Efficient Modelling and Real-Time Optimisation of Stationary Systems: Application to an Evaporation Process

Carlos Gómez Palacín, José Luis Pitarch and César De Prada

Systems Engineering and Control Department, Escuela de Ingenierías Industriales, Universidad de Valladolid.
Valladolid, Spain.
Email: {carlos.gomez, jose.pitarch, prada}@autom.uva.es

***Abstract*—Process optimisation usually leads to a nonconvex mathematical problem, which can be solved in an iterative way by using relevant information about the gradients of the objective function and constraints. In this way, the use of automatic-differentiation tools (AD) is recommended to efficiently obtain such information from a system modelled in steady state. This paper deals with the integration problem between some existent powerful modelling & simulation software and tools for numerical optimal control. The approach here is programming a communication interface between EcosimPro® (modelling), CppAD (AD for C++ code) and IPOPT (Interior-Point Optimiser). The effectiveness of the proposed combined tool has been tested in an industrial evaporation process.**

Keywords–Stationary model; EcosimPro; CppAD; RTO; NLP; Automatic differentiation; Resource efficiency.

I. INTRODUCTION

The efficient use of resources has become one of the main goals in modern industry [1]. In this framework, optimisation problems naturally arise when dealing with process efficiency. Thereby, real-time optimisation (RTO) of processes (i.e., finding the best operating conditions) is a requirement: it reduces the production cost and the environmental impact. RTO is a wide applied online activity in large-scale systems such as petroleum refineries and chemical plants [2]. As RTO problems are solved at regular intervals, very detailed dynamic models can be replaced by simplified stationary ones. As a result, the optimisation problem is generally large, with many constraints from the process model, but relatively few degrees of freedom for optimisation [3].

In order to run an optimisation problem, some quantitative computations are required to measure the goodness of a process. A very extended one is the so-called key performance indicators (KPI) [4]. Nevertheless, new indicators devoted specially to measure resource efficiency in real time (REI) have been recently developed [5]. They focus in a simpler visualization and aggregation of the information between production aspects and employed energy/materials.

After the definition of such REIs, a large amount of model-based optimisation problems on resource efficiency can be stated and solved with nonlinear programming (NLP) software. These tools usually require precise computation of Jacobians and Hessians to provide the optimiser with accurate information to advance through the right way to the optimum [3][6]. There exist several options to compute derivatives in an automatic way. The most used is the approximation by finite differences, which offers a simple and systematic methodology. Nevertheless, the obtained results are not as accurate as desired. Another option is symbolic calculus. This option gives exact results at the price of increasing the computational cost, which may become a bottle-neck in applications with demanding real-time constraints. The current trend is to use tools which implement automatic differentiation [7]. Such tools take advantage of the well-known chain rule to automatically codify the required computations for derivatives directly in machine code. Hence, precise results can be obtained in less computational time than the required for the symbolic calculus [8].

Also good object-oriented software tools for modelling and simulation have been developed in recent years [9][10]. These tools already allow to deal with dynamic optimisation problems (differential algebraic equations) by following a sequential approach [11]. However, these tools still fail in solving with pure discrete or steady-state nonlinear models (presence of algebraic loops), as numerical integration algorithms (e.g., [12]) only provide information about sensitivities in systems with differential equations. Automatic-differentiation tools [8][13][14] are therefore required for those cases. Unfortunately, the lack of integration between such tools and the software for modelling/simulation can make the resolution of the overall problem a very tedious task. Indeed, the difference between programming structures and languages employed force the user to redefine almost the whole optimisation problem each time there is a small change on the process and/or the objective function. This issue is addressed in this work in a preliminary way, proposing a two-way communication interface integrating the commercial tool EcosimPro® (System Modeling and Simulation Software) with an NLP optimisation framework implementing AD, as it is CppAD together with IPOPT (optimisation solver for large-scale nonlinear problems).

The rest of the paper is structured as follows: next section explains the programmed communication interface, Section 3 shows the application of the combined tool to an industrial evaporation process, the obtained results after optimising this case study are shown in Section 4 and, finally, a conclusion section closes the paper.

