          state[4] = estim[1]
74          state[5] = estim[2]
75          state[6] = estim[3]
76          state[7] = estim[4]
77          state[8] = uqant
78          state[9] = uqcant
79
80          -- Configuracion del controlador
81          config[1] = T_sp
82          config[2] = Cb_sp
83          config[3] = gamma[1]
84          config[4] = gamma[2]
85          config[5] = beta[1]
86          config[6] = beta[2]
87
88          -- Llamada al controlador
89          controlador.control ( t_Sample, Pred_h, state, lim_manip_low,
    lim_manip_up, lim_con_low, lim_con_up, MV, config, u_new )
90
91          setValueReal("uq[1]",u_new[1])
92          setValueReal("uqc[1]", u_new[2])
93          setValueReal("uq[2]", u_new[3])
94          setValueReal("uqc[2]", u_new[4])
95          setValueReal("uq[3]", u_new[5])
96          setValueReal("uqc[3]", u_new[6])
97
98          -- Recoleccion acciones de control
99          q = uq[1]
100         qc = uqc[1]
101
102
103
104     END IF
105
106     q_sp = q
107     qc_sp = qc
108
```

```
109
110     INTEG()
111     TIME = 0
112     TSTOP = 0
113
114 END EXPERIMENT
```

A.8.3. Componente estimador de estados

```
1  USE MATH
2  USE OPTIM_METHODS
3  COMPONENT ReactorMHE
4
5      DATA
6          -- Parametros del modelo
7          REAL V = 11.5           "Volumen del reactor (L)"
8          REAL Vc = 1             "Volumen de la camisa (L)"
9          REAL R = 0.00831        "Constante de los gases (kJ/mol/K)"
10         REAL rho = 1            "Densidad del agua (kg/L)"
11         REAL Cp = 4.184     "Capacidad Calorifica del agua (kJ/kg/C)"
12       REAL Ca0 = 5          "Concentracion de entrada de A (mol/L)"
13
14         -- Parametros estimados
15         REAL k10 = 20
16     REAL k20 = 9.66637673
17     REAL k30 = 10.9322785
18         REAL dHrxn1 = -53.285068/11.5
19     REAL dHrxn2 = -42.2083961/11.5
20     REAL dHrxn3 = -13.1949792/11.5
21         REAL Ea1 = 8.91987158
22         REAL Ea2 = 6.7666094
23         REAL Ea3 = 8.72110974
24         REAL alpha = 1.22916238
25
26         REAL t_Sample = 0.5
27         REAL Sig = 100
28
29         -- Parametros a estimar
30         REAL Ca_inicial = 0.1
31         -- Perturbaciones
32         REAL d_T[5] = 0
33         REAL d_Tc[5] = 0
34         REAL d_Cb[5] = 0
35
36         -- Optimizador
37         REAL pesos[3] = {1, 1, 1}
38         REAL gamma[3] = {1, 1, 1}
39         REAL norma[3] = {50, 20, 5}
40         REAL LiminfC = 0
41         REAL LimsupC = 5
42         REAL C = 5
43         REAL Limsupd[3] = {50, 50, 50}
44         REAL Liminfd[3] = {-50, -50, -50}
45
46         -- Tablas de datos
47         TABLE_1D tab1 = {{0, 0.5, 1, 1.5, 2}, {3.2, 3.2, 3.2, 3.2, 3.2}}     --
       q
48         TABLE_1D tab2 = {{0, 0.5, 1, 1.5, 2}, {6.41, 6.41, 6.41, 6.41, 6.41}}
          -- qc
49         TABLE_1D tab3 = {{0, 0.5, 1, 1.5, 2}, {10, 10, 10, 10, 10}}  -- T0
50         TABLE_1D tab4 = {{0, 0.5, 1, 1.5, 2}, {10, 10, 10, 10, 10}}  -- Tc0
51         TABLE_1D tab5 = {{0, 0.5, 1, 1.5, 2}, {49.41, 49.51, 49.60, 49.67,
     49.73}}   -- T
```

```
52          TABLE_1D tab6 = {{0, 0.5, 1, 1.5, 2}, {16.59, 16.61, 16.62, 16.64,
    16.65}}   -- Tc
53          TABLE_1D tab7 = {{0, 0.5, 1, 1.5, 2}, {0.7897, 0.7887, 0.7884, 0.7881,
    0.7878}}  -- Cb
54
55      DECLS
56          -- Caudales
57          REAL q                "Caudal reactivos (L/min)"
58          REAL qc               "Caudal refrigerante (L/min)"
59
60          -- Salidas medidas del proceso
61          REAL T                 "Temperatura reactor (C)"
62          REAL T0
63          REAL Tc                "Temperatura refrigerante (C)"
64          REAL Tc0
65
66          -- Otras variables del modelo
67          REAL Q                  "Calor transferido (kJ/mol)"
68          REAL Heat_rxn           "Calor generado (kJ/min)"
69          REAL Ca, Cb, Cc, Cd "Concentracion de las especies (mol/L)"
70          REAL r1, r2, r3     "Velocidad de reaccion"
71          REAL UA                 "Coeficiente de transferencia de calor"
72          REAL d[3]               "Perturbaciones"
73
74          -- Optimizador
75          REAL F_optim[10]        "Restricciones"
76          DISCR REAL J_costo  "Funcion de coste"
77          REAL y_real[3], y_modelo[3]
78          BOOLEAN Sample = TRUE
79
80      INIT
81
82          -- Lectura de variables
83          -- Variables de entrada
84          q = interp1D(tab1, CONSTANT, CONSTANT, TIME)
85          qc = interp1D(tab2, CONSTANT, CONSTANT, TIME)
86          T0 = linearInterp1D(tab3, TIME)
87          Tc0 = linearInterp1D(tab4, TIME)
88          -- Variables de salida
89          y_real[1] = linearInterp1D(tab5, TIME)
90          y_real[2] = linearInterp1D(tab6, TIME)
91          y_real[3] = linearInterp1D(tab7, TIME)
92
93          y_modelo[1] = y_real[1]
94          y_modelo[2] = y_real[2]
95          y_modelo[3] = y_real[3]
96
97          T = y_real[1]
98          Tc = y_real[2]
99          Cb = linearInterp1D(tab7, TIME)
100
101         -- Parametros a estimar
102         Ca = Ca_inicial
103
104     DISCRETE
105
106         WHEN (Sample) THEN
107             J_costo += SUM(i IN 1,3; 100*pesos[i]*((y_modelo[i]-y_real[i])/
    norma[i])**2 + gamma[i]*d[i]**2)
108         Sample = FALSE
109         Sample = TRUE AFTER t_Sample
110     END WHEN
111
```

