XXXVIII Jornadas de Automática
https://doi.org/10.17979/spudc.9788497497749.0563

APROXIMACIÓN DE MODELOS ALGEBRAICOS MEDIANTE ALAMO Y ECOSIMPRO

Carlos G. Palacín, José Luis Pitarch, Gloria Gutiérrez, César de Prada

Dpto. Ingeniería de Sistemas y Automática, Escuela de Ingenierías Industriales, Universidad de Valladolid. C/Real de Burgos s/n, 47011, Valladolid. {carlos.gomez, jose.pitarch, gloria, prada}@autom.uva.es

Resumen

El diseño de modelos apropiados para la simulación es, normalmente, un proceso largo y tedioso; por ello, no es conveniente utilizar el mismo modelo detallado para control y optimización. En la mayoría de los casos, los modelos necesarios para la optimización pueden ser más sencillos siempre y cuando reproduzcan las características principales de los modelos complejos. En este trabajo se ha diseñado un sistema para obtener modelos algebraicos sencillos a partir de un modelado más complejo desarrollado para la simulación.

***Palabras clave**: simulación, optimización, EcosimPro, ALAMO, modelo subrogado.*

1 INTRODUCCIÓN

En los procesos de simulación actuales se ha estimado que los tiempos dedicados a la codificación del sistema son cuatro veces mayores que los dedicados al modelado en sí. Esto es debido a la falta de especialización de los ingenieros de procesos, que han sido preparados en otros ámbitos [1].

Para ello, se han desarrollado nuevas herramientas de modelado y simulación, que ayudan en la codificación del sistema, incorporando sus propios lenguajes de programación de un nivel mucho mayor que los lenguajes clásicos. Dichas herramientas de modelado y simulación (e.g., EcosimPro, Dymola [2]) se han diseñado buscando un entorno que facilite la creación de modelos estacionarios y dinámicos de manera rápida e intuitiva. Sin embargo, carecen de buenas herramientas de optimización y, en la mayoría de los casos, hay que hacer uso de software externo. Esto conlleva un gran trastorno para el usuario que puede tomar dos caminos: puede volver a generar el mismo modelo en un entorno diferente, que puede no ser posible exactamente por peculiaridades de cada lenguaje; o bien puede crear un nuevo modelo personalizado, que facilita las tareas de codificación, pero añade una nueva etapa necesaria de validación sobre el sistema real.

Hay que tener en cuenta que los modelos apropiados para simulación deben representar fielmente los comportamientos dinámicos de los sistemas, a fin de poder servir como banco de pruebas, y comprender mejor el proceso que se está llevando a cabo.

Por otro lado, los modelos usados en optimización pueden ser juzgados de una manera más laxa en cuanto a similitud con el sistema real, puesto que no es necesario representar todo el proceso, sino únicamente las variables importantes para el estudio, o bien, sólo el comportamiento en estacionario frente a la dinámica total, si el tiempo del transitorio del sistema es muy breve en comparación con la duración del proceso global. Por otro lado, si se busca obtener herramientas de optimización en tiempo real, la ejecución rápida del código debe primar sobre la exactitud en el cálculo.

El resto del artículo se distribuye de la siguiente manera, en la siguiente sección se indican las herramientas externas utilizadas en detalle. En la sección 3 se describe la interfaz diseñada, y en la sección 4 se amplía el diseño inicial incluyendo paralelismo. En la sección 5 se presentan dos casos prácticos de uso. Finalmente, se finaliza el artículo con las conclusiones y posibles líneas de mejora.

2 SOFTWARE EMPLEADO

En este trabajo se han empleado como herramientas software concretas el entorno de modelado y simulación **EcosimPro**, de Empresario Agrupados, y la herramienta de estimación de modelos **ALAMO** [3], desarrollada en la universidad *Carnegie Mellon*.

