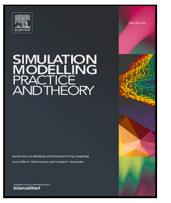
Simulation Modelling Practice and Theory 129 (2023) 102821

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat



A virtual environment of an industrial splitter to test Dynamic Real Time Optimization

Erika Oliveira-Silva [a,b,*], Jesús M. Zamarreño [a,b], Cesar de Prada [a,b], Daniel Navia [c], Sergio Marmol [d], Rafael Gonzalez [d]

[a] Department of Systems Engineering and Automatic Control, School of Industrial Engineering, Universidad de Valladolid, Dr. Mergelina, s/n, 47011 Valladolid, Spain
[b] Institute of Sustainable Processes, Dr. Mergelina s/n, 47011 Valladolid, Spain
[c] Dpto. Ingeniería Química y Ambiental, Universidad Técnica Federico Santa María, Avd. Vicuña Mackenna, 3939, Campus San Joaquín, Santiago, Chile
[d] Departamento Optimización y Control, Petróleos del Norte S.A., 48550 Muskiz, Spain

ARTICLE INFO

Keywords:
Virtual plant
OPC-UA
EcosimPro©
DMC
RTO
System integration

ABSTRACT

A virtual environment (VE) of an industrial splitter has been developed to test a dynamic real-time optimizer prior to on-site deployment. The paper describes the architecture of the VE, which represents the process using the EcosimPro© modeling and simulation software, and a real-time manager to maintain a controlled pace of simulation progress. The virtual process has been integrated with the same commercial predictive controller (Aspen© DMC) as installed in the plant (even with the same configuration) and a dynamic real-time optimizer has been developed in Matlab© and implemented on top of it, allowing the VE to test the optimization layer before deployment in the real factory. The data generated by the plant is stored using InfluxDB©, a time series data platform, for further analysis. The results show that the VE allows revamp studies to be conducted with less cost and more confidence.

1. Introduction

Simulation of any industrial plant has many practical applications, such as controller design, what-if analysis, fault detection, etc. However, integration with third-party industrial applications is not always an easy task. This makes it difficult to conduct more in-depth studies in an economical, safe, and reliable manner in existing industrial facilities.

In industry, OPC (Open Platform Communications) [1] is today a *de facto* standard in real-time data communications among industrial supervision and control equipment, and it is considered as the industrial interoperability standard. It is supported by most industrial suppliers related to the control and automation field, such as PLC (Programmable Logic Controller), DCS (Distributed Control System) or SCADA (Supervisory Control and Data Acquisition system). Since its first development in 1996, the OPC standard has evolved from its first versions (DA: Data Access, A&E: Alarms and Events, HDA: Historical Data Access), now known worldwide as OPC Classic and based on Microsoft Windows technology, to the latest (2008) OPC Unified Architecture (OPC UA), which is platform-independent and service-oriented, integrating and extending the previous OPC classic specifications.

* Corresponding author at: Department of Systems Engineering and Automatic Control, School of Industrial Engineering, Universidad de Valladolid, Dr. Mergelina, s/n, 47011 Valladolid, Spain.
E-mail addresses: erika.oliveira@uva.es (E. Oliveira-Silva), jmzamarreno@uva.es (J.M. Zamarreño), cesar.deprada@uva.es (C. de Prada), daniel.navia@usm.cl (D. Navia), smarmolg@repsol.com (S. Marmol), rgonzalezm@repsol.com (R. Gonzalez).

https://doi.org/10.1016/j.simpat.2023.102821
Received 17 May 2023; Received in revised form 7 August 2023; Accepted 21 August 2023
Available online 22 August 2023

