DYNAMIC OPTIMIZATION BY AUTOMATIC DIFFERENTIATION USING ECOSIMPRO Y CASADI

Rubén Martí*, Tania Rodríguez*, José Luis Pitarch*, Daniel Sarabia**, César de Prada*

*Dept. of Automatics and Systems Engineering, Escuela Ingenierías Industriales, Universidad de Valladolid. C/ Real de Burgos S/N, 47011, Valladolid. {ruben, tania.rodriguez, jose.pitarch, prada}@autom.uva.es

**Dept of Electromechanical Engineering (Automatics and Systems Engineering Area), Escuela Politécnica Superior, Campus Rio Vena, Universidad de Burgos. C/ Francisco de Vitoria s/n, 09006, Burgos. dsarabia@ubu.es

Abstract

The problem of the dynamic optimization of industrial processes using automatic differentiation tools is tackled in this paper. Said optimization problem is solved both with a sequential approach (integration of the continuous model of the process) as well as simultaneous one (discretization of the continuous model). A comparison of the two is also provided. The implementation and solving of the problem has also be performed, to complement the above, with the EcosimPro and CasADi software packages.

Key Words: Dynamic optimization, simultaneous approach, sequential approach, orthogonal positioning, non-linear optimization, NLP, SQP, automatic differentiation.

1. INTRODUCTION

Optimization is a necessity in practically every area of engineering. Typical process design, modelling, identification and control problems include some stage of optimization, whether static, dynamic, off-line or in real-time. Additionally, dynamic optimization plays a increasingly important role in other disciplines, like air traffic control and aerospace applications.

This article focuses on non-linear dynamic optimization [2], i.e., the system and/or function object of the problem is not linear and part of the set of constraints are differential-algebraic equations (DAEs). This is the normal type of problem to be solved in predictive control applications based on modelling (MPC), planning and sequencing of processes by batches, real-time optimization of the plant (RTO), data reconciliation, etc.

There are two methods for solving dynamic optimization problems: the so-called indirect methods, based on the classical optimal control theory, in which the derivative of the optimality conditions is found using calculus of variations [3], and the direct methods, that seek an approximate solutions using numerical calculation methods [4]. While the use of indirect methods ends in a problem of multiple boundary condition points, hard to solve in many cases, the use of direct methods results in the solving a non-linear programming problem that, given the computing power available in today's CPUs, is easier as it is not as complicated.

There are presently two ways to tackle a non-linear optimization problem using the direct method: the sequential approach and the simultaneous approach. The basic idea of the sequential approach is to solve the problem by iterating between a simulation stage, in which the dynamic constraints (ODEs and DAEs) of the system are numerically integrated, and an optimization stage, where a problem with algebraic constraints on the states or any other variable or equation of the model is solve relatively simply [5], [6]. The simultaneous approach, on the contrary, is based on the discretization of the ODEs and DAEs of the system in order to then solve directly a static optimization problem formulated as only algebraic equations [7].

This article looks that the existing problems with the use of non-linear programming tools based on gradients for solving dynamic optimization problems in practice. Moreover, the main advantages and drawbacks of the software packages available on the market (EcosimPro, CasADi) are analysed, in order to provide the end user with a methodology to follow for tackling and solving effectively a dynamic optimization problem, from the initial phase of system modelling up to the achieving of the optimal solution.

This article is organized as follows: the different approaches found in literature for solving a dynamic optimization problem are summarized in the following section; Section 3 contains a description of the CasADi package; next, a proposal for importing into CasADi models generated in EcosimPro is set out in Section 4; Section 5 describes the example dynamic optimization problem worked on, while Section 6 sets out a comparison of the results obtained. Finally, a Conclusions section closes the article.

2. DYNAMIC OPTIMIZATION

The different direct approaches found in literature for solving a dynamic optimization problem are summarized below.

