OPTIMIZACIÓN EN TIEMPO REAL DE SISTEMAS ESTACIONARIOS EN ECOSIMPRO

Carlos Gómez Palacín, José Luis Pitarch, 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, prada}@autom.uva.es

Resumen

La optimización de procesos es, en general, un problema no convexo que puede resolverse iterativamente, utilizando información sobre los gradientes de las restricciones y función objetivo. Para obtener dicha información de forma eficiente en un sistema estacionario, se requiere el uso de herramientas de diferenciación automática (DA). En este trabajo se ha programado una interfaz de comunicación entre EcosimPro® (software de modelado y simulación) y CppAD (paquete para DA de algoritmos C++) junto con IPOPT (software para resolver problemas de optimización). La eficacia de la herramienta propuesta ha sido probada experimentalmente en una línea de evaporación industrial.

Palabras clave: modelos estacionarios, CppAD, EcosimPro, optimización, NLP, diferenciación automática.

1 INTRODUCCIÓN

En los últimos años, el uso eficiente de los recursos se ha consolidado como una de las principales metas en el entorno industrial. En este sentido, la optimización de procesos en tiempo real es prácticamente obligatoria: no sólo se consigue una rebaja en el coste de producción, sino también disminuir el impacto que se refleja en el medio ambiente. Para facilitar el desarrollo en esta dirección, se ha avanzado en el diseño de indicadores específicos sobre la utilización de los recursos. Son los llamados *REI*s (del inglés *resource efficiency indicator),* que se centran en la visualización sencilla y agregación de la información entre aspectos de producción y energía empleada [1], en lugar de medir parámetros independientes. En el desarrollo del empleo de los REIs, hay una amplia variedad de problemas de optimización basada en modelos que pueden ser abordados de forma eficiente mediante software de programación no lineal (NLP).

El cálculo eficiente de Jacobianos y Hessianos es, hoy en día, una necesidad a la hora de resolver estos problemas de optimización no lineal mediante algoritmos de resolución iterativos basados en gradiente [2], [3]. Cuanta mayor exactitud se consiga al realizar este cómputo, mejor información se le proporciona al optimizador y, por tanto, se obtendrán soluciones más precisas en menor número de iteraciones.

Existen distintas opciones para realizar el cálculo de las derivadas de manera automática. La más extendida es la aproximación mediante diferencias finitas, que ofrece un cálculo sencillo y sistemático, por lo que la codificación del problema es transparente. Sin embargo, lógicamente, el cómputo no es tan exacto como sería deseable. Una opción más eficaz es el cálculo simbólico. Esta posibilidad nos devuelve resultados exactos, a cambio de aumentar el coste computacional. Este puede parecer un problema menor dada la evolución de los equipos actuales. Sin embargo, puede convertirse en un cuello de botella en aplicaciones industriales con elevadas restricciones de cómputo en tiempo real.

La tendencia actual es utilizar soluciones que implementen algoritmos de *diferenciación automática* (DA) [4]. Dichas herramientas se sirven de la conocida regla de la cadena para codificar automáticamente en lenguaje máquina los cálculos a realizar para obtener los valores de todas las derivadas necesarias. Así, se pueden obtener resultados precisos con tiempos de cálculo muy inferiores a la opción simbólica, aunque la cantidad de memoria requerida para el almacenamiento de valores para todas las variables intermedias puede ser elevado [5].

Recientemente se han desarrollado buenas herramientas de modelado y simulación [6], [7], que ya permiten también abordar problemas de optimización dinámica (ODEs y/o DAEs) siguiendo un enfoque secuencial [8]. Sin embargo, éstas fallan en la resolución con modelos puramente estacionarios o discreti-

zados (lazos algebraicos), ya que los algoritmos de integración numérica sólo ofrecen información sobre las sensibilidades (utilizada posteriormente para el cálculo de derivadas) en sistemas con ecuaciones diferenciales. En estos casos se requiere el uso de herramientas que proporcionen DA [4], [9], [10]. Por desgracia, la falta de integración entre éstas y los softwares de modelado/simulación puede llegar a hacer muy tediosa la resolución del problema completo en procesos complejos. En efecto, las diferentes estructuras y lenguajes de programación empleados por las herramientas existentes de cálculo computacional obligan a que el usuario tenga que redefinir, casi por completo, el problema de optimización ante cualquier cambio en proceso y/o función objetivo. Esta problemática ya ha sido abordada recientemente, de forma preliminar, para la resolución eficiente de problemas de optimización dinámica mediante enfoques secuencial y simultáneo [11].