2.1 EcosimPro

EcosimPro es una potente herramienta de simulación y modelado de sistemas continuos y discretos. Posee su propio lenguaje de programación de alto nivel basado en el paradigma de la orientación a objetos, EL (**E**cosimPro **L**anguage). Además, permite realizar dicha programación tanto en un modo textual como en un modo gráfico. Por otro lado, gracias a la

orientación a objetos, es posible diseñar sistemas complejos usando código programado anteriormente tanto en forma de componentes de un sistema mayor, como en forma de clases con métodos y variables propias. EcosimPro permite su instalación tanto en sistemas operativos Windows como Linux. Reduciendo, o incluso eliminando, las limitaciones para compartir librerías desarrolladas en distintos equipos o en distintas instalaciones. Esto permite además, disponer de equipos de trabajo especializados en el modelado de librerías de bajo nivel, y otros equipos que desarrollan sistemas más complejos que hagan uso o demanden estas.

2.2 ALAMO

Por su parte, ALAMO es un software que busca identificar ecuaciones algebraicas sencillas que relacionen las entradas y las salidas de sistemas de caja negra. Para ello plantea un problema mixto-entero que debe minimizar el error del valor devuelto por la función propuesta frente al valor real de la variable. La ecuación solución consiste en una combinación de funciones algebraicas base.

Para la obtención de la combinación óptima, ALAMO hace uso del software externo GAMS, para la resolución de los problemas de programación mixta-entera lineal y cuadrática planteados. El algoritmo de resolución que se emplea se puede seleccionar, siendo el valor predefinido y aconsejado el algoritmo BARON. En caso de que GAMS no se encuentre disponible en el sistema donde se ejecuta, se utilizarán enfoques enumerativos para la optimización combinatoria. [4], que pueden alargar el tiempo de resolución en problemas de gran escala, y hacer que el modelo subrogado obtenido no sea preciso y/o fiable.

La interfaz de ALAMO es un punto débil importante del programa. Las llamadas tienen que hacerse mediante línea de comandos, mientras que la comunicación de datos de entrada y salida se realiza mediante ficheros de texto plano. Por ejemplo, en el fichero de entrada se debe especificar el número de variables de entrada que tiene el sistema, y el número de salidas que se deben estimar. Además se pueden indicar los nombres de las variables de entrada del sistema y de las variables a calcular. En caso de no indicar los nombres, el programa por defecto etiquetará con una letra “x” seguida de un índice numérico las entradas y con una letra “z” seguida, igualmente, de un índice numérico las salidas.

En el fichero de entrada se deben introducir también las funciones base elegidas, con las que el programa debe intentar ajustar la curva de salida. Una vez ejecutado, ALAMO presenta varias soluciones que disminuyen el error pero incrementan la dificultad añadiendo más funciones o aumentando la complejidad de éstas.

ALAMO está preparado para el trabajo con datos obtenidos de planta, que deben introducirse en el fichero de entrada en columnas ordenadas donde las primeras columnas serán los valores de las entradas y las siguientes los de las salidas, en el mismo orden que se han indicado los nombres de estas. Sin embargo, también se permite el trabajo con simuladores, que deben responder a una sintaxis muy concreta en la llamada.

3 DISEÑO DE LA INTERFAZ

En esta sección se definirá la interfaz implementada que permite realizar la llamada desde ALAMO empleando los modelos codificados en EcosimPro. Inicialmente se debe desarrollar el modelo completo en EcosimPro, que puede ser tanto dinámico como estacionario, a este modelo se le denomina componente en lenguaje EL. Una vez que se tiene el componente, se definen las llamadas particiones, que son la traducción a lenguaje matemático y conllevan también la obtención de código en el lenguaje C++.

Todas las particiones heredan de una clase padre, por lo que tienen la misma interfaz. Esta ofrece funciones que permiten escribir y leer el valor de variables. También incluye llamadas para indicar tiempos de simulaciones, o el cálculo del estacionario. En nuestro caso se hará uso de este último. Esta implementación de la herencia nos permite minimizar los cambios que se deben realizar en el código por el usuario al mínimo, utilizando punteros a la clase padre e inicializando en cada caso a la partición concreta, como se indica en la Figura 1.