2.1 SEQUENCIAL APPROACH

This approach, as shown in Figure 1 below, is based on resolving the optimization problem iteratively, alternating between one step of simulation, in which the complete set of DAE's is integrated numerically, and one step of optimization, in which the solver or optimizer proposes new values for the decision variables taking into account the results of the simulation (value of the cost function and restraints).

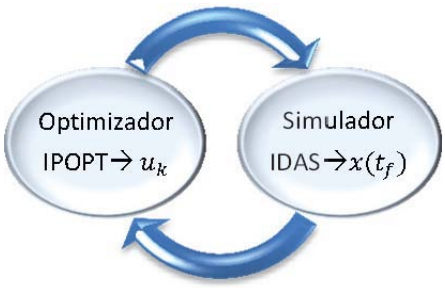

Figure 1: Sequential resolution

The most commonly used is the so-called single shooting method [8] through which, after selecting a parameterization over time of the control actions or of the parameters with respect to which one is going to optimize, the system is integrated from a start time to an end time (given by the integration horizon) at each step of the simulation.

Advantages:

· Optimization problem with a small number of variables (only those corresponding to the parameterization of the control and other parameters)

· Good prediction of dynamic behaviour of the solution obtained, due to rigorous integration using variable step integrators

· Good handling of discrete events

Limitations:

· High computing cost to obtain solutions for large scale dynamically complex problems (time constants very disparate, hybrid systems, etc)

· Including path constraints on states requires complex conditions

· The gradients of the cost function and of the constraints are difficult to obtain (analytically or by automatic differentiation), as the chain rule need to be applied under the symbol derivative and integration symbol

In order to overcome this limitation, 'multiple shooting' methods were developed, which propose dividing the integration horizon into n_k intervals, unrelated among themselves from the point of view of the simulator. This opens up the possibility of halting the execution of tasks in the simulation phase (when the greatest amount of CPU time is generally consumed), and speeds up the resolution of the problem. But the optimization problem still grows in size as new decision variables appear (initial states in each interval) and new constraints (continuity between intervals) [8].

In this method, the path constraints can be included more easily at the initial and final points of each interval. However, this does not guarantee its strict compliance (unless they are explicitly included as "single shooting") at the intermediate points of the intervals.

2.2 SIMULTANEOUS APPROACH

As its name implies, this approach resolves the problem of optimization and evaluation of the constraints simultaneously. See Figure 2 below. To do this, the dynamic problem first has to be converted into an algebraic one by discretization of the original set of DAE's.

Figure 2: Simultaneous resolution scheme.

In order to obtain a good discrete representation of the process, the literature recommends using a technique known as discretization by positioning. This technique divides the time horizon into n_k intervals or finite elements (not necessarily uniform) in which the dynamics of the system is approximated by a linear combination of Lagrange polynomials $P_k(t)$ of preset degree. See [7] for more a more detailed description.

Advantages:

- Discretization by orthogonal positioning can be easily automated
- The calculations required by an integrator are eliminated in order to simulate a continuous system
- The path constraints are easily implemented (in each collocation point)
- The gradients of the cost function and of the constraints are obtained practically exactly and in much shorter calculation times, thanks to the purely algebraic model

Limitations:

- The size of the optimization problems grows excessively both with regard to the decision variables (states at each collocation point) and to the constraints (path constraints and continuity between finite elements).
- It is necessary to select a priori the number of finite elements and degree of the polynomials in order to obtain a suitable discretization of the differential equations. It may not be possible to adequately represent the original dynamics, especially in problems with very complex dynamics.
- It may be difficult to deal with discrete events, due to the fact that the collocation points are selected a priori and there is no concept of variable step.

3. **CASADI:** SOFTWARE FOR NUMERICAL OPTIMIZATION

CasADi [9], [10] is a general purpose package that offers communications tools and interfaces aimed at proposing and solving dynamic optimization problems. It was originally written in C++ but now it also exists as a module that can be integrated in Python and as a 'toolbox' for Octave.

CasADi may be understood, see Figure 3, as an environment that efficiently combines a symbolic engine with automatic differentiation algorithms and incorporates communication interfaces adapted to make calls to different numerical integrators of differential equations (CVodes, IDAS, etc) and resolvers of non-linear optimization problems (IPOPT, SNOPT, etc).