En este trabajo se aborda dicho problema (modelado + optimización) para sistemas estacionarios, proponiendo una interfaz de comunicación bilateral integrada entre la herramienta de modelado EcosimPro® y un entorno de optimización NLP que implementa DA, como es CppAD. En la siguiente sección se explica la programación de la interfaz de comunicación entre ambas herramientas, en la Sección 3 presenta la utilidad del trabajo desarrollado para optimizar un sistema de evaporación industrial, la Sección 0 muestra los resultados obtenidos en dicho proceso y, finalmente, se exponen las conclusiones generales a las que se ha llegado.

2 DISEÑO DE LA INTERFAZ

La idea de esta sección es crear una interfaz general para la optimización de problemas estacionarios. De esta forma, se puede realizar la codificación de distintos problemas de optimización para el mismo modelo de manera independiente.

Como segundo objetivo, se busca además que los resultados de dichas optimizaciones sean accesibles desde el software de modelado y simulación, para poder aplicarse a un modelo dinámico más complejo y detallado, y analizar así el comportamiento antes de implementar el control en el proceso real. Para ello, la compilación del código de optimización se realizará creando bibliotecas dinámicas, de forma que puedan ser ejecutados desde programas externos.

Utilizando el potencial de la herencia que ofrece el paradigma de la programación orientada a objetos, la llamada a las funciones de paso de datos y la obtención de resultados se implementa como una interfaz similar para cualquier modelo.

2.1 SOFTWARE EMPLEADO

EcosimPro® es de una herramienta de simulación y modelado orientado a objetos de gran potencia. Permite la programación en diferentes lenguajes (desde un lenguaje gráfico a texto estructurado de alto nivel) sin necesidad de cumplir relaciones de causalidad en el orden de aparición de las ecuaciones del modelo en el código del programa [12].

Una vez que el modelo se ha generado, EcosimPro® lo reduce internamente a un modelo matemático causal. Dicho modelo queda exportado a un fichero bajo código *html*. Con ello, se dispone de una visualización a través de un explorador web de todo el modelo perfectamente ordenado: se muestran los datos numéricos, las variables necesarias con el tipo de tratamiento necesario en el problema, y las ecuaciones ordenadas por causalidad. Por tanto, a través de este fichero se puede realizar una traducción del modelo a otros lenguajes, en muchos casos automatizable.

EcosimPro® también permite el uso de funciones y clases externas. Por lo tanto, se podrá ejecutar un código de optimización en cualquier instante de la simulación, tomando como entradas datos provenientes de ésta y modificando las variables de decisión con los resultados de aquella. De esta forma se emula el procedimiento que se realizaría en una planta industrial, tomando las medidas de los distintos sensores y actuando sobre las acciones de control.

Como software de optimización se utilizará un algoritmo NLP de punto interior como es IPOPT [3], junto con CppAD [4]. Este proyecto, auspiciado por la organización Coin-Or, ofrece una interfaz en lenguaje de programación C++ para codificar nuestras ecuaciones del modelo y realizar diferenciación automática. Esta herramienta se basa en el empleo de un tipo de datos propio durante la codificación, necesario para el cálculo de las derivadas de forma automática.

CppAD permite trabajar con matrices de funciones y variables, por lo que también facilita el cálculo de Jacobianos y Hessianos. Además, permite seleccionar el sentido para recorrer el árbol que se generará por diferenciación automática (sentido directo o inverso),

según convenga en cada problema. Dependiendo del número de variables y el número de funciones, un camino será más eficiente que otro: en sentido positivo el cálculo de las derivadas se encuentra acotado sobre tres veces el número de variables, mientras que con el cálculo en sentido inverso, se relaciona con realizar tres veces el cálculo de las funciones, sin depender del número de variables. Así, esta herramienta supera a otros desarrollos similares en el tratamiento de problemas de optimización pequeños, pero con un gran número de variables.