```
112     CONTINUOUS
113
114         -- Lectura de variables
115         -- Variables de entrada
116         q = interp1D(tab1, CONSTANT, CONSTANT, TIME)
117         qc = interp1D(tab2, CONSTANT, CONSTANT, TIME)
118         T0 = linearInterp1D(tab3, TIME)
119         Tc0 = linearInterp1D(tab4, TIME)
120         -- Variables de salida
121         y_real[1] =linearInterp1D(tab5, TIME)
122         y_real[2] = linearInterp1D(tab6, TIME)
123         y_real[3] = linearInterp1D(tab7, TIME)
124
125         -- Perturbaciones
126         d[1] = d_T[1] + SUM(i IN 2,5; (d_T[i] - d_T[i-1])*sigmoide(TIME,
    t_Sample,Sig))
127         d[2] = d_Tc[1] + SUM(i IN 2,5; (d_Tc[i] - d_Tc[i-1])*sigmoide(TIME,
    t_Sample,Sig))
128         d[3] = d_Cb[1] + SUM(i IN 2,5; (d_Cb[i] - d_Cb[i-1])*sigmoide(TIME,
    t_Sample,Sig))
129
130         -- Balances de materia
131         V*Ca' = q*(Ca0 - Ca) + (-r1 - 2*r3)
132         V*Cb' = -q*Cb + ( r1 -   r2) + d[3]
133         V*Cc' = -q*Cc + ( r2       )
134         V*Cd' = -q*Cd + (        r3)
135         r1 = k10*exp(-Ea1/(R*(T+273.15)))*Ca*V
136         r2 = k20*exp(-Ea2/(R*(T+273.15)))*Cb*V
137         r3 = k30*exp(-Ea3/(R*(T+273.15)))*Ca**2*V
138
139         -- Balances de energia
140         -- Reactor
141         rho*Cp*V*T' = q*rho*Cp*(T0 - T) - Q + Heat_rxn + d[1]
142         -- Camisa
143         rho*Cp*Vc*Tc' = qc*rho*Cp*(Tc0 - Tc) + Q + d[2]
144         -- Transferencia de calor
145         Q = UA*(T - Tc)
146         UA = alpha*qc**0.8
147         -- Calor generado por la reaccion
148         Heat_rxn = V*( - dHrxn1*r1 - dHrxn2*r2 - dHrxn3*r3 )
149
150         -- Restricciones no lineales (expresiones  c(x) <= 0)
151         F_optim[1] = J_costo
152         F_optim[2] = MATH.min(Ca,LiminfC) - LiminfC
153        F_optim[3] = MATH.max(Ca,LimsupC) - LimsupC
154         F_optim[4] = MATH.min(Cb,LiminfC) - LiminfC
155        F_optim[5] = MATH.max(d[1],Limsupd[1]) - Limsupd[1]
156         F_optim[6] = MATH.min(d[1],Liminfd[1]) - Liminfd[1]
157         F_optim[7] = MATH.max(d[2],Limsupd[2]) - Limsupd[2]
158         F_optim[8] = MATH.min(d[2],Liminfd[2]) - Liminfd[2]
159         F_optim[9] = MATH.max(d[3],Limsupd[3]) - Limsupd[3]
160         F_optim[10] = MATH.min(d[3],Liminfd[3]) - Liminfd[3]
161
162         -- Asignacion de valores del modelo
163         y_modelo[1] = T
164         y_modelo[2] = Tc
165         y_modelo[3] = Cb
166
167 END COMPONENT
```

A.8.4. Clase estimador de estados

```
1 USE MATH
```

```
2  USE OPTIM_METHODS
3
4  CLASS optimiz_MHE IS_A ReactorMHE_partition1
5
6  DECLS
7      CONST INTEGER numC = 9          -- numero de restricciones del problema de
       optimizacion
8      CONST INTEGER numU = 16         -- numero de variables de decision
9
10 OBJECTS
11
12 METHODS
13
14     METHOD INTEGER coste_y_restricciones(IN REAL esnopt_x[], IN INTEGER needF,
       OUT REAL esnopt_F[], IN INTEGER explicit_derivatives, IN INTEGER needG, OUT
        REAL esnopt_G[])
15
16     BODY
17
18         RESET_VARIABLES ()
19
20         Ca_inicial = esnopt_x[1]
21         FOR (i IN 1,5)
22             d_T[i] = esnopt_x[1+i]
23         END FOR
24         FOR (i IN 1,5)
25             d_Tc[i] = esnopt_x[6+i]
26         END FOR
27         FOR (i IN 1,5)
28             d_Cb[i] = esnopt_x[11+i]
29         END FOR
30         SET_INIT_ACTIVE(TRUE)
31         TIME  = 0
32         TSTOP = 4*t_Sample
33         CINT  = t_Sample
34         INTEG()
35
36         FOR (i IN 1,numC)
37             esnopt_F[i] = F_optim[i]
38         END FOR
39
40     RETURN 0
41     END METHOD
42
43     METHOD NO_TYPE ejecutar (IN REAL t_sam, IN REAL tab_q[], IN REAL tab_qc[],
       IN REAL tab_T0[], \
44                                  IN REAL tab_Tc0[], IN REAL tab_T[], IN
       REAL tab_Tc[], IN REAL tab_Cb[], \
45                                  OUT REAL estim[], OUT REAL inicializacion
       [])
46
47     DECLS
48         REAL param_estim[numU]      -- variables de decision, size n_dec_var
49         REAL xlow[numU]            -- valor inferior de las variables de
       decision, size n_dec_var
50         REAL xupp[numU]            -- valor superior de las variables de
       decision, size n_dec_var
51         REAL Flow[numC+1]           -- valor inferior de la funcion objetivo y
        las restricciones, size numC+1
52         REAL Fupp[numC+1]           -- valor superior de la funcion objetivo y
        las restricciones, size numC+1
53
```