However, CasADi is ***not*** a powerful and complete symbolic engine (eg, Mathematica, Maple, etc), nor is it an automatic differentiation source code software program in general, nor is it a simulator or an optimizer.

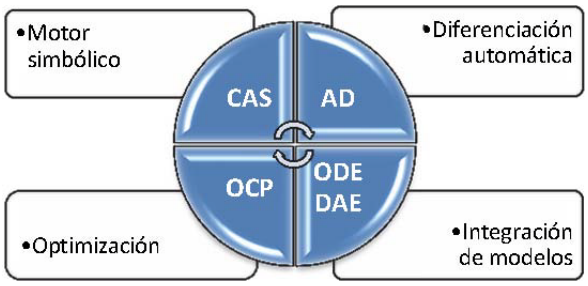

Figure 3: Structure of the CasADi package

Advantages of CasADi:

· Includes the automatic calculation of gradients and Hessian matrices in a manner that is efficient and transparent to the user, by automatic differentiation, which results in a substantial improvement in computing time

· Once the dynamic model has been installed, any direct method (sequential/simultaneous) can be used to resolve the dynamic optimization problem

· It is written in open-source software and works with IPOPT, which is a non-linear optimizer based on interior point algorithms [11], which are more efficient today

Limitation: The process dynamic models have to be entered following a given syntax, based on a structured modelling language, which means less flexibility during the modelling phase

4. IMPORTING MODELS

As mentioned above, the automatic differentiation package is implemented as a module in Python, C++ and a toolbox for Octave. Among the different options available, the authors have opted for Python mainly because it is a high-level interpreted language, it provides multi-paradigm programming, and because it supports multiple platforms, ie it allows users to execute the same program on different machines just by having a Python interpreter installed, and so is independent of the operating system used.

The description of the dynamic models for CasADi has to be done following a specific syntax. The interfaces offered by CasADi for communication with external integrators and optimizers assume that the models will be provided following a algebraic-differential structure, as shown below:

$$\begin{aligned} \dot{x} &= f_{ode}(t, x, z, p) \quad x(0) = x_0 \\ 0 &= f_{alg}(t, x, z, p) \\ \dot{q} &= f_{quad}(t, x, z, p) \quad q(0) = x_0 \end{aligned} \tag{1}$$

Where the equations are divided into differential, algebraic and quadratic, where t is the function of time, x the state variables, z the algebraic variables, and the parameters p.

At this point we come across the main limitation of CasADi: the model entered using following the syntax (1) is not very flexible and if a physical component is changed, added or eliminated, the equations in state space are no longer valid, and the all have to be rewritten (1). Here it becomes necessary to connect powerful object oriented modelling tools, like EcosimPro, to CasADi.

EcosimPro is a powerful modelling and simulation software package that includes an object oriented modelling language for modelling and simulating both continuous and/or discrete systems. EcosimPro can be used in any problem that can be formulated as a system of differential-algebraic equations and discrete events. It is therefore a general purpose acausal object oriented simulation software package.

For that reason, this paper proposes the combination of both tools to make use of the advantages of both the automatic differentiation and the dynamic optimization (CasADi and IPOPT), as well as its flexibility in the development and maintenance of models provided by the object oriented modelling languages (EcosimPro).

To import the EcosimPro models in the CasADi interface, a script has been developed which translates the EcosimPro model into form (1). The modelling phase is shown in Figure 4 and consists of the following steps:

1. Create the model of the system in EcosimPro, using all its powerful features.
2. EcosimPro automatically generates the mathematical model, understood as a complete and final set of DAE's and their order of execution, thanks to a graphical wizard which indicates which variables correspond to state, input or boundary variables.
3. Said mathematical model is generated automatically by EcosimPro in HTML format and will be used for its subsequent transformation to CasADi.
4. Execution of the script developed by the authors. This file, in Python, is a semantic translator that extracts information from the HTML file and automatically generates a Python code with the system model in form (1).