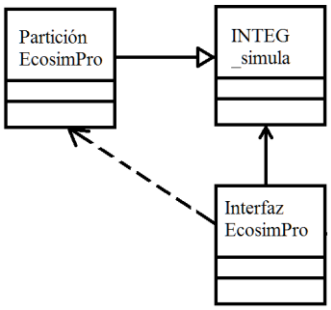

Figura 1. Implementación llamada EcosimPro

Con las particiones iniciadas al modelo concreto, se compila el ejecutable que se ha definido con la interfaz que necesita ALAMO. Este tiene que leer de un fichero de texto que el programa de estimación genera, donde indica una serie de valores que las variables de entrada deben tomar, y genera un fichero de salida con la respuesta del sistema.

Para facilitar el uso por el ingeniero de procesos, se han desarrollado una serie de funciones que permiten la creación del fichero de texto plano de configuración que requiere ALAMO como entrada, desde un experimento de EcosimPro. Más tarde, este fichero de configuración se utilizará para generar el simulador de manera automática y no tener que realizar la introducción de las variables de manera manual en distintos puntos, lo que disminuye la posibilidad de fallos humanos. Del mismo modo, se han implementado funciones de llamada para permitir la ejecución del estimador desde el mismo programa EcosimPro, a fin de poder observar las funciones respuesta sugeridas por el estimador.

4 SIMULACIÓN CONCURRENTE

Gracias a las arquitecturas actuales de los procesadores multinúcleo, que incluso pueden llegar a incorporar varios núcleos lógicos dentro de un mismo núcleo físico, se puede conseguir la ejecución concurrente de procesos. Existen diversos enfoques en la programación paralela: siendo los más usados las llamadas a las instrucciones del sistema operativo; y la utilización de directivas especiales ofrecidas por los compiladores actuales.

Históricamente, la paralelización se basaba en la ejecución multiproceso [5]. En esta versión, es necesaria la existencia de varios núcleos para tener concurrencia real. Cada proceso dispone de su propio espacio en memoria para el almacenamiento del código y de los valores de las variables. Existe un proceso padre que crea tantos procesos hijos como desee, siendo el número máximo adecuado el número de núcleos menos uno. Todos comparten el mismo código, variando en su número de identificación de proceso. Al no tener memoria en común, es importante indicar previamente las zonas de memoria que se deben compartir para el paso de variables, o el acceso a la misma.

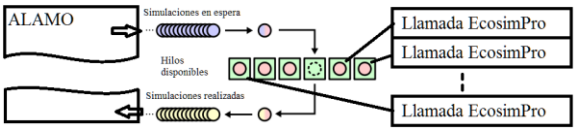

Figura 2. Implementación de la concurrencia

Una versión más actual es la implementación multihilo [6], donde un proceso padre genera tantos hilos como necesite. En este caso todos los hilos comparten la memoria, haciendo más sencillo el paso de parámetros, pero también necesario evitar que varios hilos accedan a la misma variable a la vez. Esto implica la necesidad de implementar algoritmos de reserva de recursos y prioridades, evitando que un dato pueda sufrir accesos concurrentes.

Actualmente se está desarrollando una nueva vertiente denominada paralelización por tareas [7], donde no se impone el código que debe ejecutar cada hilo, sino que se indican las tareas que se pueden realizar de manera concurrente y el número de hilos para ejecutarlas. El sistema repartirá tantas tareas pendientes de la lista como hilos creados haya entre los núcleos lógicos del sistema, cuando un hilo finaliza su ejecución, y en caso de que la pila de tareas pendientes aún no haya finalizado, recibirá otra tarea. De esta manera se pueden conseguir implementar programas paralelos con mayor eficiencia con independencia del número de núcleos disponible en el equipo donde se ejecute, puesto que el número de tareas a distribuir siempre será el mismo, y el de hilos puede llegar a ser máximo admitido por el equipo. Para poder ejecutar este paradigma de programación paralela en el lenguaje C++ es necesario utilizar herramientas externas que implementen a bajo nivel el reparto de los núcleos de los procesadores entre los hilos creados, y asignen las tareas a estos. Dos posibles opciones son la: librería Threading Building Blocks (TBB) desarrollada por Intel [8]; o la biblioteca Qt [9], que ofrece clases especiales, como *QThreadPool*, para codificar dicha implementación.