```
54      OBJECTS
55      BODY
56
57          t_Sample = t_sam
58          DEBUG_LEVEL = 0
59          setSilentMode(TRUE)
60          SET_REPORT_ACTIVE("#MONITOR",FALSE)
61
62          FOR (i IN 1,5)
63              tab1.changeValue(i,i,1, tab_q[i])
64              tab2.changeValue(i,i,1, tab_qc[i])
65              tab3.changeValue(i,i,1, tab_T0[i])
66              tab4.changeValue(i,i,1, tab_Tc0[i])
67              tab5.changeValue(i,i,1, tab_T[i])
68              tab6.changeValue(i,i,1, tab_Tc[i])
69              tab7.changeValue(i,i,1, tab_Cb[i])
70          END FOR
71          setSilentMode(FALSE)
72          SET_REPORT_ACTIVE("#MONITOR",TRUE)
73          -- Se pueden dar valores de iniciacion de las variables. Mejora la
    velocidad
74          -- Inicializaciones
75          Ca_inicial  = inicializacion[1]
76          FOR (i IN 1,4)
77              d_T[i] = inicializacion[2+i]
78          END FOR
79          d_T[5] = inicializacion[6]
80          FOR (i IN 1,4)
81              d_Tc[i] = inicializacion[7+i]
82          END FOR
83          d_Tc[5] = inicializacion[11]
84          FOR (i IN 1,4)
85              d_Cb[i] = inicializacion[12+i]
86          END FOR
87          d_Cb[5] = inicializacion[16]
88
89      --  Formar vector de variables de decision
90          param_estim[1] = Ca_inicial
91          FOR (i IN 1,5)
92              param_estim[1+i] = d_T[i]
93          END FOR
94          FOR (i IN 1,5)
95              param_estim[6+i] = d_Tc[i]
96          END FOR
97          FOR (i IN 1,5)
98              param_estim[11+i] = d_Cb[i]
99          END FOR
100
101         --  Limites de las variables de decision
102         xlow[1] = LiminfC
103         xupp[1] = LimsupC
104         FOR (i IN 1,5)
105             xlow[1+i] = Liminfd[1]
106             xupp[1+i] = Limsupd[1]
107         END FOR
108         FOR (i IN 1,5)
109             xlow[6+i] = Liminfd[2]
110             xupp[6+i] = Limsupd[2]
111         END FOR
112         FOR (i IN 1,5)
113             xlow[11+i] = Liminfd[3]
114             xupp[11+i] = Limsupd[3]
115         END FOR
```

```
116
117         -- Restricciones
118         Flow[1] = -1e10
119         Fupp[1] =  1e10
120         FOR (i IN 2,numC+1)
121             Flow[i] = 0
122             Fupp[i] = 0
123         END FOR
124
125         --Optimization extern routine call
126         setSilentMode(TRUE)
127         SET_REPORT_ACTIVE("#MONITOR",FALSE)
128
129         esnopt_init(numU, numC)
130         esnopt_set_variables_bounds_and_initial_values(xlow, xupp, param_estim)
131         esnopt_set_constraints_bounds_and_initial_values(Flow, Fupp, F_optim)
132         esnopt_set_cost_function_and_constraints(coste_y_restricciones)
133         esnopt_set_explicit_derivatives(0)
134         esnopt_set_function_precision(1.0e-4)
135         esnopt_set_iterations_limit(100)
136         esnopt( )
137         REL_ERROR = 1.0e-5
138
139         RESET_VARIABLES ()
140
141         esnopt_get_variables_values(param_estim)
142         esnopt_free()
143         setSilentMode(FALSE)
144         SET_REPORT_ACTIVE("#MONITOR",TRUE)
145
146         Ca_inicial = param_estim[1]
147         FOR (i IN 1,5)
148             d_T[i] = param_estim[1+i]
149         END FOR
150         FOR (i IN 1,5)
151             d_Tc[i] = param_estim[6+i]
152         END FOR
153         FOR (i IN 1,5)
154             d_Cb[i] = param_estim[11+i]
155         END FOR
156
157         SET_INIT_ACTIVE(TRUE)
158         TIME = 0
159         TSTOP = 4*t_Sample
160         CINT = t_Sample
161         INTEG()
162
163         estim[1] = Ca
164         estim[2] = d[1]
165         estim[3] = d[2]
166         estim[4] = d[3]
167
168         inicializacion[1] = Ca_inicial
169         FOR (i IN 1,5)
170         inicializacion[1+i] = d_T[i]
171         END FOR
172         FOR (i IN 1,5)
173         inicializacion[6+i] = d_Tc[i]
174         END FOR
175                 FOR (i IN 1,5)
176         inicializacion[11+i] = d_Cb[i]
177         END FOR
178     END METHOD
```

```
END CLASS
```

A.8.5. Componente controlador

```
USE MATH
COMPONENT CostReactorHibrido (INTEGER Nu = 3)

    DATA
        -- Parametros del modelo
        REAL V = 11.5           "Volumen del reactor (L)"
        REAL Vc = 1             "Volumen de la camisa (L)"
        REAL R = 0.00831        "Constante de los gases (kJ/mol/K)"
        REAL rho = 1            "Densidad del agua (kg/L)"
        REAL Cp = 4.184     "Capacidad Calorifica del agua (kJ/kg/C)"
      REAL Ca0 = 5          "Concentracion de entrada de A (mol/L)"
        REAL T0 = 10            "Temperatura de entrada de los reactivos (C)"
        REAL Tc0 = 10           "Temperatura de entrada de los reactivos (C)"

        -- Parametros estimados
        REAL k10 = 20
    REAL k20 = 9.66637673
    REAL k30 = 10.9322785
        REAL dHrxn1 = -53.285068/11.5
    REAL dHrxn2 = -42.2083961/11.5
    REAL dHrxn3 = -13.1949792/11.5
        REAL Ea1 = 8.91987158
        REAL Ea2 = 6.7666094
        REAL Ea3 = 8.72110974
        REAL alpha = 1.22916238

        -- Optimizador
        REAL LiminfT = 5
        REAL LimsupT = 60
        REAL LiminfCb = 0
        REAL LimsupCb = 5
        REAL Liminfq = 0.3
        --REAL Limsupq = 3.2
        REAL Limsupq = 1.5
        REAL Liminfqc = 0.83
        REAL Limsupqc = 15

        -- Parametros del controlador
        REAL t_Sample = 0.5                     "Tiempo de muestro (min)"
        REAL Sig = 100                              "Parametro sigmoide"

        -- Acciones de control
        REAL uq[Nu] = 1
        REAL uqc[Nu] = 7

    DECLS
        -- Caudales
        REAL q                "Caudal reactivos (L/min)"
        REAL qc               "Caudal reqcigerante (L/min)"

        -- Salidas medidas del proceso
        REAL T                 "Temperatura reactor (C)"
        REAL Tc                "Temperatura reqcigerante (C)"

        -- Otras variables del modelo
        REAL Q                  "Calor transferido (kJ/mol)"
        REAL Heat_rxn           "Calor generado (kJ/min)"
        REAL Ca, Cb, Cc, Cd "Concentracion de las especies (mol/L)"
        REAL r1, r2, r3     "Velocidad de reaccion"
```