II. INTERFACE DESIGN

This work mainly focuses on the creation of a generic interface for optimisation of stationary systems. This interface will make the optimisation results accessible from external software, like environments devoted to numerical integration. Therefore, the interface will act as a communication system to provide information from the simulation tool to the NLP optimiser and vice versa. In this way, we will be able to initialise or modify process variables and to recover the values obtained when it is executed. Consequently, the code has to be compiled as a dynamic-link library which can be executed later on inside any simulation software, with the aim of testing performance before implementing the real on-site control algorithms.

By using the inheritance provided by the object-oriented paradigm, we can facilitate the coding of each optimisation problem with a similar interface.

A. Software packages

In this work we decided to use EcosimPro® as modelling & simulation tool, therefore some decisions and developments below are made particularly for this software. Nevertheless, there is no loss of generality due to such choice, because the underlying basic ideas are also valid with other software implementations.

EcosimPro® offers both; [A] a high-level programming language in order to code dynamic models and [B] interfaces to modern numerical integration algorithms. The modelling can be done graphically or with a C++ alike text language. The graphic programming option is only available for certain systems, that have been already defined either by the software development team, or by the user. Also, both styles can be mixed with ease. It can use external functions and classes definitions, with the only need of dynamic-link and static libraries and a header file for the classes definitions. In summary, EcosimPro® is a very powerful tool for modelling/simulation which already incorporates algorithms for dynamic systems optimisation, including the computation of exact gradients via sensitivities. However, derivatives cannot be computed in an exact way for optimisation of stationary models, resulting in a loss of performance. This issue limits also the use of efficient NLP solvers like IPOPT [6], which requires good information about first and second derivatives as input.

In order to incorporate support for efficient optimisation methods linked to stationary models, we suggest here the use of the optimisation tool inside the simulator by using libraries. In order to perform a stationary model-based optimisation, we chose the interior-point NLP solver IPOPT for this work. This package has been proven efficient in large-scale nonlinear optimisations. It is supplied with public license (GPL) under the project Coin-Or, that pretends to give free access mathematical tools globally. Although IPOPT can be ran from several ways, we will use its C++ interface in order to create the files previously defined as needed.

IPOPT is able to reach a local optimum spending less iterations than other gradient-based algorithms thanks to, not only asking for the first derivatives with respect to the decision variables, but also for the second ones. Therefore, it will obtain faster and more accurate results if this information is given with precision.

To efficiently compute and provide derivatives, we chosen the tool CppAD [7], because it implements automatic differentiation in C++ source and it is also maintained as part of Coin-Or project. It provides matrix work functions with derivatives, making the computations of Jacobians and Hessians easier. This software implements AD by using an internal data type to record the operations chain. Then, it can make the differentiation with forward or reverse direction. Forward direction has a calculus time expended around three times the number of variables. While, the reverse option is related to three times the time expended in computing the functions. This means that the appropriate choice will depend on the kind of problem, i.e., for a small problem with a huge number of variables it's better to choose a reverse calculus and for a large problem with very few decision variables is better to work with the forward option. Although CppAD is quite powerful, it doesn't have interfaces to any numerical integration algorithm. Therefore, it cannot be used to solve dynamic problems as it is.

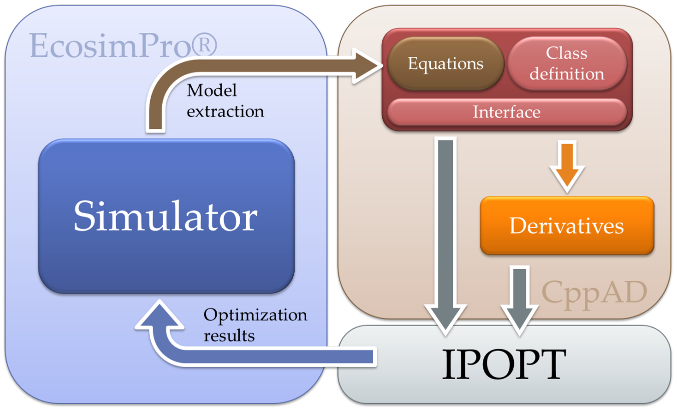

Figure 1. Overall resolution procedure.