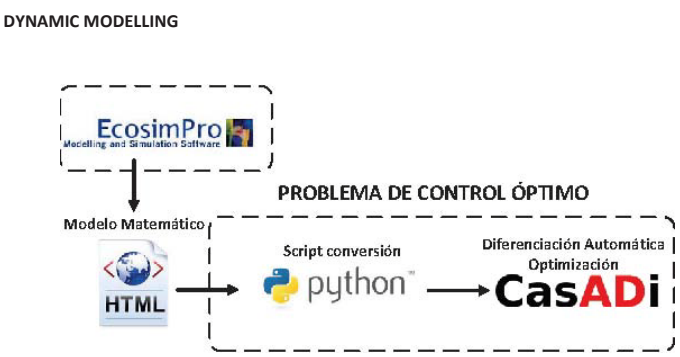

Figure 4: Importing EcosimPro-CasADi models

These steps allow users to use their dynamic models in EcosimPro to solve dynamic optimization problems with the CasADi package, using any of the approaches explained in section 2. In this way, the methodology proposed allows users to flexibly model systems in EcosimPro, to generate a model (internal or simplified) for optimization with CasADi, and to subsequently run simulations with another more detailed and rigorous model of the process (plant), with all this being transparent to the final user, ie it is no longer necessary to manually remodel the equations in the internal representation of the model when some part of the physical process changes.

The integration of both tools (EcosimPro and CasADi) provides the following advantages in comparison to other available combinations:

- Uses software designed integrally for modelling complex dynamic systems

- Maintains coherence between the different formulations of the same original model

- These models are exported to a high level programming environment that has an interface with CasADi, which allows users to precisely calculate Jacobean and Hessian matrices This information is fundamental of obtaining precise solutions in shorter calculation times when resolving dynamic optimization problems of a certain complexity.

5. EXAMPLE

The approach to the dynamic optimization problem has been based directly on the simulation of a specific industrial piece of equipment and it was used to compare the solutions obtained with different approaches to the same optimization problem.

We have created an EcosimPro model of a continuous tank reactor agitated with a cooling sleeve in which a first order exothermal reaction takes place, where a reagent A is partially converted into a product B.

5.1 MODEL OF THE REACTOR

The dynamic model (see Figure 5) has 3 states defined by: the material balance of the reagent A (2), the energy balance in the reactor (3) and the energy balance in the jacket (4).

$$V \cdot \frac{dC_A}{dt} = q \cdot (C_{A0} - C_A) - V \cdot k \cdot C_A \tag{2}$$

$$V \cdot \rho \cdot C_\rho \cdot \frac{dT}{dt} = q \cdot \rho \cdot (T_e - T) - V \cdot k \cdot C_A \cdot \Delta H - Q \tag{3}$$

$$V_r \cdot \rho_r \cdot C_{\rho r} \cdot \frac{dT_r}{dt} = F_r \cdot \rho_r \cdot C_{\rho r} (T_{re} - T_r) + Q \tag{4}$$

$$k = 5.967 e\left(\frac{-826}{T + 278.15}\right) \tag{5}$$

$$Q = U \cdot A \cdot (T - T_r) \tag{6}$$

$$x = \frac{C_B}{C_{A0}} = \frac{C_{A0} - C_A}{C_{A0}} \tag{7}$$

where:

V: Volume of the reactor (m3)

Vr: Volume of the cooling jacket (m3)

CA: Concentration of A in the reactor (kg/m3) simulation (path restraint):

CA0: Concentration of A in the supply (kg/m3) -

q: Volumetric flow of the supply (m3/h)

Fr: Volumetric flow of the coolant (m3/h)

k: Kinetic constant of the reaction (1/h)

p: Density of supply stream (kg/m3)

Cp: Heat capacity of the mixture (kJ/kg K)

CPr: Heat capacity of the coolant (kJ/kg K)

T: Reactor temperature (ºC)