```
60          REAL UA                 "Coeficiente de transferencia de calor"
61
62          -- Optimizador
63          REAL F_optim[5]     "Restricciones"
64          REAL J_costo            "Funcion de coste"
65          DISCR REAL T_inic = 10
66          DISCR REAL Tc_inic = 10
67          DISCR REAL Ca_inic = 0
68          DISCR REAL Cb_inic = 0
69          DISCR REAL Cc_inic = 0
70          DISCR REAL Cd_inic = 0
71          DISCR REAL d_T, d_Tc, d_Cb      "Perturbaciones"
72          DISCR REAL uqant                    "Accion de control anterior"
73          DISCR REAL uqcant                   "Accion de control anterior"
74          DISCR REAL omega[2], gamma[2], beta[2]      "Parametros del controlador
    "
75          REAL Set_points                 "Componente funcion de coste"
76          REAL Cambios                        "Componente funcion de coste"
77
78      INIT
79
80          T = T_inic
81          Tc = Tc_inic
82          Ca = Ca_inic
83          Cb = Cb_inic
84          Cc = Cc_inic
85          Cd = Cd_inic
86
87      CONTINUOUS
88
89          q = uq[1] + (uq[2]-uq[1])*sigmoide(TIME,t_Sample,Sig) + (uq[3]-uq[2])*
    sigmoide(TIME,2*t_Sample,Sig)
90          qc = uqc[1] + (uqc[2]-uqc[1])*sigmoide(TIME,t_Sample,Sig) + (uqc[3]-uqc
    [2])*sigmoide(TIME,2*t_Sample,Sig)
91
92          -- Balances de materia
93          V*Ca' = q*(Ca0 - Ca) + (-r1 - 2*r3)
94          V*Cb' = -q*Cb + ( r1 -   r2) + d_Cb
95          V*Cc' = -q*Cc + ( r2       )
96          V*Cd' = -q*Cd + (        r3)
97          r1 = k10*exp(-Ea1/(R*(T+273.15)))*Ca*V
98          r2 = k20*exp(-Ea2/(R*(T+273.15)))*Cb*V
99          r3 = k30*exp(-Ea3/(R*(T+273.15)))*Ca**2*V
100
101         -- Balances de energia
102         -- Reactor
103         rho*Cp*V*T' = q*rho*Cp*(T0 - T) - Q + Heat_rxn + d_T
104         -- Camisa
105         rho*Cp*Vc*Tc' = qc*rho*Cp*(Tc0 - Tc) + Q + d_Tc
106         -- Transferencia de calor
107         Q = UA*(T - Tc)
108         UA = alpha*qc**0.8
109         -- Calor generado por la reaccion
110         Heat_rxn = V*( - dHrxn1*r1 - dHrxn2*r2 - dHrxn3*r3 )
111
112         -- Optimizador
113         Set_points = gamma[1]*((T - omega[1])/(LimsupT-LiminfT))**2 + gamma
    [2]*((Cb - omega[2])/(LimsupCb - LiminfCb))**2
114         Cambios = beta[1]*((uq[1] - uqant)**2 + SUM(i IN 2,3; (uq[i]-uq[i-1])
    **2)) + \
115                   beta[2]*((uqc[1] - uqcant)**2 + SUM(i IN 2,3; (uqc[i]-uqc[
    i-1])**2))
116     -- Esta funcion de costo es solo para visualizacion
```

```
117         J_costo = Set_points + Cambios
118
119         -- Restricciones no lineales (expresiones  c(x) <= 0)
120         F_optim[1] = J_costo
121         F_optim[2] = MATH.min(T,LiminfT) - LiminfT
122        F_optim[3] = MATH.max(T,LimsupT) - LimsupT
123        F_optim[4] = MATH.min(Cb,LiminfCb) - LiminfCb
124        F_optim[5] = MATH.max(Cb,LimsupCb) - LimsupCb
125
126 END COMPONENT
```

A.8.6. Clase del controlador

```
 1
 2 USE MATH
 3 USE OPTIM_METHODS
 4
 5 CLASS optimiz IS_A CostReactorHibrido_partition1
 6
 7     DECLS
 8
 9         REAL Pred_h = 30
10         CONST INTEGER numC = 4          -- numero de restricciones del problema
       de optimizacion
11         CONST INTEGER numU = 6          -- numero de variables de decision
12
13     OBJECTS
14
15         EIDAS calculoSens
16
17    METHODS
18
19      METHOD NO_TYPE funcionResiduos( IN REAL tiempo, IN REAL var[], IN REAL der
       [], IN INTEGER numPar, IN STRING nomPar[], IN REAL par[], OUT REAL res[] )
20             DECLS
21             OBJECTS
22             BODY
23                 FOR( i IN 1,numPar )
24                     setValueReal( nomPar[i], par[i] )
25                 END FOR
26
27                 FRES_ARGS( tiempo, var, der, res, FALSE )
28
29         END METHOD
30
31 -- llamada a las funciones de cuadraturas por parte de IDAS, para el calculo de
       sensibilidades
32 -- utilizar el vector F_optim que se actualiza al llamar a los residuos
33         METHOD NO_TYPE funcionQuadraturas( IN REAL tiempo, IN REAL var[], IN
       REAL der[], IN INTEGER numPar, IN STRING nomPar[], IN REAL par[], OUT REAL
       quad[] )
34             DECLS
35             OBJECTS
36             BODY
37                 FOR( i IN 1,numPar )
38                     setValueReal( nomPar[i], par[i] )
39                 END FOR
40
41                 FRES_ARGS( tiempo, var, der, der, FALSE )
42
43         -- funciones de cuadraturas
44                 FOR( i IN 1, numC+1 )
45                     quad[i] = F_optim[i]
```