B. Model extraction

First, let the user generate a dynamic model with EcosimPro® in a very intuitive way: equations don't need to hold causality neither a syntax order. Then, once the model is correctly finished, the simulation tool translates it internally to a causal mathematical one. This last model is saved as an *html* file. However, this file can't be directly used by AD tools; it needs a previous translation phase to be coded into the proper programming language (e.g., C++). In this work this translation is done almost completely by a command-line translator, developed to implement the previously described interface from the mathematical-view file. See Figure 1 for a clearer representation of the overall procedure.

Our translator runs over the command line under a 64-bit Windows® OS. It was compiled in this way in order to take advance of the improved memory-handling capabilities of 64 bits systems, however a 32-bit version can be compiled too.

The translator works as follows: first, all the model variables are found and classified as internal variables (the ones that might be modified during the optimisation), data variables (fixed values) or decision variables. Then, the equations are translated into C++ language and, as EcosimPro® already has arranged them by type, the ones that have to be constraints in the stationary model are chosen. The dynamics (if there is any) is removed (by setting the derivatives to zero on the corresponding state equations) to get a stationary model.

The auxiliary functions, which have been already coded using the modelling software (i.e., those belonging to separate libraries in EcosimPro®), have to be translated to CppAD code too. However, their code is not included in the main *html* file,

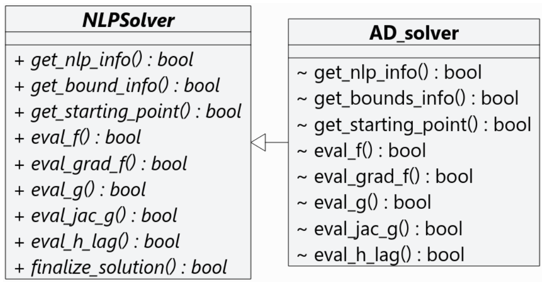

Figure 2. Communication with the optimiser.

despite being used in the model equations. Only the function's name together with its corresponding library is included as a label in the *html* file (e.g., the function maximum coded inside the math library will be labelled as *math.maximum*). This translation procedure can be done by slightly modifying a plain C++ code that EcosimPro® automatically creates each time the functions are defined. However, due to the issue of finding the particular path of these C++ files in each computer, this step is not fully automatised yet.

C. Integrating the optimiser

In order to use IPOPT as a C++ coded program, the methods for an already defined abstract class inside IPOPT have to be implemented. It has to offer a function-call in order to obtain the cost-function and constraints values, besides the Jacobian and the Hessian. All these calculations are implemented using the automatic differentiation algorithm of CppAD, as shown in Figure 2. It will also provide the automation kernel with other valuable information about the problem, as the number of constraints or the limits for those and decision variables. It also has to calculate the sparsity of the Jacobian and Hessians. This is a very important point in order to actually take advantage of all the IPOPT potential.

Therefore, once we have applied the translator to the dynamic model that EcosimPro® previously generated, we have to mix it with the solver interface. This class will be different for each optimisation problem, as it may have different targets: the cost function can't be the same to optimise, for instance the required power, than to execute a data-reconciliation procedure. Despite this, the general root class for each model can be the same, modifying only the few different aspects that have been listed before. In this way, we can state different particular optimisation problems, all of them inheriting from the same stationary model. This idea is summarized in the Figure 3.

Then, when the cost function and constraints are introduced, the dynamic-link library has to be compiled with the chosen compiler. In this particular case, we used MSVC++®.