En el caso de la interfaz presentada, el programa de estimación de modelos da una serie de combinaciones de las variables de entrada y espera recibir sendos valores de las salidas del sistema estimadas. Por lo tanto, es necesario lanzar la simulación ese mismo número de veces. Debido a que cada simulación es independiente de las demás, todas podrían ejecutarse de manera concurrente. Para implementar esta paralelización se ha definido como tarea la ejecución de la simulación, y se ha utilizado el paradigma de la paralelización de tareas. Un pequeño diagrama se puede ver en la Figura 2.

5 RESULTADOS

El enfoque integrado propuesto se ha utilizado en dos ejemplos prácticos para comprobar su eficiencia: inicialmente un modelo de un reactor químico, usado principalmente como banco de pruebas; y posteriormente sobre un modelo más complejo de una planta de evaporación industrial real. La respuesta seleccionada no es en todos los casos la que minimiza el error con la respuesta del sistema[1], en algunos casos se ha balanceado la precisión con complejidad del modelo subrogado.

[1] Aumentar la precisión implica aumentar la complejidad y/o el número de las funciones base seleccionadas por ALAMO, pudiendo reducir la robustez del modelo.

5.1 REACTOR CONTINUO

Se dispone de un reactor químico de flujo continuo, donde un reactivo A se convierte en un producto B mediante una reacción exotérmica. Para mantener el reactor en un estado estable es necesario, por lo tanto, enfriarlo, lo cual se consigue mediante una camisa refrigerante.

5.1.1 MODELO DEL REACTOR

A continuación se presentan las ecuaciones diferenciales del modelo dinámico de simulación del reactor.

$$V \cdot \frac{dC_A}{dt} = q \cdot (C_{A0} - C_A) - V \cdot k \cdot C_A \quad (1)$$

$$V \cdot \rho \cdot C_P \cdot \frac{dT}{dt} = q \cdot \rho \cdot (T_e - T) - V \cdot k \cdot C_A \cdot \Delta H - Q \quad (2)$$

$$V_r \cdot \rho_r \cdot C_{Pr} \cdot \frac{dT_r}{dt} = F_r \cdot \rho_r \cdot C_{Pr} \cdot (T_{re} - T_r) + Q \quad (3)$$

$$k = 5{,}967 \cdot e^{\frac{-826}{T+273{,}15}} \quad (4)$$

$$Q = U \cdot A \cdot (T - T_r) \quad (5)$$

$$x = \frac{C_B}{C_{A0}} \equiv \frac{C_{A0} - C_A}{C_{A0}} \quad (6)$$

Donde:

V: Volumen del reactor (m^3)
V_r: Volumen de la camisa refrigerante (m^3)
C_A: Concentración de A en el reactor (kg/m^3)
C_{A0}: Concentración de A en la alimentación (kg/m^3)
q: Flujo volumétrico de alimentación (m^3/h)
F_r: Flujo volumétrico del refrigerante (m^3/h)
k: Constante cinética de la reacción (1/h)
ρ: Densidad corriente de alimentación (kg/m^3)
ρ_r: Densidad del refrigerante (kg/m^3)
C_P: Capacidad calorífica de la mezcla (kJ/kg K)
C_{Pr}: Capacidad calorífica del refrigerante (kJ/kg K)
T: Temperatura del reactor (ºC)
T_e: Temperatura de entrada de la alimentación (ºC)
T_r: Temperatura media del refrigerante (ºC)
T_{re}: Temperatura de entrada del refrigerante (ºC)
ΔH: Calor de reacción (kJ/kg)
Q: Calor intercambiado (kJ/h)
U: Coef. global de transmisión de calor (kJ/hm^2K)
A: Superf. de transmisión de calor de la camisa (m^2)
x: Ratio de conversión de A en B (adimensional)