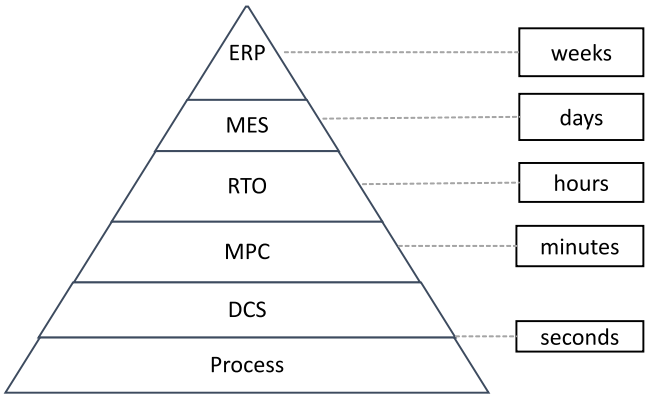


Fig. 1. General automation pyramid.

In this sense, if an OPC envelope is attached to a simulation model, it could be integrated with the same industrial software as in a real installation. Incorporating OPC capability into a simulation model is not new [2–5] but these earlier works involved a lot of programming effort and low flexibility. More recently, [6] presented an innovative commercial solution for generating an OPC DA Server from any simulation written in the EcosimPro modeling and simulation software. EcosimPro [7] is an advanced modeling and simulation tool for modeling systems based on DAEs (Differential–Algebraic Equations) and discrete events. Recently, the EcosimPro software has added the ability to generate OPC UA servers. To our knowledge, no other multidisciplinary simulation software offers this functionality.

An OPC server based on a simulation model is not enough to mimic a real plant. An external real time manager (RTM) is required to pace the integration of the DAEs that form the core of the simulation. The combination of simulator OPC server and real-time manager becomes a Virtual Environment (VE) that is indistinguishable from the real plant from the point of view of external OPC compliant software. This opens up new possibilities for analyzing the model and any other components (such as controllers) in a real-world environment.

Therefore, in order to test algorithms and methods on the simulation before applying them to the real plant, this paper presents a methodology to build a VE of an industrial plant. First, an algorithm of a real-time manager is developed. Then the process is modeled in the Ecosimpro software using a plugin to generate an OPC-UA server. Then the process model is connected to an industrial controller (Aspen©DMC) with the same configuration as in the real plant. Finally, an optimization layer is included in this VE to perform experiments to evaluate if the inclusion of an optimization layer could improve the economic performance of the real process. The ability to use the same industrial control architecture by simply replacing the real plant with a simulation reduces the effort and cost of such revamp studies. It also increases the confidence of industry personnel in new developments.

In addition to describing the architecture for building the virtual environment, this paper shows an example where this virtual plant corresponds to an industrial splitter of the Spanish company Repsol.

After this introduction, the paper is organized as follows: Section 2 gives an overview of the software used in the virtual plant architecture and the other two components to be tested: Dynamic Matrix Control (DMC) and Dynamic Real Time Optimization (dRTO). Section 3 presents the architecture, its application in an industrial splitter and the results obtained as a practical case study. The paper ends with some conclusions and a bibliography.

2. Components of the Virtual Plant architecture

This section provides a brief description of the components and software used in the virtual plant architecture. The virtual environment will mimic different layers of the automation pyramid of the real plant.

The automation pyramid represents the layers of automation in a typical factory, as presented in Fig. 1. As the name suggests, it is represented by a pyramid, with each level providing different functions and working at different timeframes. It is derived from the ISA-95 standard, which describes the interface content between manufacturing operations and control functions and other enterprise functions. Each layer uses different timescales and models to achieve its objectives. The top layer is Enterprise Resource Planning (ERP) and is responsible for production planning over a long time horizon. The next level corresponds to Manufacturing Execution Systems (MES), which is responsible for scheduling operations and dealing with the assignment of products and tasks to appropriate equipment and timing over a period of days or hours. This is followed by Real-time Optimization (RTO), which uses real-time process measurements to calculate the optimal setpoints to apply to the process. Typically, a Model Predictive Control (MPC) layer and then a Distributed Control System (DCS) layer with the basic controllers, actuators and sensors in the field are responsible for making the process achieve these setpoints.