As soon as we get the compiled code, we can execute it inside external software. Before running the optimisation, in order to obtain reasonable results, the initial guess for decision variables plus upper and lower limits for them and the rest of constraints have to be set up. They can be stated in the same EcosimPro® experiment and then read before calling IPOPT. In order to facilitate this communication, all the variables are addressed by strings in CppAD. The constraints can be labelled also with strings, but by default they are indexed by their position. Instead of using C++ index type, where the zero is the first, we chosen one as the first to keep the EcosimPro® index style.

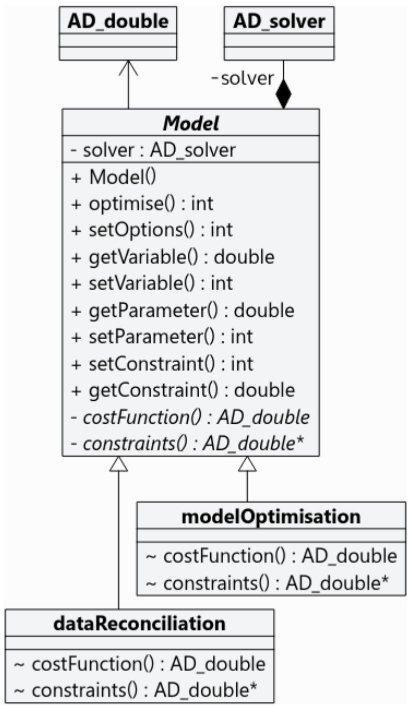

Figure 3. Full classes diagram.

TABLE I. MAIN RETURNED VALUES.

Value	*Meaning*
1	Succesful operation.
2	Maximum iterations achieve.
3	The proggresion is too small.
4	Enough accurate solution.
5	Infeasible problem.
12	Not enough freedom degrees.

Henceforth, with all the initialisation done, the optimisation can be executed and runs in a totally transparent way for the end user. This procedure will return an integer depending on the result of the internal computation. The more common outputs are listed in Table I. If the problem finds a local optimum, we can recover the decision-variables values in order to use them in simulation to check if it approaches the prediction given by the optimiser. In this way, we can pretend the behaviour on a real plant by using the simulation variables as sensor signals, changing model parameters or external input disturbances and updating the control inputs with the decision-variables values.

III. INDUSTRIAL APPLICATION

The software developments presented in the above section have been applied in practice to optimise the hot steam consumption in a multiple-effect evaporation process. This process is formed by: two sets of evaporation chambers and two sets of heat exchangers in serial connection, a barometric condenser, a cooling tower and a saturator (see Figure 4). The system receives and recirculates a liquid mixture of water plus several chemical components with the aim of concentrating it by evaporating a certain amount of water (to partially recover such chemicals by later crystallization).

A. Mathematical model

A grey-box model has been created to represent the process in steady state. The main part is based on first principles:

- Energy and mass balances in the main equipment (evaporators, heat exchangers and condenser):

$$\sum_{i=1}^{n_i} F_i = \sum_{j=1}^{n_o} F_j \tag{1}$$

$$\sum_{i=1}^{n_i} F_i \times H(T_i, C_i) - \sum_{j=1}^{n_o} F_j \times H(T_j, C_j) = \\ = K_p \times [\bar{T} - T_{amb}] \tag{2}$$

  Where F_i, F_j are the n_i input or n_o output masic flows from each equipment respectively, $H(T,C)$ is the specific enthalpy function, T_{amb} is the ambient temperature, $\bar{T}$ can be the average or maximum temperature in each equipment and K_p is a coefficient representing the heat loss to ambient.
- Density relationships between mass and volumetric flows of the mixture, water and steam as a function of temperature and/or pressure (omitted for brevity).
- Heat transmission between fluids inside the heat exchangers:

$$Q = UA \times LMTD \tag{3}$$

$$LMTD = \left[\Delta T_1 \times \Delta T_2 \frac{\Delta T_1 + \Delta T_2}{2}\right]^{\frac{1}{3}} \tag{4}$$