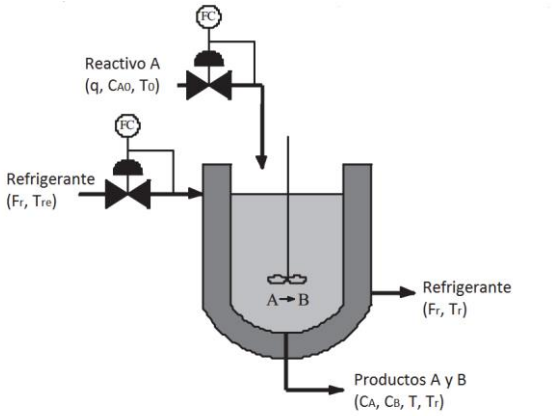

Figura 3. Esquema del reactor

5.1.2 MODELO ESTIMADO

Las variables controlables son el caudal de entrada del reactivo y el flujo del refrigerante, por lo que serán las variables de entrada que se deben indicar a ALAMO. Por otro lado, se desea maximizar la concentración del producto a la salida, manteniendo la temperatura dentro de los límites permitidos. Por consiguiente, las funciones a estimar son el caudal del producto a la salida, y la temperatura en el interior del reactor. Esta solución sirve para buscar el punto de trabajo en el estacionario, dejando de lado el control de la dinámica del sistema que se supone existente. Las respuestas obtenidas se muestran junto a los valores de la simulación en la Figura 4 y la Figura 5, donde en el eje de coordenadas se puede ver el valor de la variable en sí, y el eje de abscisas representa combinaciones de las variables de entrada.

$$C_B = -1{,}2 \cdot q + 0{,}21 \cdot q^2 - 0{,}15 \cdot 10^{-1} \cdot q^3 + 0{,}18 \cdot 10^{-6} \cdot F_r^{-3} - 0.43 \cdot 10^{-2} \cdot q \cdot Fr + 0{,}23 \cdot 10^{-5} \cdot (q \cdot Fr)^2 + 14 \quad (7)$$

$$T = 26 \cdot q + 1{,}9 \cdot \ln(q) - 21 \cdot \ln(Fr) - 0{,}97 \cdot q^2 + 0{,}23 \cdot 10^{-2} \cdot F_r^2 - 0{,}18 \cdot q \cdot F_r + 0{,}84 \cdot 10^{-4} \cdot (q \cdot F_r)^2 + 100 \quad (8)$$