Como punto negativo, CppAD no es apropiado para tratar modelos dinámicos (no incorpora interfaz con integradores numéricos), por lo que todo el trabajo se debe realizar en estado estacionario.

2.2 EXTRACCIÓN DEL MODELO

Como etapa inicial, se ha diseñado una clase abstracta que será similar en todos los problemas de optimización. Esta clase variará internamente entre modelos, ya que cada una tendrá las ecuaciones propias, pero la interfaz con el exterior será igual, disponiendo de funciones para fijar parámetros del modelo, y límites y valores para las variables y las restricciones. Además, ofrece la comunicación para la obtención de los estados, el valor de la función de coste final, las restricciones, y las variables.

Para mayor comodidad, todas las variables internas, tanto parámetros como variables de decisión, son instanciadas mediante cadenas de texto. E inclusive, se pueden etiquetar las restricciones.

La creación de esta clase básica de simulación se realiza de manera automática mediante un ejecutable, compilado sobre una arquitectura de 64 bits. Actualmente no dispone de una interfaz gráfica, sino que debe lanzarse sobre la línea de comandos del sistema operativo Windows®, pasando como argumento el nombre del fichero *html* de la partición generada por EcosimPro®.

En la traducción, las variables definidas en la partición como *ALGEBRAIC*, *BOUNDARY* o *DYNAMIC*, se convierten en variables de decisión del optimizador. Las demás, se convierten en parámetros y variables internas dentro del modelo. Además, fijando las variables de tipo *DERIVATIVE* a cero, conseguimos convertir cualquier modelo dinámico a uno estacionario.

Las funciones auxiliares del modelo se reproducen en una función de llamada. Se traducen desde el lenguaje de EcosimPro® a lenguaje C++, lo que requiere modificar algún elemento, como las potencias o las secuencias condicionales. Las llamadas a funciones definidas en otro código, ya sea de la misma o de otra biblioteca dinámica, se mantienen con el mismo nombre de función. Debido a que no están en el mismo fichero, éstas sí deben ser traducidas manualmente y añadirse bien en el mismo fichero del modelo, o bien en un archivo diferente e incluir una llamada a éste.

2.3 INTEGRACIÓN DEL OPTIMIZADOR

En este caso se emplea como optimizador IPOPT. Este posee una interfaz para codificar los problemas en lenguaje C++. Para ello hay que realizar la implementación de una clase abstracta ya definida, y que sirve para devolver los valores de las restricciones y la función de coste al núcleo de optimización. En la Figura 1 se representa el esquema de herencia, donde una clase concreta calcula el Jacobiano y el Hessiano mediante diferenciación automática.

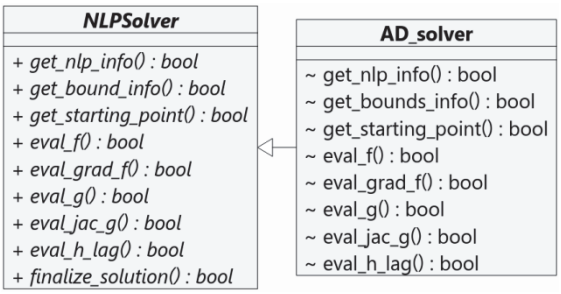

Figura 1. Comunicación con el optimizador.

Por lo tanto, una vez que se ha obtenido el modelo extrayéndolo desde el programa de simulación y se tiene la interfaz para la llamada al procedimiento de optimización, se debe generar una clase que aúne ambas para poder resolver cada problema concreto. Para ello se crea una clase que herede del modelo de simulación y que implemente las funciones de la optimización, i.e. la función de coste y las restricciones. Es decir, para el mismo modelo, se pueden crear distintos problemas de optimización teniendo que escribir solamente las funciones concretas de cada caso. Se puede ver un ejemplo en la Figura 2.