  Where Q is the heat power, U is the heat transmission coefficient and the *mean-logarithmic temperature difference* Δ_T has been computed using the Chen's approximation [15].
- Chemical equilibrium relationships between pressure-temperature-concentration of the mixture inside the evaporation chambers:

$$\bar{P} = P_v(\bar{T}_{mix}, \bar{C}), \quad \bar{T}_{steam} = T_v(\bar{P}) \tag{5}$$

  Where $P_v(T,C)$ and $T_v(P)$ are functions to compute the vapour pressure of the mixture and the saturated steam temperature respectively, and $\bar{T}_{mix}$, $\bar{C}$, $\bar{T}_{steam}$, $\bar{P}$ are averaged temperature and concentration of the mixture and averaged temperature and pressure of the saturated steam respectively, all inside a chamber.

Note that these temperatures and pressures to be used here do not correspond to particular variables in the real plant, i.e., they are fictitious variables corresponding to a mixture of real *unknown* ones.

In addition, and due to the fact that actual processes are of high order and knowing all relationships between internal variables may be difficult, the above physical laws are complemented with experimental patterns. Data reconciliation techniques [16] have been used to identify model unknown parameters, such as the heat-transmission coefficients, and possible patterns/relationships between internal variables (non-measurable) from a set of measurements with the following procedure:

1) Data pretreatment of raw measurements in order to exclude outliers (out-of-range values or sensor failure).
2) Parameter and process variables estimation by solving the data reconciliation problem with the model based on first principles.
3) Experimental identification of patterns.
4) Complete model (physical and experimental laws) validation by data reconciliation with a different set of measurements.

With the aim of reducing the effect of an eventual presence of gross errors in the measurements, robust estimators, like the so-called *fair function* [17], have been used as objective function for the data reconciliation instead of the standard least-squares one.

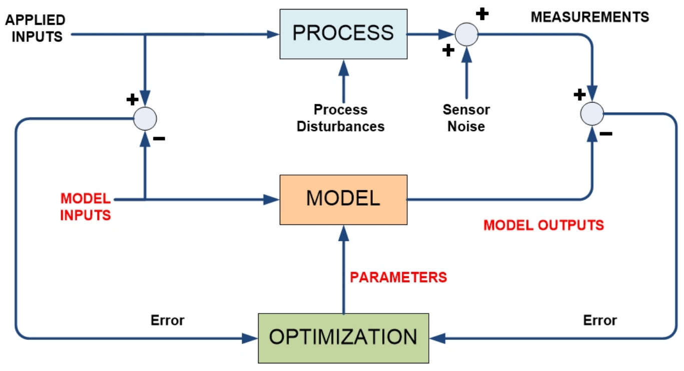

Figure 5. Schema of the data reconciliation procedure.

Consequently with the above procedure, set of tests have been done to collect data from the evaporator line in continuous operation. The tests were executed by running the evaporator either:

- each week choosing different pairings for the set points of the control variables and waiting for the steady state,
- at a constant level between two cleaning cycles.

Note importantly that each variable in the model (input, output, state or parameter) is actually considered as decision variable for the data-reconciliation setup (see Figure 5).

At the end stage, the proposed nonlinear stationary model is formed by 46 equations (6 algebraic loops appearing) and has been successfully validated with plant measurements recorded during 8 months of normal operation.

B. Optimising the efficiency

The goal here is moving the process to an operation point where the resource consumption (mainly the hot steam flow) is minimised by the selection of optimal control set points. To achieve this, 3 REIs have been defined for this process:

Definition 1. Denote by *specific steam consumption* (REI_1) to the overheated steam consumed by amount of evaporated water. Then, the *relative steam consumption* will denote the