```
46                END FOR
47
48            END METHOD
49
50            METHOD NO_TYPE iniciar()
51                DECLS
52                OBJECTS
53                    VECTOR_STRING nombresX
54                BODY
55                    calculoSens.addU( "uq[1]",  uq[1], FALSE)
56                    calculoSens.addU("uqc[1]", uqc[1], FALSE)
57                    calculoSens.addU( "uq[2]",  uq[2], FALSE)
58                    calculoSens.addU("uqc[2]", uqc[2], FALSE)
59                    calculoSens.addU( "uq[3]",  uq[3], FALSE)
60                    calculoSens.addU("uqc[3]", uqc[3], FALSE)
61
62                    EXEC_INIT()
63
64                    nombresX = getVariablesWithCategory( T_JACOBIAN_VARS )
65                    FOR( i IN 1,nombresX.size() )
66                        calculoSens.addX( nombresX.at(i), getValueReal( nombresX.at
    (i) ), ( getVarCategory (nombresX.at(i)) == 2 ) )
67                    END FOR
68
69                    calculoSens.iniciarRes(TIME, funcionResiduos)
70                    calculoSens.iniciarQuad(numC + 1, funcionQuadraturas)
71                    calculoSens.setTol(1e-05)
72
73            END METHOD
74
75             METHOD INTEGER coste_y_restricciones (IN REAL esnopt_x[], IN INTEGER needF
    , OUT REAL esnopt_F[], IN INTEGER explicit_derivatives, IN INTEGER needG,
    OUT REAL esnopt_G[])
76
77                DECLS
78                    REAL arr_quad[ numC + 1 ]                        -- array de
    integrales temporales
79                    REAL arr_quadsen[ (numC + 1) * numU ]       -- array de
    sensibilidad de las integrales
80                    REAL arr_quadsen_p[ (numC + 1) * numU ] -- array de la derivada
     temporal de la sensibilidad de las integrales
81                OBJECTS
82
83                BODY
84                    FOR( j IN 1,numC + 1 )
85                        arr_quad[ j ] = 0
86                    END FOR
87                    FOR( j IN 1, (numC + 1) * numU )
88                        arr_quadsen[ j ] = 0
89                        arr_quadsen_p[ j ] = 0
90                    END FOR
91
92            -- Actualizar las variables de decision a los valores que propone el
    optimizador
93                    calculoSens.setU( "uq[1]", esnopt_x[1])
94                    calculoSens.setU("uqc[1]", esnopt_x[2])
95                    calculoSens.setU( "uq[2]", esnopt_x[3])
96                    calculoSens.setU("uqc[2]", esnopt_x[4])
97                    calculoSens.setU( "uq[3]", esnopt_x[5])
98                    calculoSens.setU("uqc[3]", esnopt_x[6])
99
100                   SET_INIT_ACTIVE(TRUE)
101                   TSTOP = Pred_h
```

```
102                 TIME = 0
103
104             -- Reiniciar el calculo de sensibilidades con los nuevos valores de los
     parametros
105                 calculoSens.reiniciar()
106                 calculoSens.calculateSens( TSTOP )
107                 -- calculo de las cuadraturas y sus sensibilidades
108                 calculoSens.getQuad( arr_quad, arr_quadsen )
109                 -- calculo de la derivada de las sensibilidades de las
    cuadraturas
110                 calculoSens.getQuadN( arr_quadsen_p, 1 )
111
112             --  Introduccion de los gradientes de la funcion de costo y
    restricciones a SNOPT
113                 IF ( explicit_derivatives == 1) THEN
114                     --FOR (i IN 1,6)
115                     --  esnopt_G[i] = arr_quadsen_p[i]
116                     --END FOR
117                     FOR (i IN 1,(numC+1)*numU)
118                         esnopt_G[i] = arr_quadsen[i]
119                     END FOR
120
121                 END IF
122
123             -- Introduccion de los valores de las funcines a SNOPT
124                     --esnopt_F[1] = F_optim[1]
125                 FOR (i IN 1, numC+1)
126                     esnopt_F[i] = arr_quad[i]
127                 END FOR
128
129             RETURN 0
130
131             END METHOD
132
133             METHOD NO_TYPE control (IN REAL t_sam, IN REAL t_pred, IN REAL state[],
     IN REAL Lu_i[], IN REAL Lu_s[], IN REAL Ly_i[], IN REAL Ly_s[],
134                                     IN INTEGER MV, IN REAL config[], OUT REAL con[])
135
136                 DECLS
137                     REAL dec_var[numU]          -- valor inicial de las
    variables de decision, size numU. Necesario para las llamadas a ESnopt
138                     REAL xlow[numU]             -- valor inferior de las
    variables de decision, size numU. Necesario para las llamadas a ESnopt
139                 REAL xupp[numU]             -- valor superior de las variables
    de decision, size numU. Necesario para las llamadas a ESnopt
140                     REAL Flow[numC + 1]         -- valor inferior de la funcion
     objetivo y las restricciones, size (numC + funcion de coste). Necesario
    para las llamadas a ESnopt
141                 REAL Fupp[numC + 1]         -- valor superior de la funcion
    objetivo y las restricciones, size (numC + funcion de coste). Necesario
    para las llamadas a ESnopt
142                     INTEGER calcularSens = 1
143                     INTEGER infoESnopt = 0
144
145             OBJECTS
146
147                  VECTOR_STRING nombresX     -- variable auxiliar para la
    inicializacion de las sensibilidades
148
149                 BODY
150                 DEBUG_LEVEL = 0
151                     t_Sample = t_sam
152                     Pred_h = t_pred
```

```
153
154 --  Decision variables for this sampling period are initialized to the ones of
    the
155 -- previous period desplaced one sampling time. The index on the
156                   FOR ( i IN 1,numU - MV )
157                         dec_var[i]= con[MV+i]
158                     END FOR
159                   FOR ( i IN numU - MV + 1, numU )
160                         dec_var[i]= con[i]
161                     END FOR
162
163         -- Inicializaciones de q y Fr, Nu = 3
164
165                 FOR ( j IN 1, MV)
166                     FOR ( i IN 1,Nu )
167                         xlow[MV*(i-1) + j] = Lu_i[j]    -- limites  de las MV
168                     xupp[MV*(i-1) + j] = Lu_s[j]
169                     END FOR
170                 END FOR
171
172                 setValueReal( "uq[1]", dec_var[1])
173                 setValueReal("uqc[1]", dec_var[2])
174             setValueReal( "uq[2]", dec_var[3])
175                 setValueReal("uqc[2]", dec_var[4])
176                 setValueReal( "uq[3]", dec_var[5])
177                 setValueReal("uqc[3]", dec_var[6])
178
179                 -- Parametros del controlador
180                 setValueReal("omega[1]", config[1])
181                 setValueReal("omega[2]", config[2])
182                 setValueReal("gamma[1]", config[3])
183                 setValueReal("gamma[2]", config[4])
184                 setValueReal("beta[1]", config[5])
185                 setValueReal("beta[2]", config[6])
186
187                 -- Estados y variables manipuladas
188                 setValueReal("T_inic", state[1])
189                 setValueReal("Tc_inic", state[2])
190                 setValueReal("Ca_inic", state[3])
191                 setValueReal("Cb_inic", state[4])
192                 setValueReal("d_T", state[5])
193                 setValueReal("d_Tc", state[6])
194                 setValueReal("d_Cb", state[7])
195                 setValueReal("uqant", state[8])
196                 setValueReal("uqcant", state[9])
197
198                 SET_INIT_ACTIVE(TRUE)
199                 EXEC_INIT()
200                 SET_INIT_ACTIVE(TRUE)
201
202                 nombresX = getVariablesWithCategory( T_JACOBIAN_VARS )
203                 FOR( i IN 1,nombresX.size() )
204                     calculoSens.setX( nombresX.at(i), getValueReal( nombresX.at
    (i) ) )
205                 END FOR
206
207         TSTOP = Pred_h
208         CINT = t_Sample
209            TIME = 0         --  Tiempo del modelo
210
211             calculoSens.iniciarRes(TIME, funcionResiduos)
212             calculoSens.iniciarQuad(numC + 1, funcionQuadraturas)
213             calculoSens.setTol( 1e-05 )
```