Una vez se tiene codificado el problema concreto, mediante el entorno de programación que se desee (en este caso se utilizó MSVC++©) se compila el código incluyendo las bibliotecas de optimización. Para poder usarse desde EcosimPro®, se debe crear

un archivo de código fuente donde se especifique la interfaz de la clase de modelo general, el archivo de cabecera de la misma, y la librería estática. De esta forma el programa conoce los nombres de las funciones externas y su configuración, y no presentará problemas en la compilación de los experimentos.

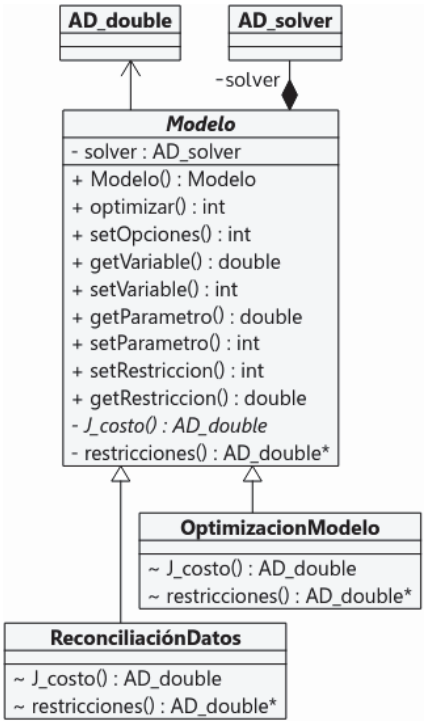

Figura 2. Interfaz de comunicación de la optimización.

Las variables internas, por defecto, están iniciadas a cero, por lo que antes de realizar la llamada al optimizador, es necesario configurar todos los valores de los parámetros. En el caso de las variables, se debe introducir el valor inicial, y los límites entre los que debe encontrarse.

Para introducir los límites de las restricciones se debe indicar la posición de las mismas y los límites entre los que debe permanecer acotada. Para mantener la filosofía del lenguaje de programación de EcosimPro®, se considera la primera restricción con índice 1, en lugar de índice 0 como sería en C++. Los límites de las restricciones también son, por definición, nulos. Es necesario inicializar todos para obtener un resultado correcto.

Una vez que todos los valores se han inicializado, se puede ejecutar la optimización mediante la llamada a la función correspondiente. Dependiendo del resultado de la optimización, este procedimiento nos devolverá un valor diferente, los principales casos se pueden ver en la Tabla 1.

Si el problema ha resultado factible, se pueden recoger los resultados de las variables de decisión en la simulación, así como el valor de las restricciones y la función de coste.

En caso de querer cambiar las opciones de la llamada al optimizador, como aumentar o reducir la cantidad de información imprimida por pantalla, se debe declarar una variable de tipo cadena. En ella se definirá cada opción que se desee modificar con tres elementos separados entre sí por espacios: el tipo de dato (*String*, *Integer* o *Numeric*), el nombre de la opción a configurar y el nuevo valor[1]. Se debe incluir un carácter de final de línea al terminar cada terno, e.gr., “Integer print_level 3\n”.

Tabla 1. Posibles valores devueltos por IPOPT

Valor	Significado
1	Éxito
2	Número máximo de iteraciones
3	Falta de progreso alrededor del óptimo
4	Solución con precisión aceptable
5	Problema no factible
12	Insuficientes grados de libertad

3 APLICACIÓN INDUSTRIAL

Los desarrollos de software presentados en la sección anterior se han aplicado en la práctica sobre un proceso de evaporación industrial de múltiple efecto, formado por: dos cámaras de evaporación y dos intercambiadores de calor en serie, un condensador barométrico, una torre de refrigeración y un saturador (Figura 3). El sistema recibe y recircula una mezcla líquida de agua con varios componentes químicos con el objetivo de concentrarla, evaporando una cierta cantidad de agua, para poder recuperar parcialmente dichos componentes mediante cristalización posterior.