For the purposes of this article, we will work from the RTO level down, ignoring the ERP and MES levels. So, the automation pyramid of the Virtual Environment presented in this paper looks like Fig. 2. The physical process and the basic controllers are represented by an EcosimPro simulation controlled by a Real Time Manager (RTM) algorithm developed in Python. The simulation

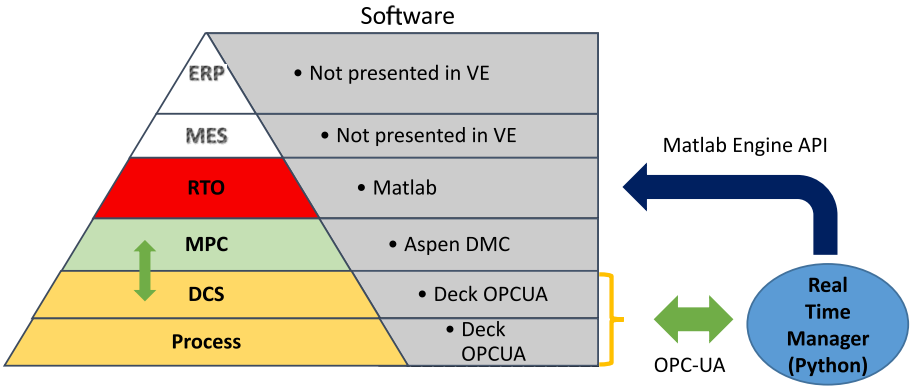


Fig. 2. Virtual Environment automation pyramid.

Table 1
Deck variables and commands.

Tag	Data type	Description
TIME	Double	Simulation time
CINT	Double	Communication interval
TSTOP	Double	Simulation final time
command_reset	Int32	When assigned a number different from 0, the deck is reset
command_run	Int32	When assigned a number different from 0, the server runs the experiment defined in the Deck
command_integ_cint	Int32	When assigned a number different from 0, the server integrates CINT units of time

communicates via OPC UA with the MPC controller, represented here by a real industrial controller, Aspen DMC. Optimization is represented by a dynamic RTO algorithm, a function developed in Matlab and called by the same Real Time Manager algorithm for maintaining synchronization between the virtual process dynamics and the optimization layer.

2.1. Virtual process

As mentioned in the previous section, the simulation of the process was developed on EcosimPro. EcosimPro [7] is a modeling and simulation software based on the object-oriented modeling paradigm. Its simulation language is called EL. Once the model has been built from the DAEs of the process and expressed in the EL language, it is compiled and then it is possible to run experiments and view the results graphically. One of the most interesting features of this software is that it includes a deck generation tool. A deck is a simulation model designed to run as a standalone black box, completely independent from the main program. There are several types of decks, but the most interesting one from an industrial point of view is the Deck OPC UA Server, where the experiment is encapsulated as a binary file to act as an OPC UA server in a network.

Historically, the OPC Foundation® [1] has released several specifications, but the most recent is OPC UA (Unified Architecture). The OPC UA information modeling framework turns data into information and includes full object-oriented capabilities. Today, it has a high level of acceptance in the industry and is one of the pillars of the so-called Industry 4.0.

Each Deck OPC UA server generated from EcosimPro provides a common set of nodes (see Table 1) corresponding to deck variables and commands (nodes corresponding to actions expressed as a variable) that allow a step-by-step execution of the embedded simulation.

Compared to the set of variables and commands available in [6], we can see that the set of commands has been simplified. This simplification was a company decision when migrating the tool from OPC DA to OPC UA, and the capability for the Deck OPC UA server to run the simulation in real time by simply activating an internal flag was lost. However, as we will see later, this is not a handicap in terms of usability, since we can force the simulation to run in real time, for which an external client (Real Time Manager, RTM) has been developed. The RTM is responsible for calling the specific command to integrate numerically the DAEs at a frequency given by the user.