```
214
215         -- inicializacion de los limites de las restricciones y la funcion de
        coste
216             Flow[1] = -1.0e10
217             Fupp[1] =  1.0e10
218
219             FOR (i IN 2, numC)
220                 Flow[i] = 0
221                 Fupp[i] = 0
222             END FOR
223
224     --Optimization extern routine call
225             setSilentMode(TRUE)
226             SET_REPORT_ACTIVE("#MONITOR",FALSE)
227
228             esnopt_init (numU, numC)
229             esnopt_set_variables_bounds_and_initial_values (xlow, xupp, dec_var
        )
230             esnopt_set_constraints_bounds_and_initial_values (Flow, Fupp,
        F_optim)
231             esnopt_set_cost_function_and_constraints (coste_y_restricciones)
232             esnopt_set_explicit_derivatives ( calcularSens )
233             esnopt_set_function_precision ( 1.0e-4 )
234             esnopt_set_iterations_limit (200)
235             infoESnopt = esnopt ()
236
237         -- Final de la optimizacion, obtencion de los resultados para la
        simulacion.
238             esnopt_get_variables_values(dec_var)
239             esnopt_free ()
240             setSilentMode(FALSE)
241             SET_REPORT_ACTIVE("#MONITOR",TRUE)
242
243             FOR ( i IN 1,numU )
244                 con[i]  = dec_var[i]
245             END FOR
246
247         END METHOD
248
249 END CLASS
```

A.9. Codigo InterfazOPC

```
from opcua import Client
from opcua import ua
import OpenOPC
import pywintypes
import csv
import time
import matplotlib.pyplot as plt
import tkinter as tk

# Linea necesaria para que funcione la comunicacion con OPC-DA
pywintypes.datetime = pywintypes.TimeType

# Ventana grafica
root = tk.Tk()
root.title("MPC Reactor Hibrido")
canvas1 = tk.Canvas(root, width=300, height=300)
canvas1.pack()
button1 = tk.Button(root, text='Cerrar Aplicacion', command=exit)
canvas1.create_window(150, 280, window=button1)
label = tk.Label(text="Controlador en ejecucion...")
label.place(x=0, y=0)

# Iniciliazacion del programa
Controlador = False         # Activa el controlador luego de recoger datos
PeriodosEjecucion = 0
t_Sample = 30

def conexion_OPC():
    # Conexion con el servidor del controlador
    MPC = Client("opc.tcp://Labo6:16700/")    # Creacion objeto OPC-UA
    MPC.connect()                             # Conexion Controlador
    MPC.load_type_definitions()

    # Conexion con los demas servidores
    SCADA = OpenOPC.client()                  # Creacion objeto OPC-DA
    PID1 = OpenOPC.client()                   # Creacion objeto OPC-DA
    PID2 = OpenOPC.client()                   # Creacion objeto OPC-DA

    SCADA.connect("LecturaOPC.1.0")
    PID1.connect("OPC.PID ISA (CreaOPC).1")
    PID2.connect("OPC.PID ISA 2 (CreaOPC).1")

    return [MPC, SCADA, PID1, PID2]

def lectura_SCADA():
    # Lectura de las variables de la planta y los parametros del controlador
    q = SCADA.read("Deck Variables.q")[0]
    qc = SCADA.read("Deck Variables.qc")[0]
    T0 = SCADA.read("Deck Variables.T0")[0]
    Tc0 = SCADA.read("Deck Variables.Tc0")[0]
    T = SCADA.read("Deck Variables.T")[0]
    Tc = SCADA.read("Deck Variables.Tc")[0]
    Ca = SCADA.read("Deck Variables.Ca")[0]
    T_sp = SCADA.read("Deck Variables.T_sp")[0]
    Ca_sp = SCADA.read("Deck Variables.Ca_sp")[0]
    gamma1 = SCADA.read("Deck Variables.gamma1")[0]
    gamma2 = SCADA.read("Deck Variables.gamma2")[0]
    beta1 = SCADA.read("Deck Variables.beta1")[0]
    beta2 = SCADA.read("Deck Variables.beta2")[0]

    return [q, qc, T0, Tc0, T, Tc, Ca, T_sp, Ca_sp, gamma1, gamma2, beta1,
    beta2]
```

```
62
63 def configuracion_MPC(T_sp, Ca_sp, gamma1, gamma2, beta1, beta2):
64     MPC.get_node("ns=4;s=T_sp").set_value(
65         ua.Variant(T_sp, ua.VariantType.Double))
66     MPC.get_node("ns=4;s=Ca_sp").set_value(
67         ua.Variant(Ca_sp, ua.VariantType.Double))
68     MPC.get_node("ns=4;s=gamma[1]").set_value(
69         ua.Variant(gamma1, ua.VariantType.Double))
70     MPC.get_node("ns=4;s=gamma[2]").set_value(
71         ua.Variant(gamma2, ua.VariantType.Double))
72     MPC.get_node("ns=4;s=beta[1]").set_value(
73         ua.Variant(beta1, ua.VariantType.Double))
74     MPC.get_node("ns=4;s=beta[2]").set_value(
75         ua.Variant(beta2, ua.VariantType.Double))
76
77     return 0
78
79 def Datos_MPC(q, qc, T0, Tc0, T, Tc, Ca):
80     MPC.get_node("ns=4;s=uqant").set_value(
81         ua.Variant(q, ua.VariantType.Double))
82     MPC.get_node("ns=4;s=uqcant").set_value(
83         ua.Variant(qc, ua.VariantType.Double))
84     MPC.get_node("ns=4;s=q").set_value(
85         ua.Variant(q, ua.VariantType.Double))
86     MPC.get_node("ns=4;s=qc").set_value(
87         ua.Variant(qc, ua.VariantType.Double))
88     MPC.get_node("ns=4;s=T0").set_value(
89         ua.Variant(T0, ua.VariantType.Double))
90     MPC.get_node("ns=4;s=Tc0").set_value(
91         ua.Variant(Tc0, ua.VariantType.Double))
92     MPC.get_node("ns=4;s=T").set_value(
93         ua.Variant(T, ua.VariantType.Double))
94     MPC.get_node("ns=4;s=Tc").set_value(
95         ua.Variant(Tc, ua.VariantType.Double))
96     MPC.get_node("ns=4;s=Ca").set_value(
97         ua.Variant(Ca, ua.VariantType.Double))
98     return 0
99
100 def ejecutar_MPC():
101     MPC.get_node("ns=2;s=server_methods").call_method(
102
103         "5:method_run", ua.Variant("A String", ua.VariantType.String))
104     return 0
105
106 def actualizar_PID():
107     # Lectura set-point de las acciones de control
108     q_sp = MPC.get_node("ns=4;s=q").get_value()
109     qc_sp = MPC.get_node("ns=4;s=qc").get_value()
110
111     # Escritura set-points al SCADA
112     PID1.write(("EscrituraLectura.SP", q_sp))
113     PID2.write(("EscrituraLectura.SP", qc_sp))
114
115     return [q_sp, qc_sp]
116
117 def reportar(q, qc):
118     print("\n" + time.strftime("%H:%M:%S") + ": Ejecucion terminada\n")
119     print(f"\t  q  = {q:.2f} L/min")
120     print(f"\t  qc = {qc:.2f} L/min")
121
122     return 0
123
124 try:
```