3.1 MODELO MATEMÁTICO

Se ha realizado un modelo de caja gris que representa al proceso en estado estacionario. La parte principal del mismo está basada en leyes físicas:

[1] Para ver todas las opciones disponibles, consultar la web:

http://www.coin-or.org/Ipopt/documentation/node39.html

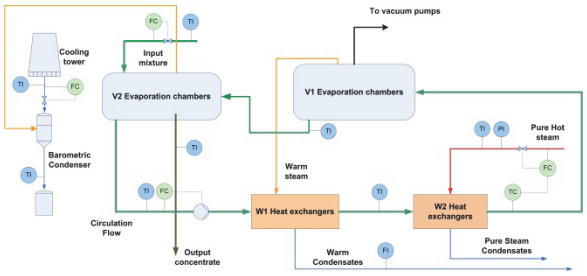

Figura 3. Esquema del proceso de evaporación e instrumentación asociada.

- Balances de masa y energía en los principales equipos.
- Relaciones de densidad entre caudales másicos y volumétricos de diferentes fluidos en función de temperatura y/o presión.
- Transmisión de calor entre fluidos en los intercambiadores: $Q = U \times \Delta_T$, donde Q es la potencia calorífica, U el coeficiente de transmisión de calor y la diferencia de temperatura media logarítmica Δ_T se ha calculado utilizando la aproximación de Chen [13].
- Equilibrios químicos en función de temperatura-presión-concentración de la mezcla en las cámaras de evaporación.

Además, debido a que los procesos reales son de orden elevado y es difícil conocer las relaciones entre algunas variables internas, las anteriores ecuaciones físicas se complementan con relaciones identificadas experimentalmente. Para ello se ha utilizado la técnica de *reconciliación de datos* [14] acorde al siguiente procedimiento:

1. Pretratamiento de medidas en bruto para excluir valores erróneos (fuera de rango o fallo de comunicaciones con el sensor).
2. Estimación de parámetros y variables del proceso resolviendo reconciliación de datos[2] con el modelo basado en primeros principios.
3. Identificación experimental de patrones.
4. Validación del modelo completo (leyes físicas + experimentales) mediante una nueva fase de reconciliación con nuevas medidas.

En total, el modelo no lineal propuesto consta de 46 ecuaciones de igualdad (donde aparecen 6 lazos algebraicos) y ha sido validado satisfactoriamente con medidas de 8 meses tomadas en planta.

[2] Se han utilizado estimadores robustos [15] como función objetivo para el optimizador, con el fin mitigar problemas de errores gruesos en las medidas.

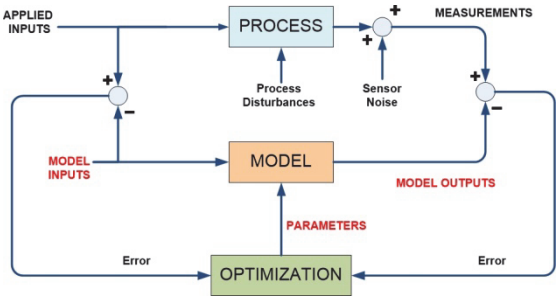

Figura 4. Esquema de reconciliación de datos.

3.2 OPTIMIZACIÓN DE LA EFICIENCIA

El objetivo aquí es llevar al proceso a un punto de operación donde se minimice el consumo de recursos (principalmente caudal de vapor sobrecalentado) mediante la selección de consignas de control óptimas. Para ello, se han definido tres indicadores de eficiencia energética (REI):

Definición 1. Denótese por *consumo específico de vapor* (REI_1) al vapor sobrecalentado consumido por cantidad de agua evaporada. Entonces, el *consumo específico relativo* es el consumo específico actual comparado con un caso de referencia histórico (REI_2) o con el resultado de optimización basada en modelo (REI_3).

$$REI_2 = \frac{REI_1\ Actual}{REI_1\ Histórico} \times 100$$

$$REI_3 = \frac{REI_1\ Actual}{REI_1\ Óptimo} \times 100$$

Para calcular el REI_3 se ha planteado el siguiente problema RTO [2]:

Minimizar $J(u)$ *sujeto a:*

- *Evaporar la cantidad deseada de agua*
- *Las ecuaciones algebraicas del modelo*
- *Las saturaciones en las acciones de control*
- *Máxima potencia de enfriamiento en la torre*

Donde la función objetivo a minimizar $J(u)$ es el caudal de vapor sobrecalentado y las variables de decisión u son las consignas óptimas a fijar controladores (ver Figura 5).