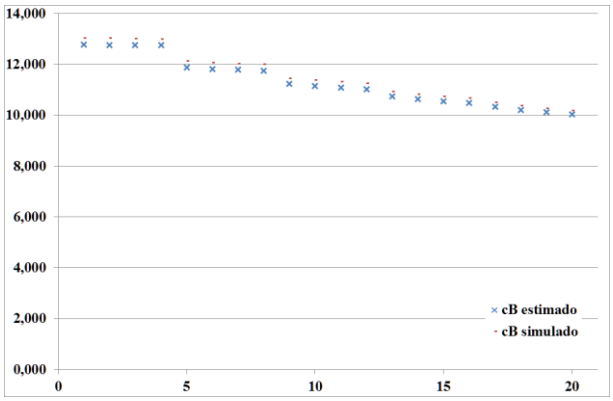

Figura 4. Caudal de producto a la salida

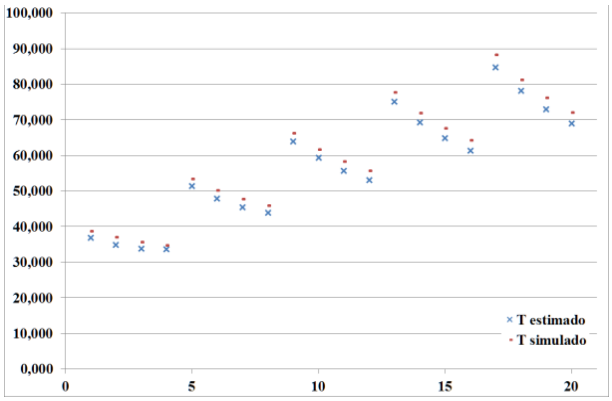

Figura 5. Temperatura del reactor

5.2 PLANTA DE EVAPORACIÓN

El sistema a estimar consiste en una planta de evaporación industrial. El sistema debe evaporar una cantidad fija de agua de una disolución de ácidos y sales, a fin de mejorar la cristalización posterior de dicha disolución. El sistema consta de una serie de intercambiadores de calor iniciales para incrementar la temperatura del líquido y, después, una combinación de cámaras de evaporación. Un pequeño diagrama del sistema se puede ver en la Figura 6. Para calentar el producto se emplea inicialmente el vapor obtenido en las cámaras de evaporación 1 y, en los últimos intercambiadores, vapor vivo, cuyo consumo se desea reducir a fin de economizar el proceso.

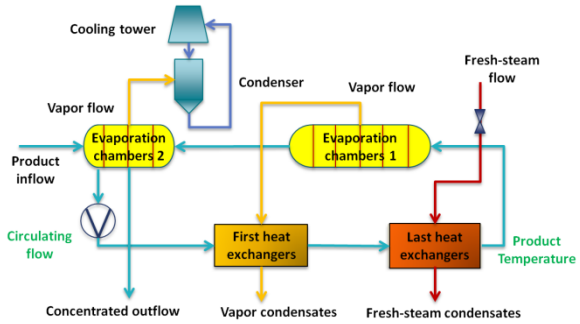

Figura 6. Diagrama de la línea de evaporación

Las variables controladas son la temperatura del producto a la salida de los intercambiadores y el caudal de recirculación. Por lo tanto, dichas variables serán los parámetros de entrada que se indicarán al estimador. Puesto que se desea reducir el consumo de vapor específico, éste, junto con el caudal de agua evaporada, serán las variables que se deben estimar. Igualmente, la solución obtenida sirve para el sistema estacionario, no prestando atención a la dinámica del sistema, mucho más rápida. La comparación entre los valores devueltos por la simulación y los estimados por el modelo subrogado se muestran en la Figura 7 y la Figura 8, donde nuevamente el eje de coordenadas representa el valor de la variable y en el eje de abscisas se indican las combinaciones de entradas.

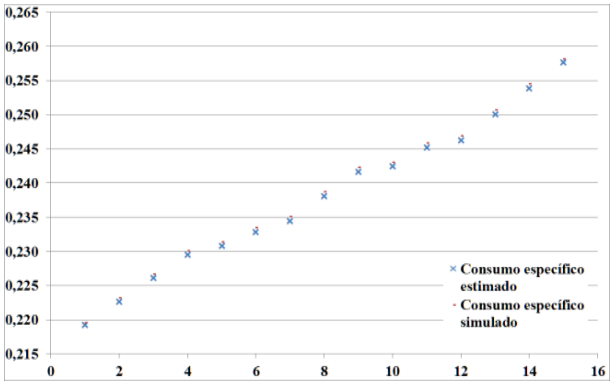

Figura 7. Consumo específico de vapor

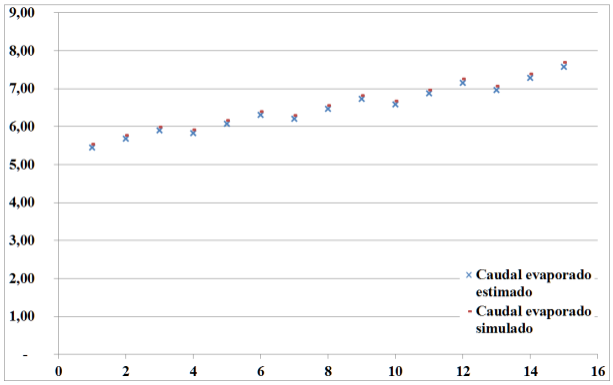

Figura 8. Caudal de agua evaporada

Las ecuaciones del modelo completo han sido omitidas por brevedad. Se emplaza al lector que así lo desee a consultar el modelo completo en [10].