```
125     # Conexion a los servidores OPC
126     [MPC, SCADA, PID1, PID2] = conexion_OPC()
127 
128     while True:
129         tiempo_inicio = time.time()             # Contador de tiempo
130 
131         # Verificar si se ha activado o desactivo el controlardor desde el
        SCADA
132         Auto_MPC = SCADA.read("Deck Variables.AutoMPC")[0]
133         #Auto_MPC = 0
134         if Auto_MPC == 1:
135             Controlador = False
136             MPC.get_node("ns=4;s=FlagControlador").set_value(
137                 ua.Variant(0, ua.VariantType.Int32))
138             # Actualizacion del periodo de ejecucion
139             PeriodosEjecucion += 1
140             if PeriodosEjecucion > 6:
141                 Controlador = True
142                 MPC.get_node("ns=4;s=FlagControlador").set_value(
143                     ua.Variant(1, ua.VariantType.Int32))
144 
145             # Lectura variables de la planta y parametros del controlador
146             [q, qc, T0, Tc0, T, Tc, Ca, T_sp, Ca_sp, gamma1, gamma2, beta1,
        beta2] = lectura_SCADA()
147 
148             # Escritura de los parametros del controlador
149             configuracion_MPC(T_sp, Ca_sp, gamma1, gamma2, beta1, beta2)
150 
151             # Escritura de los datos de la planta al controlador
152             Datos_MPC(q, qc, T0, Tc0, T, Tc, Ca)
153 
154             # Ejecucion controlador, solo se soluciona el MHE y MPC si
        FlagControlador = 1
155             ejecutar_MPC()
156 
157             if Controlador == True:
158                 # Obtener acciones de control y actualizar set-poit de los PID
159                 [q_sp, qc_sp] = actualizar_PID()
160                 # Reportar en pantallas las acciones de control
161                 reportar(q_sp, qc_sp)
162             else:
163                 print("\n" + time.strftime("%H:%M:%S") + ": Controlador activo,
         recolectando datos para MHE")
164 
165             a = (time.time() - tiempo_inicio)
166             root.update()
167             time.sleep(t_Sample - (time.time() - tiempo_inicio))
168 
169         elif Auto_MPC == 0:
170             MPC.get_node("ns=4;s=FlagControlador").set_value(
171                 ua.Variant(0, ua.VariantType.Int32))
172             Controlador = False
173             PeriodosEjecucion = 0
174             print("\n" + time.strftime("%H:%M:%S") + ": Controlador desactivado
        ")
175             time.sleep(t_Sample)
176 
177 except KeyboardInterrupt:
178 
179     SCADA.close()
180     MPC.disconnect()
181     PID1.disconnect()
182     PID2.disconnect()
```

```
183
184     print("\nSe han desconectado los servidores \n")
185     exit()
186
187 finally:
188
189     SCADA.close()
190     MPC.disconnect()
191     PID1.disconnect()
192     PID2.disconnect()
193
194     print("\nSe han desconectado los servidores \n")
195     exit()
```

A.10. Optimizador en tiempo real

El optimizador en tiempo real se programo en EcosimPro ® de una forma muy similar al problema de estimacion de parametros.

A.10.1. Componente RTO

```
1  USE MATH
2  COMPONENT Reactor_RTO (INTEGER Nu = 3)
3
4      DATA
5          -- Parametros del modelo
6          REAL V = 11.5            "Volumen del reactor (L)"
7          REAL Vc = 1              "Volumen de la camisa (L)"
8          REAL R = 0.00831         "Constante de los gases (kJ/mol/K)"
9          REAL rho = 1             "Densidad del agua (kg/L)"
10         REAL Cp = 4.184      "Capacidad Calorifica del agua (kJ/kg/C)"
11       REAL Ca0 = 5           "Concentracion de entrada de A (mol/L)"
12         REAL T0 = 10             "Temperatura de entrada de los reactivos (C)"
13         REAL Tc0 = 10            "Temperatura de entrada de los reactivos (C)"
14
15         -- Parametros estimados
16         REAL k10 = 20
17     REAL k20 = 9.66637673
18     REAL k30 = 10.9322785
19         REAL dHrxn1 = -53.285068/11.5
20     REAL dHrxn2 = -42.2083961/11.5
21     REAL dHrxn3 = -13.1949792/11.5
22         REAL Ea1 = 8.91987158
23         REAL Ea2 = 6.7666094
24         REAL Ea3 = 8.72110974
25         REAL alpha = 1.22916238
26
27         -- Optimizador
28         REAL LiminfT = 5
29         REAL LimsupT = 60
30         REAL LiminfCa = 0
31         REAL LimsupCa = 5
32         REAL Liminfq = 0.3
33         REAL Limsupq = 1.5
34         REAL Liminfqc = 0.83
35         REAL Limsupqc = 15
36
37         REAL q = 1.2
38         REAL qc = 2
39
40     DECLS
```