2.2. Real Time Manager

The Real Time Manager is an OPC UA client that communicates with the Deck OPC UA server that contains the process simulation. It uses the variables and commands provided by the server to control the speed at which the simulation is numerically integrated. It was written in Python [8] and its algorithm can be seen in Fig. 3.

First of all, we need to define the temporal variables of our simulation, such as: the communication interval (CINT), which is the period of time at which the simulation updates the values of its variables; TSTOP, which is the simulation time at which the simulation will stop (a high enough value is necessary to carry out an experiment with large settling time); and a speedup factor

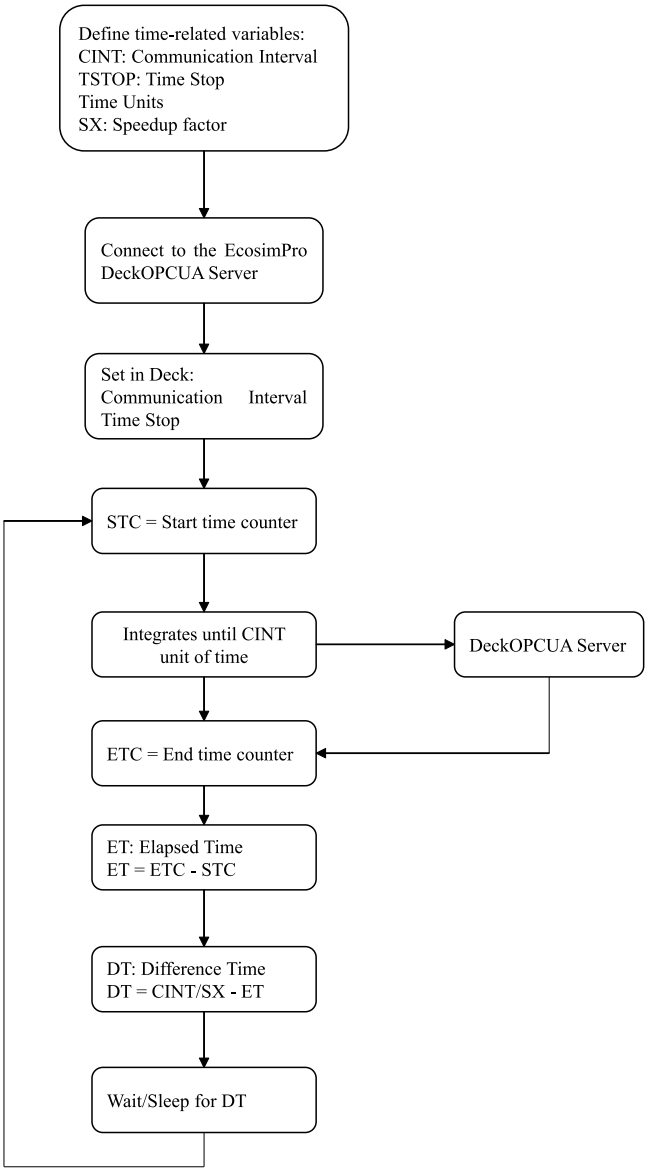


Fig. 3. Real time manager algorithm.

(SX) to accelerate the simulation to obtain results quicker than real time, if necessary. Next, the connection to the Deck OPC UA Server is established and CINT and TSTOP values are written to the simulation engine. Finally, an iterative loop is performed in which the manager calls the *command_integ_cint* command from the Deck OPC UA Server while calculating the Elapsed Time (ET) required for this action. With this time, the algorithm knows how much time to wait until the next integration/iteration so that the simulation runs in real time (subject to a possible speed-up factor). This is explained in Fig. 4 where it can be seen how the Real Time Manager calls the IntegCINT method of the Deck OPC UA server at each time period defined as $CINT/SX$. Although not shown in Fig. 3, for the sake of simplicity, the algorithm performs complementary tasks such as conversion of time units (the simulation time may have been defined in seconds, minutes or hours), checking if the execution time (ET) was faster than real time (if not, a warning is raised) and live information about the evolution of the integration.