5.2.1 MODELO ESTIMADO

$$SSC = 0{,}82 \cdot 10^{-3} \cdot R_F + 0{,}74 \cdot 10^{-2} \cdot \ln(O_T) + 0{,}38 \cdot 10^{-5} \cdot O_T \cdot R_F \quad (9)$$

$$E_F = 0{,}34 \cdot 10^{-6} \cdot O_T^3 - 0{,}17 \cdot 10^{-6} \cdot R_F^3 + 0{,}41 \cdot 10^{-3} \cdot O_T \cdot R_F \quad (10)$$

Donde:

SSC: Consumo específico de vapor (adimensional)
R_F: Caudal de recirculación (kg/s)
O_T: Temperatura de salida de los intercambiadores (ºC)
E_F: Caudal de evaporación (kg/s)

6 CONCLUSIONES Y LINEAS DE TRABAJO FUTURAS

Se ha desarrollado una interfaz para la obtención de modelos simplificados adecuados para la optimización partiendo de modelos complejos. Estos modelos se han generado previamente con un enfoque en la simulación de procesos. Además, se ha comprobado su eficacia en varios sistemas.

Se han mejorado los tiempos de cómputo necesarios inicialmente incluyendo una implementación que hace uso de técnicas de paralelismo.

Aunque el sistema es relativamente sencillo de editar, se pretende incorporar una interfaz gráfica que facilite aún más dicha tarea. De esta manera, el operario podrá usar el sistema sin necesidad de utilizar otro programa que el simulador.

También se está trabajando en una versión que permita obtener modelos subrogados de las dinámicas del sistema, lo que implica añadir el tiempo como variable de entrada y el tratamiento de un número mayor de datos y simulaciones. Esta versión del programa permitirá obtener modelos útiles para optimizaciones dinámicas.

Por otro lado, aunque la simulación es el mayor consumidor de tiempo en la ejecución del programa, la lectura de ficheros, también supone un cuello de botella, por lo que la paralelización de tareas se podría aumentar a la lectura del mismo, puesto que las primeras simulaciones no deben esperar a que el fichero se haya terminado de leer para empezar a ejecutarse. Para esto habría que añadir restricciones en el acceso a variables mediante la implementación de semáforos de acceso a un vector con los valores

Agradecimientos

Este trabajo ha sido financiado por el programa de innovación y desarrollo de la Unión Europea Horizonte 2020, proyecto CoPro (contrato nº123575) y por el Gobierno de España con fondos MINECO /FEDER (DPI2015-70975-P).

Referencias

[1] *EcosimPro. Dynamic Modeling & Simulation Tool*. EA International. http://www.ecosimpro.com

[2] Cellier, F. E. "Dymola: Environment for object-oriented modeling of physical systems." Lund, Sweden: Dassault Systemes AB 2015.

[3] Cozad, Alison, N. V. Sahinidis, and David C. Miller. "Alamo: Automatic learning of algebraic models for optimization." American Institute of Chemical Engineers, 2013 Annual Meeting, San Francisco. 2013.

[4] Ibaraki, Toshihide. Enumerative approaches to combinatorial optimization. Baltzer, 1987.

[5] Almasi, George S., and Allan Gottlieb. "Highly parallel computing." 1988.

[6] Alverson, G. A., Charles, D. C. I., Kahan, S. H., Koblenz, B. D., Porterfield, A., & Smith, B. J. "Synchronization Techniques in a Multithreaded Environment." U.S. Patent No. 6,862,635. 1 Mar, 2005.

[7] Yang, Gang, Xingshe Zhou, and Huifang Pan. "Feedback-based Framework of Adaptive Threads-pool Management." Computer Engineering 21, 2006: 65-67.

[8] Reinders, James. Intel threading building blocks: outfitting C++ for multi-core processor parallelism. " O'Reilly Media, Inc.", 2007.

[9] Summerfield, Mark. Advanced Qt Programming: Creating Great Software with C++ and Qt 4. Pearson Education, 2010.

[10] Pitarch, J. L., Palacín, C. G., De Prada, C., Voglauer, B., & Seyfriedsberger, G. "Optimisation of the resource efficiency in an industrial evaporation system." Journal of Process Control 56, 2017: 1-12.