```
41          -- Variables de decision
42          REAL T = 30
43          REAL Tc  = 15                "Temperatura refrigerante (C)"
44          REAL Ca = 0.2
45          REAL Cb = 2
46          REAL Cc = 0.3
47          REAL Cd = 0.2
48          -- Optimizador
49          REAL F_optim[5]      "Restricciones"
50          REAL J_costo             "Funcion de coste"
51          DISCR REAL c_A, c_B, c_C, c_D, c_ref
52
53      INIT
54
55      CONTINUOUS
56
57          -- Balances de materia
58          0 = q*(Ca0 - Ca) + (-k10*exp(-Ea1/(R*(T+273.15)))*Ca*V - 2*k30*exp(-Ea3
        /(R*(T+273.15)))*Ca**2*V)
59          0 = -q*Cb + ( k10*exp(-Ea1/(R*(T+273.15)))*Ca*V -   k20*exp(-Ea2/(R*(T
        +273.15)))*Cb*V)
60          0 = -q*Cc + ( k20*exp(-Ea2/(R*(T+273.15)))*Cb*V       )
61          0 = -q*Cd + (        k30*exp(-Ea3/(R*(T+273.15)))*Ca**2*V)
62
63          -- Balances de energia
64          -- Reactor
65          0 = q*rho*Cp*(T0 - T) - alpha*qc**0.8*(T - Tc) + V*( - dHrxn1*k10*exp(-
        Ea1/(R*(T+273.15)))*Ca*V - dHrxn2*k20*exp(-Ea2/(R*(T+273.15)))*Cb*V -
        dHrxn3*k30*exp(-Ea3/(R*(T+273.15)))*Ca**2*V )
66          -- Camisa
67          0 = qc*rho*Cp*(Tc0 - Tc) + alpha*qc**0.8*(T - Tc)
68
69      -- Esta funcion de costo es solo para visualizacion
70          J_costo = -( q*(c_A*Ca + c_B*Cb + c_C*Cc + c_D*Cd) - q*c_A*Ca0 - qc*
        c_ref )
71
72          -- Restricciones no lineales (expresiones  c(x) <= 0)
73          F_optim[1] = J_costo
74          F_optim[2] = T
75         F_optim[3] = Ca
76         F_optim[4] = Cb
77         F_optim[5] = Cc
78
79  END COMPONENT
```

A.10.2. Experimento RTO

```
1
2  USE OPTIM_METHODS
3  FUNCTION INTEGER coste_y_restricciones(IN REAL esnopt_x[], IN INTEGER needF,
       OUT REAL esnopt_F[], IN INTEGER explicit_derivatives, IN INTEGER needG, OUT
        REAL esnopt_G[])
4
5  BODY
6
7      --RESET_VARIABLES ()
8
9          q  = esnopt_x[1]
10         qc = esnopt_x[2]
11
12
13     SET_INIT_ACTIVE(TRUE)
14     REL_ERROR = 1.0e-16
```

```
15      TIME  = 0
16      TSTOP = 0
17      CINT  = 0.1
18
19      INTEG()
20
21      esnopt_F[1] = F_optim[1]
22      esnopt_F[2] = F_optim[2]
23      esnopt_F[3] = F_optim[3]
24       esnopt_F[4] = F_optim[4]
25       esnopt_F[5] = F_optim[5]
26
27
28       --WRITE ("costeyrestriciones2 ffinal: %f %f %f %f %f\n", esnopt_F[1],
     esnopt_F[2], esnopt_F[3], esnopt_F[4], esnopt_F[5])
29
30      RETURN 0
31 END FUNCTION
32
33 EXPERIMENT RTO ON Reactor_RTO.partition1
34    DECLS
35         REAL numU = 2
36         REAL numC = 4
37         REAL param_estim[2]      -- variables de decision, size n_dec_var
38         REAL xlow[2]            -- valor inferior de las variables de decision,
       size n_dec_var
39         REAL xupp[2]            -- valor superior de las variables de decision,
       size n_dec_var
40         REAL Flow[5]            -- valor inferior de la funcion objetivo y las
      restricciones, size n_total
41         REAL Fupp[5]            -- valor superior de la funcion objetivo y las
      restricciones, size n_total
42         REAL CostoA = 0.5
43         REAL CostoB = 5
44         REAL CostoC = 0.3
45         REAL CostoD = 0.2
46         REAL CostoRef = 0.01
47
48    OBJECTS
49    INIT
50    BOUNDS
51    BODY
52
53         setValueReal( "c_A", CostoA)
54         setValueReal( "c_B", CostoB)
55         setValueReal( "c_C", CostoC)
56         setValueReal( "c_D", CostoD)
57         setValueReal( "c_ref", CostoRef)
58
59      --  Formar vector de variables de decision
60         param_estim[1] = q
61         param_estim[2] = qc
62
63
64         --  Limites de las variables de decision
65         xlow[1] = Liminfq
66         xupp[1] = Limsupq
67
68         xlow[2] = Liminfqc
69         xupp[2] = Limsupqc
70
71         -- Restricciones
72         Flow[1] = -1e50
```

```
73          Fupp[1] =  1e50
74
75          Flow[2] = 0
76          Fupp[2] = 60
77
78          Flow[3] = 0
79          Fupp[3] = 5
80
81          Flow[4] = 0
82          Fupp[4] = 5
83
84          Flow[5] = LiminfCa
85          Fupp[5] = LimsupCa
86
87          --Optimization extern routine call
88          setSilentMode(TRUE)
89          SET_REPORT_ACTIVE("#MONITOR",FALSE)
90
91          esnopt_init(numU, numC)
92          esnopt_set_variables_bounds_and_initial_values(xlow, xupp, param_estim)
93          esnopt_set_constraints_bounds_and_initial_values(Flow, Fupp, F_optim)
94          esnopt_set_cost_function_and_constraints(coste_y_restricciones)
95          esnopt_set_explicit_derivatives(0)
96          esnopt_set_function_precision(1.0e-15)
97          esnopt_set_iterations_limit(1000)
98          esnopt_print_data()
99          esnopt()
100         REL_ERROR = 1.0e-16
101
102         RESET_VARIABLES ()
103
104         esnopt_get_variables_values(param_estim)
105         esnopt_free()
106
107         setSilentMode(FALSE)
108         SET_REPORT_ACTIVE("#MONITOR",TRUE)
109
110
111         q = param_estim[1]
112         qc = param_estim[2]
113
114
115         WRITE("\n\n\n\n\Caudal de reactivos: %f ", q)
116         WRITE("\nCaudal de refrigerante: %f ", qc)
117
118         SET_INIT_ACTIVE(TRUE)
119         TIME = 0
120         TSTOP =0
121         CINT = 0.1
122         INTEG()
123
124 END EXPERIMENT
```