Te: Inlet temperature of the supply (ºC)

Tr: Average temperature of the coolant (º)

Tre: Inlet temperature of the coolant (ºC)

ΔH: Reaction heat (kJ/kg)

Q: Heat exchanged (kJ/h)

U: Heat transfer global coefficient (kJ/hm2K)

A: Heat transfer surface of the jacket (m2)

x: Conversion ration of A to B (non-dimensional)

5.2 FORMULATION OF THE OPTIMIZATION PROBLEM

The purpose of the optimization problem is to maximize the quantity of product B obtained in the reactor at the end of the simulation (t_f=10h), and so the optimization problem to be solved is as follows:

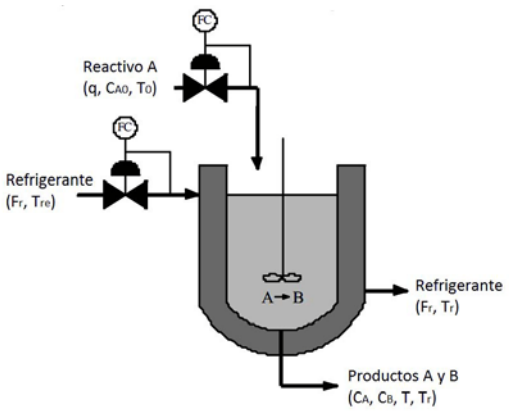

Figure 5: Diagram of the continuous jacketed reactor

minimize
(q, F_r)

$$J = -q(t_f)\cdot C_B(t_f) = -q\cdot(C_A - C_A(t_f)) \qquad (8)$$

subject to:

- Reactor model, equations (2) – (7)

- Final constraint limits of the conversion of reagent A:

$$0 \le x(t_f) \le 1 \qquad (9)$$

- Limits of the temperature of the reactor. The coolant that circulates through the sleeve must be capable of absorbing sufficient reaction heat in order to maintain the temperature of the reactor between certain limits during the entire duration of the simulation (path constraint):

$$L_{\text{int}} \le T(t) \le L_{\text{sup}} \qquad (10)$$

- The following temporary parameterization for each one of the control actions:

$$u(t) = \begin{cases} u_0 \ if \ 0 \le t < 2 \\ u_1 \ if \ 2 \le t < 4 \\ u_2 \ if \ 4 \le t < t_f \end{cases}$$

Where u_0, u_1 and u_2 are the associated decision variables, to be calculated by the optimizer.

The resulting optimization problem is not linear, both in cost function and in constraints, with control actions q and F_r.

5.2.1 Sequential Approach

The path constraint (10) will be implemented in the approach to simple integration with 2 conditions in the form:

$$\max(T(t)) \leq Lim_{\sup} \qquad (12)$$
$$\min(T(t)) \leq Lim_{\inf}$$

For the approach with multiple integration, the constraint (10) has been relaxed to comply with the temperature limits at the extremes of the n_k intervals considered. This means the inclusion of nk algebraic constraint in the problem, in the form:

$$Lim_{\inf} \leq T_k \leq Lim_{\sup} \quad k:1,...,n_k \qquad (13)$$

For this process specifically, it was verified that taking $n_k = 15$ intervals is sufficient to obtain an acceptable solution with regard to compliance with the constraints.

5.2.2 Simultaneous Approach

First, users have to decide on the number of finite element collocation points to provide a good discretization of the system. To do this the process is simulated in an open loop both continuously by means of direct integration of the ODE's, and also discretely by means of a selection of orthogonal positioning to be tested. In this way a number and distribution of finite elements is estimated to assure a reasonably precise representation of the continuous dynamics.