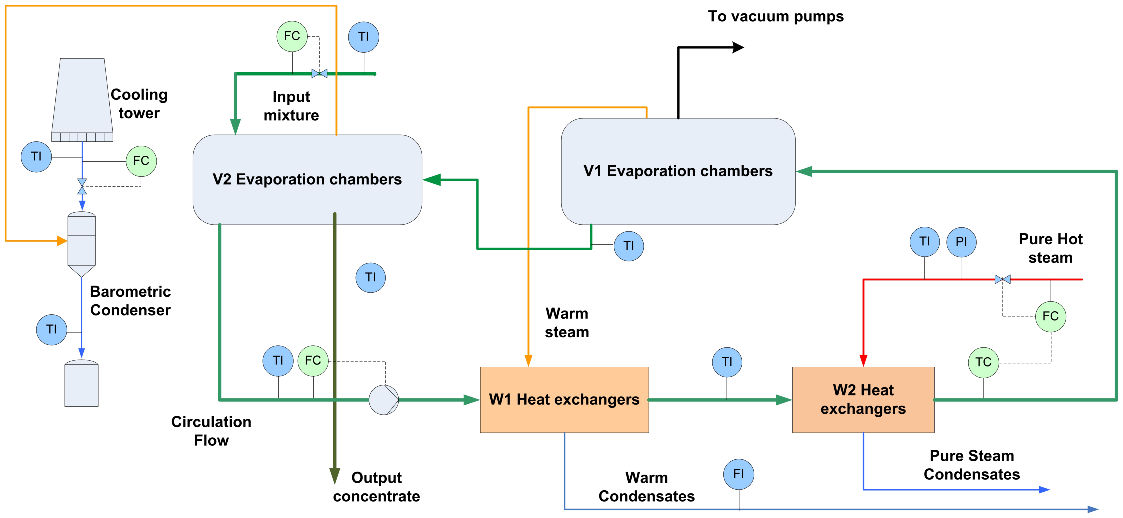

Figure 4. Schema of the evaporation process with existent instrumentation.

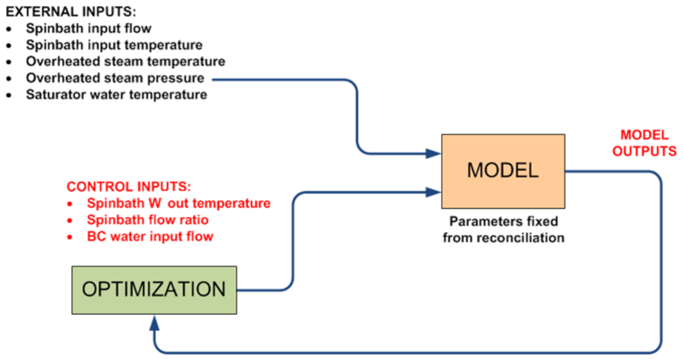
EXTERNAL INPUTS:
Spinbath input flow
Spinbath input temperature
Overheated steam temperature
Overheated steam pressure
Saturator water temperature
MODEL
MODEL OUTPUTS
CONTROL INPUTS:
Spinbath W out temperature
Spinbath flow ratio
BC water input flow
Parameters fixed from reconciliation
OPTIMIZATION

Figure 6. Stated RTO problem.

actual specific steam consumption compared with an historical reference case (REI_2), or with the result of a model-based optimisation (REI_3).

$$REI_2 = \frac{Actual\ REI_1}{Hist\ REI_1}100 \quad REI_3 = \frac{Actual\ REI_1}{Opt\ REI_1}100$$

The following RTO problem [3] has been stated to compute the REI_3. *Minimise $J(u)$ subject to:*

- *Evaporate a desired rate of water*
- *The model equations*
- *Saturation in the control actions*
- *A limit on the available cooling power*

Where the objective function $J(u)$ to minimise is the overheated steam flow and the decision variables u are the optimal set points to the controllers (see Figure 6).

The implementation of this modelling & optimisation problem is carried out by using the software tools & developments presented in Section 2. Then, the idea is to periodically perform an automatic RTO with adequate frequency, e.g., one reasonable enough to update the control actions.

IV. Results

With the aim of analysing the results given by the optimisation setup stated in the above section, a set of experimental data has been recorded with the process in steady state in different operation points. The optimiser has been fed with the reconciled values of the process inputs (flow and temperature of the input mixture, ambient temperature, pressure and temperature of the overheated steam input, etc.), solving a RTO problem for each one of the sample instants taken.