Este problema de optimización se resuelve de forma periódica con una frecuencia adecuada (e.g., a la que sea viable y aceptable actualizar las acciones de control) utilizando los desarrollos software presentados en la Sección 2.

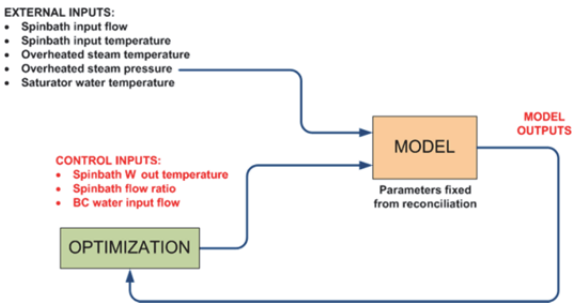

Figura 5. Problema RTO planteado.

4 RESULTADOS

Con el fin de analizar los resultados de la optimización planteada en la sección anterior, se han tomado medidas experimentales en distintos instantes de tiempo con el proceso en estado estacionario. Se ha alimentado al optimizador con los valores reconciliados de las entradas de proceso (caudal y temperatura de la mezcla entrante, temperatura exterior, presión y temperatura de vapor sobrecalentado, etc.), resolviendo un problema de optimización para cada uno de los instantes de muestreo tomados.

Los valores obtenidos para las acciones de control principales muestran una tendencia clara: la **máxima eficiencia** en cuanto al consumo específico de vapor se consigue con la **máxima temperatura** posible de la mezcla al final de los intercambiadores y **mínimo caudal de recirculación** que permita mantener la tasa de evaporación. En la Figura 6 se muestra una simulación de la respuesta del modelo dinámico detallado ante el cambio de las acciones de control existentes en un determinado instante por las predichas para operación óptima.

La visualización del REI_1 nos indica que se podría haber conseguido un ahorro medio alrededor de un 3% (ver Figura 7) con la operación óptima (consignas de los controladores) en los instantes tomados.

Coste computacional. El problema NLP resultante de plantear este ejemplo consta de 21 variables de decisión y 39 restricciones, que se resuelve en menos de 0.1s ejecutándose en un Intel® Core™ i3-2310M con Windows 7 64 bits.

Por tanto, dado que el proceso tiene un transitorio de 45 minutos aproximadamente, este problema RTO se puede resolver de forma periódica (e.gr. cada hora), sin que esto suponga dificultad computacional alguna para un procesador comercial estándar.

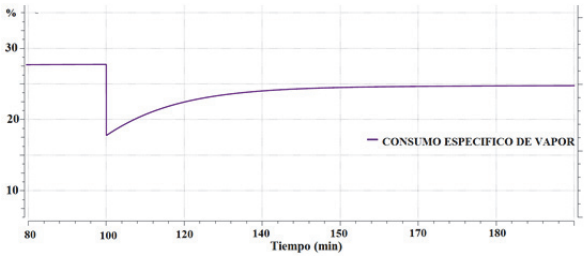

Figura 6. Cambio en el consumo de vapor [3] al aplicar las acciones de control óptimas.

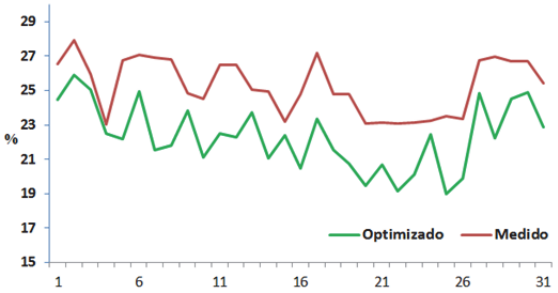

Figura 7. Evolución del consumo de vapor específico para diferentes instantes de muestreo.

5 CONCLUSIONES

Las herramientas de modelado y simulación actuales ofrecen un entorno de trabajo de muy alto nivel. El gran potencial de éstas hace deseable el uso integrado de otro software para resolver problemas de optimización o control. En este sentido, las modernas herramientas NLP permiten obtener soluciones eficientes para problemas de optimización en procesos industriales reales.