For the simulation, the system has been excited with some constant value inputs:

$$q = 5\ \mathrm{m}^3/\mathrm{h} \qquad Fr = 54.82\mathrm{m}^3/\mathrm{h} \qquad (14)$$

The discretization by selected orthogonal positioning is $n_k = 8$ finite elements with p = 5 collocation points each (degree 4 in P_k) defined by the Radau coefficients table. With the aim of obtaining a good representation of the transient response, the length of the finite elements h_i is not homogenous, but have been distributed under the following pattern:

$$h=[11,1,1,1,1,2,2] \quad (15)$$

The result of the simulation, starting from initial conditions $c_A(0) = 2.891\ \mathrm{kg/m^3}$, $T_r(0) = 65^{\circ}\mathrm{C}$ and $T_c(0) = 51.5^{\circ}\mathrm{C}$ is shown in Figures 6 and 7 below. As can be seen, the discretized model perfectly predicts the dynamics of the continuous process. Therefore, it would seem that this precision in the discretization is sufficient to take on the optimization problem.

A script in CasADi has been generated that automatically returns the discrete algebraic equations by orthogonal positioning corresponding to the DAE's of the system.

In this approach, the constraint (1o) is implemented and forces compliance with the temperature limits at each collocation point:

$$Lim_{\inf} \leq T_{ki} \leq Lim_{\sup} \quad k:1,...,n_k \qquad (13)$$
$$i:1,...,p$$

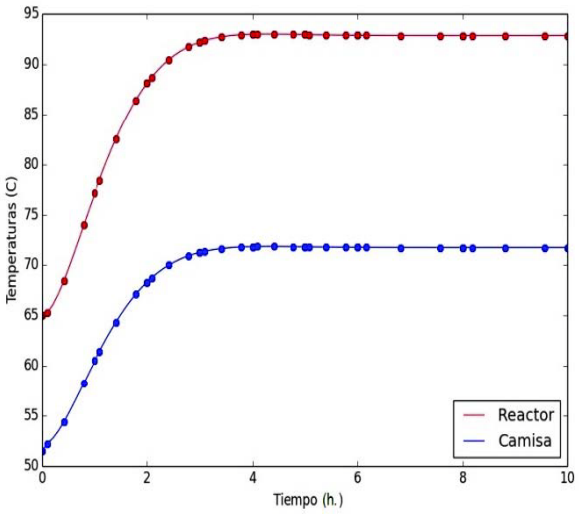

Figure 6: Evolution of temperatures integrating continuous ODE's (solid lines) and using discretization by orthogonal positioning (circular markers).

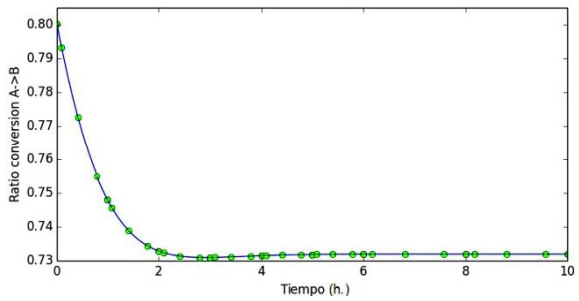

Figure 7: Evolution of the conversion ration from A to B integrating continuous ODE's (solid lines) and by orthogonal positioning (circular markers).

[1] All the calculations were carried out on an Intel®Core™ i5 CPU 3.2 GHZ with 8GB of RAM running Python 2.7 on Windows 7 64 bits SP1.

6. RESULTS

This section presents and compares the results obtained for the reactor example, using both the sequential approach (simple and multiple integration) and the simultaneous approach (orthogonal positioning) with IPOPT as non-linear optimizer. In the three cases, CasADi obtained the gradients and Hessian matrices needed by the optimizer using automatic differentiation.

The optimization problem set out in section 5 has been resolved under the premises and constraints imposed beforehand. The solutions obtained (1) and the size of the resulting optimization problem for each approach are shown in Table 1 and Figure 8 (optimal control actions).

Lastly, the system was simulated applying the previous optimal control actions obtained. Figures 9 and 10 show the evolution of the system states, and compares them to the sampling points predicted by the optimizer with the discretized model set out in section 5.1.