The computed optimal values for the control set points show a clear trend: the higher the temperature and the lower the circulating flow are, the better the process efficiency is. Figure 7 shows a simulation of the dynamic response of the process (obtained with a more detailed dynamic model, different from the simplified stationary one for optimisation) with a change of the current control set points to the optimal ones predicted by the optimiser. The visualisation of REI_1 for different sample points out that average savings around a 3% might be obtained (see Figure 8) with the optimal operation (controller set points).

Be aware that the optimisation setup stated by means of the proposed automatic-translation tool, reaches the same solution than the one obtained by a manual programming approach.

Note that actual experimental data is not included due to a confidentiality agreement with the industrial partner. Values

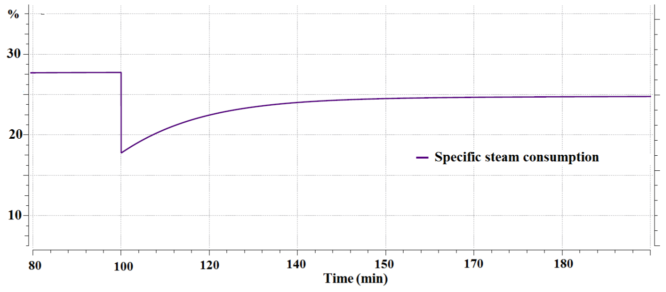

Figure 7. Evolution of the steam consumption before and after applying the optimal control set points.

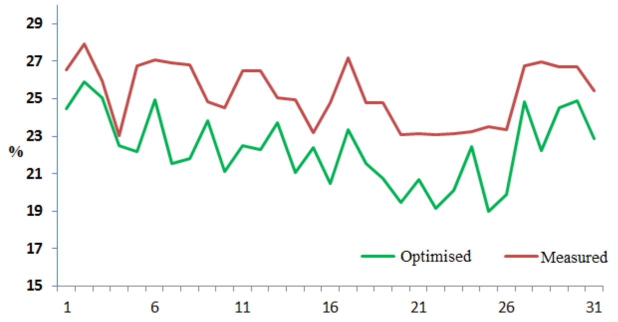

Figure 8. Evolution of the specific steam consumption with sampling.

in Figures 7 and 8 are also scaled in % by the same reason.

Note on computational cost. The resulting NLP problem for this case study has 21 decision variables and 39 constraints. It is solved in less than 0.1s, running in an Intel® Core™ i3-2310M under Windows 7® 64 bits. Therefore, as the transient duration of the process is 45 minutes approximately, this RTO problem can be periodically launched (e.g., each hour) in any standard machine without compromising real-time constraints.

V. Conclusion

The high potential offered by the current software for modelling and simulation makes the use of it desirable in other advanced tasks, such as optimisation. In this sense, modern NLP tools for numerical optimal control allow obtaining efficient solutions for optimisation of actual industrial processes.

This work proposed a communication interface between both specialised software tools in order to facilitate the model translation into an optimisation problem and also to test the proposed *theoretically optimal* solutions later in simulation. In comparison with a manual procedure, this automatic translation tool saves time in the initial model codification and its further possible modifications. In addition, it avoids translation errors and loss of data associated to the human factor. Furthermore, the proposed interface, together with the powerful existent software, opens the door to other demanding applications in which solving large-scale problems in short periods of time is required.

The developments in this work have been successfully applied to address a problem of resource efficiency in an evaporation process, very common in current industry. However, the obtained predictions of savings are, obviously, overly optimistic (existence of mismodelling, etc.). Therefore, they only give qualitative information about a possible right way to improve. The proposed recommendations for the control actions must be implemented in the control system to be evaluated on site.

Obviously, one case study is not enough to completely validate the proposed interface, as there may be complicated process models leading to unexpected situations in the translation procedure. Nevertheless, this issue is very common in software development, where further feedback and polishing stages are needed after a product launch, and it does not detract the usefulness of the approach.