Una opción efectiva consiste en crear una interfaz entre los distintos programas especializados en sendos apartados. Todos ellos deben permitir la comunicación automática con los demás, o la resolución se volvería tediosa para el usuario final, con los posibles equívocos y pérdida de datos asociados al proceso (principalmente manual) de traducción.

La interfaz de comunicación programada en este trabajo propone una vía para paliar dicha problemática y abre la puerta a otras aplicaciones en las que se requiere resolver problemas de gran escala en cortos periodos de tiempo.

Los desarrollos de este trabajo se han aplicado para resolver un problema de eficiencia energética en un proceso de evaporación, bastante habitual en la in-

[3] Los valores reales se han escalado y mostrado en porcentaje por razones de confidencialidad.

dustria actual. Sin embargo, las predicciones de ahorro energético obtenidas son obviamente muy optimistas (existencia de error de modelado, etc.) por lo que solo dan una información cualitativa de la posible mejora. Las recomendaciones obtenidas por el optimizador se deben implementar en el sistema de control de la planta para poder ser evaluadas cuantitativamente.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el séptimo programa marco de la Unión Europea (FP7/2007-2013) bajo el contrato nº 604068 y por el Gobierno de España, MINECO (DPI2012-37859). Los autores agradecen a la empresa Lenzing AG por la adquisición de datos y los test experimentales realizados en el proceso real.

Referencias

[1] M. Kalliski, D. Krahé, B. Beisheim, S. Krämer, and S. Engell, "Resource efficiency indicators for real-time monitoring and optimization of integrated chemical production plants," *Comput. Aided Chem. Eng.*, vol. 37, pp. 1949–1954, 2015.

[2] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Society for Industrial and Applied Mathematics, 2010.

[3] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[4] B. M. Bell and J. V. Burke, "Algorithmic Differentiation of Implicit Functions and Optimal Values," in *Advances in Automatic Differentiation*, vol. 64, C. H. Bischof, H. M. Bücker, P. Hovland, U. Naumann, and J. Utke, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 67–77.

[5] X. Li, Z. Shao, and J. Qian, "Module-oriented automatic differentiation in chemical process systems optimization," *Comput. Chem. Eng.*, vol. 28, no. 9, pp. 1551–1561, Aug. 2004.

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

[7] *Modelica. A Unified Object-Oriented Language for Physical Systems Modeling*. Modelica Association. https://www.modelica.org/

[8] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Rev.*, vol. 47, no. 1, pp. 99–131, Jan. 2005.

[9] J. Andersson, J. Åkesson, and M. Diehl, "CasADi: A Symbolic Package for Automatic Differentiation and Optimal Control," in *Recent Advances in Algorithmic Differentiation*, vol. 87, S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 297–307.

[10] A. Walther and A. Griewank, "Getting started with ADOL-C," in *Combinatorial Scientific Computing*, U. Naumann and O. Schenk, Eds. Chapman-Hall CRC Computational Science, 2012, pp. 181–202.

[11] R. Martí, T. Rodriguez, J.L. Pitarch, D. Sarabia, and C. De Prada, "Optimización dinámica mediante diferenciación automática usando EcosimPro y CasADi," presented at the XXXV Jornadas de Automática, Valencia, 2014, pp. 354–361.

[12] F. Vázquez, J. Jiménez, J. Garrido, and A. Belmonte, *Introduction to Modelling and Simulation with EcosimPro*. Madrid: Prentice Hall, 2010.

[13] J. J. J. Chen, "Comments on improvements on a replacement for the logarithmic mean," *Chem. Eng. Sci.*, vol. 42, no. 10, pp. 2488–2489, Jan. 1987.

[14] M. J. Leibman, T. F. Edgar, and L. S. Lasdon, "Efficient data reconciliation and estimation for dynamic processes using nonlinear programming techniques," *Comput. Chem. Eng.*, vol. 16, no. 10–11, pp. 963–986, Oct. 1992.

[15] P. J. Huber, "Robust Statistics," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1248–1251.