Table 1: Results of the optimization

	Simple Integration	**Multiple Integration**	**Orthogonal Collocation**
Decision Variables	6	51	123
Restraints	9	67	163
Cost Function	-51.0269	-51.0268	-51.0266
Conversion ration x	0.705767	0.705766	0.705765
Total Computing Time	5.48 s	1.765 s	0.06 s
Supply Flow	q_1=1.7896 q_2=3.6328 q_3=5	q_1=1.7334 q_2=3.666 q_3=5	q_1=1.7714 q_2=3.6267 q_3=5
Coolant Flow	F_1=42.16 F_2=51.5 F_3=87.428	F_1=41.93 F_2=51.638 F_3=87.429	F_1=42.054 F_2=51.5 F_3=87.433

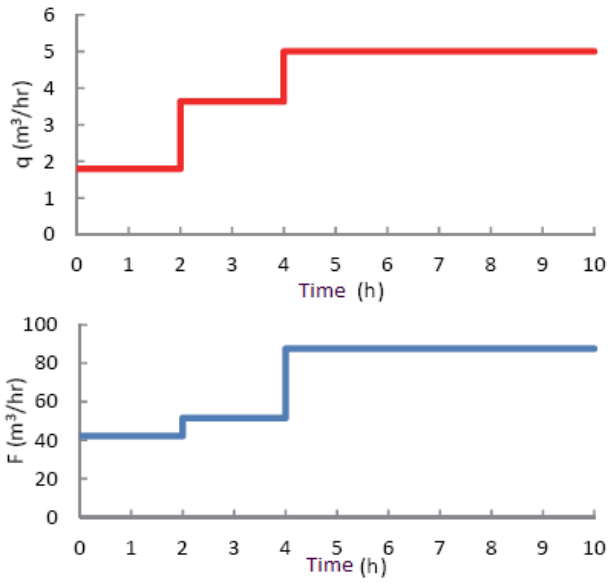

Figure 8: Optimal control actions found, supply flow q and coolant flow F.

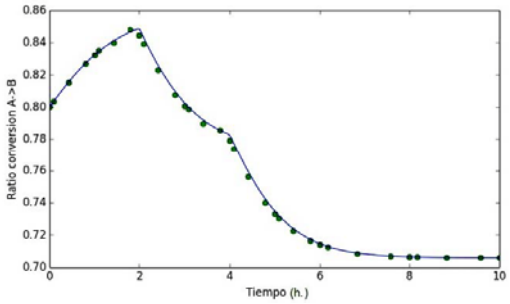

Figure 9: Evolution of the conversion ratio with optimal control. The circular markers indicate the discrete solution obtained by orthogonal positioning.

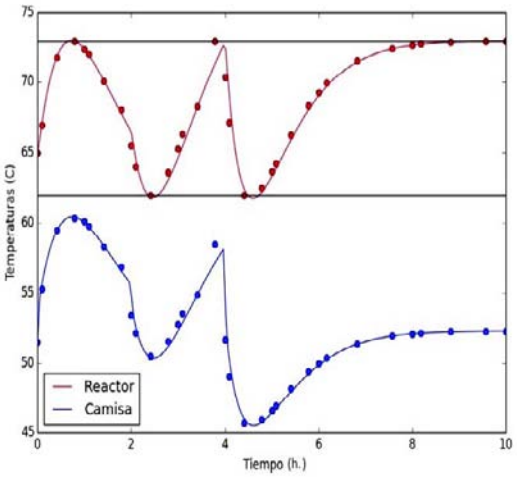

Figure 10: Evolution of temperatures with optimal control. The circular markers indicate the discrete solution obtained by orthogonal positioning.