Further work will be focused on increasing robustness to mismodelling and external disturbances by implementing stochastic optimisations and modifier-adaptation setups.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o}604068 and from the Spanish MINECO under project DPI2012-37859.

References

[1] J. Dahmus, “Can efficiency improvements reduce resource consumption?” Journal of Industrial Ecology, vol. 18, no. 6, 2014, pp. 883–897, ISSN: 1530-9290.

[2] J. Han, G. S. Forman, A. Elgowainy, H. Cai, M. Wang, and V. B. DiVita, “A comparative assessment of resource efficiency in petroleum refining,” Fuel, vol. 157, 2015, pp. 292 – 298, ISSN: 0016-2361.

[3] L. Biegler, Nonlinear Programming. Society for Industrial and Applied Mathematics, 2010, ISBN: 0898717027, 9780898717020.

[4] D. Parmenter, Key performance indicators (KPI): developing, implementing, and using winning KPIs. John Wiley & Sons, 2010, ISBN: 9780470095881.

[5] 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,” in 25th Euro. Sym. on Computer Aided Process Eng., ser. Computer Aided Chemical Engineering, K. V. Gernaey, J. K. Huusom, and R. Gani, Eds. Elsevier, 2015, vol. 37, pp. 1949 – 1954, ISSN: 1570-7946.

[6] A. Wächter and L. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” Math. Programming, vol. 106, no. 1, 2006, pp. 25–57, ISSN: 0025-5610.

[7] B. Bell and J. Burke, “Algorithmic differentiation of implicit functions and optimal values,” in Advances in Automatic Differentiation, ser. Lecture Notes in Computational Science and Engineering, C. Bischof, H. Bcker, P. Hovland, U. Naumann, and J. Utke, Eds. Springer Berlin Heidelberg, 2008, vol. 64, pp. 67–77, ISBN: 978-3-540-68935-5.

[8] X. Li, Z. Shao, and J. Qian, “Module-oriented automatic differentiation in chemical process systems optimization,” Computers & Chemical Engineering, vol. 28, no. 9, 2004, pp. 1551 – 1561, ISSN: 0098-1354.

[9] “EcosimPro. Dynamic Modeling & Simulation Tool,” URL: http://www.ecosimpro.com/ [accessed: 2015-09-09].

[10] “Modelica. A Unified Object-Oriented Language for Physical Systems Modeling,” developed by the Modelica Association. URL: https://www.modelica.org/ [accessed: 2015-09-09].

[11] P. E. Gill, W. Murray, and M. A. Saunders, “SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization,” SIAM Review, vol. 47, no. 1, Jan. 2005, pp. 99–131, ISSN: 0036-1445, 1095-7200.

[12] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, “Sundials: Suite of nonlinear and differential/algebraic equation solvers,” ACM Trans. Math. Softw., vol. 31, no. 3, Sep. 2005, pp. 363–396, ISSN: 0098-3500.

[13] J. Andersson, J. Åkesson, and M. Diehl, “Casadi: A symbolic package for automatic differentiation and optimal control,” in Recent Advances in Algorithmic Differentiation, ser. Lecture Notes in Comp. Science and Engineering, S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther, Eds. Springer Berlin Heidelberg, 2012, vol. 87, pp. 297–307, ISBN: 978-3-642-30022-6.

[14] 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, ISBN: 978-1-4398-2735-2.

[15] J. J. J. Chen, “Comments on improvements on a replacement for the logarithmic mean,” Chemical Engineering Science, vol. 42, no. 10, 1987, pp. 2488–2489, ISSN: 0009-2509.

[16] M. Leibman, T. Edgar, and L. Lasdon, “Efficient data reconciliation and estimation for dynamic processes using nonlinear programming techniques,” Computers & Chemical Engineering, vol. 16, no. 1011, 1992, pp. 963 – 986, ISSN: 0098-1354.

[17] P. Huber, “Robust statistics,” in International Encyclopedia of Statistical Science, M. Lovric, Ed. Springer Berlin Heidelberg, 2014, pp. 1248–1251, ISBN: 978-3-642-04897-5.