6.1 DISCUSSION OF THE RESULTS

As can be seen from Table 1, the three approaches resolved with CasADi by automatic differentiation for the calculation of the Jacobean and Hessian matrices find the same solution (variations of around 0.1% are considered insignificant).
In the simultaneous approach, this is possible due to the suitable choice of the discretization parameters (pattern of finite elements and collocation points): obviously the a posteriori simulation of the continuous system with the optimal solution obtained is necessary to verify that the complexity selected in the discretization phase as sufficient.
Note that, in contrast to the simulation in section 5.2.2, the evolution of the states with the optimal control solution that predict the collocation points, does not at times exactly coincide with the simulation of the continuous system (see Figure 10). This is due to the fact that the conditions imposed by the orthogonal collocation do not force the states of the continuous system to coincide with the discrete at the collocation points; they only force the gradient of the system to coincide at those points. In the case that the continuous system is purely polynomial and of the same degree as the one set for the Lagrange polynomial used in the discretization, then the discretization by orthogonal collocation does represent the continuous system exactly. In any other case, this method only provides a more or less accurate approximation depending on the number of preset finite elements and collocation points.

It is worth pointing out that despite the fact that all three approaches reached the same solution, the sequential multiple differentiation approach did it 25% faster than the simple differentiation approaches, and the simultaneous by orthogonal collocation did it using less than 1% of the time taken by the simple differentiation sequential approach.
Note also that the data obtained for the comparison of the sequential approach by multiple integration in this example are simply qualitative, as the possibility of parallelizing the simulation phase offered by this approach makes the final performance depend on the number of cores/processors available, while the performances of the other approaches depend almost exclusively on the maximum frequency of the CPU and of the memory available to resolve the matrix calculations related to the Jacobean and Hessian matrices.

7. CONCLUSIONS

This work has presented the current status of dynamic optimization of processes, approaching the existing problem using the different techniques and tools available. Emphasis is placed on the need to obtain precise gradients and sensitivities rapidly in order to obtain good solutions to optimization problems with algorithms that base the search direction on the first and second derivatives of the model. In this sense the inclusion of automatic differentiation techniques represents a great step forwards.

We have also analyzed the existing software tools and have reached the conclusion that it is necessary to integrate (or at least communicate) different powerful object orientated modelling programs with non-linear optimization programs that include automatic differentiation (EcosimPro and CasADi, in this case). This provides control engineers with good computational efficiency in solving optimization problems, without losing flexibility during the modelling phase.

Lastly, we have compared (by combining EcosimPro and CasADi) the different possible approaches to resolving a dynamic optimization problem in a simple process. The results obtained suggest that the simultaneous approach can deliver an extraordinary advantage with regards to computational time to solve large-scale problems. However, further analyses need to be carried out to study the possible problems of using this approach when the system contains complex dynamics (variable structure associated to discrete events, batch processes, etc) and dynamics with very different time constants that make it complicated to select the orthogonal collocation points. In fact, if a problem requires an extremely precise discretization in certain areas and not so precise in other areas, it would possibly be more efficient to use a variable step numerical integrator.

Acknowledgements

This work was possible thanks to the financing received from the Government of Spain ((DPI2012-37859).

References

[1] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*, 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics, 2010.

[2] G. Hadley, *Nonlinear and dynamic programming*. Addison-Wesley Publishing Company, Incorporated Reading, MA 06187 USA, 1964.

[R. F. Hartl, S. P. Sethi, and R. G. Vickson, "A Survey of the Maximum Principles for Optimal Control Problems with State Constraints," *SIAM Review*, vol. 37, no. 2, pp. 181–218, Jun. 1995.

[4] L. T. Biegler and I. E. Grossmann, "Retrospective on optimization," *Computers & Chemical Engineering*, vol. 28, no. 8, pp. 1169–1192, Jul. 2004.

[5] L. T. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," *Computers & Chemical Engineering*, vol. 8, no. 3-4, pp. 243– 247, 1984.

[6] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, Jan. 2002.

[7] L. T. Biegler, A. M. Cervantes, and A. Wächter, "Advances in simultaneous strategies for dynamic process optimization," *Chemical Engineering Science*, vol. 57, no. pp. 575–593, Feb. 2002.

[8] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. Philadelphia: Society for Industrial and Applied Mathematics : Mathematical Programming Society, 2010.

[9] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," *Tesis doctoral*. Arenberg Doctoral School, KU Leuven, Belgium, 2013.

[10] 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.

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