Overall, this joint scheme of virtual process and real-time manager implies that any third party application with OPC UA capability could communicate with the model in the same way as if it were the real process. In this paper, these external components will be a DMC controller and a dRTO.

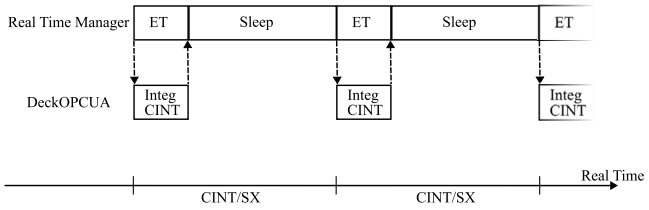


Fig. 4. Real time control of the simulation.

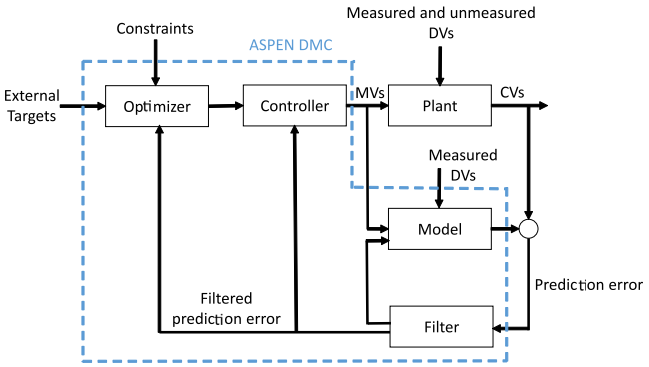


Fig. 5. Structure of an Aspen DMC Controller (Aspen Technology Inc, 2021). DVs are disturbance variables, MVs are manipulated variables and CVs are controlled variables.

2.3. Dynamic Matrix Controller

The standard DMC (Dynamic Matrix Controller) algorithm is an MPC controller that uses a matrix of numerical coefficients to represent the process dynamics (using a step response model) and aims to minimize the difference between the controlled variables and the setpoints, penalizing movements of the manipulated variables (MV) [9]. The DMC plus software, and its latest generation DMC3 from AspenTech®, is a current commercial software based on this technology with other components that allow the inclusion of constraints in the problem and other functionalities.

The Aspen DMC controller application consists of four components: model, filter, optimizer and controller, as shown in Fig. 5. The model aims to calculate the outputs with respect to the change in the inputs without any control action (free response of the system). The filter is an observer that estimates the unmeasured disturbances to calculate the current prediction errors of the model. The combination of model and filter determines the future predictions. The optimizer calculates the best operating point within the multiple possible solutions that satisfy all the constraints, according to the objective function. Finally, the unconstrained controller finds the move plan to achieve the setpoints from the optimizer. All process constraints are handled in the optimizer component. If no feasible solution is found, DMC has tuning parameters to allow relaxation of some constraints. The relaxation can be done by minimizing the global constraint violation or by organizing the CV constraints into a priority order using ranking groups. The constraints with higher rank are considered less important than those with lower rank, so they can be relaxed first to try to find a feasible solution, always trying to minimize the violation.

At each sampling time, Aspen DMC runs a two-layer architecture (see Fig. 6): in the first layer, the local optimizer calculates the future targets subject to the process constraints for the second layer, the controller. The controller (an unconstrained standard DMC) then calculates the future moves to achieve these targets and applies the first move to the process.

2.3.1. Prediction model

The first step in developing an Aspen DMC application is to build a dynamic model. Aspen DMC uses the linear model FSR (Finite Step Response) obtained from an identification algorithm. The process input–output data used by the identification algorithm is obtained from a plant test. During the plant test, several step moves are performed in each MV and the plant data is collected. The plant test is the most critical part of the DMC project as it defines the accuracy of the model. A good model will improve the performance of the DMC. The resulting model represents the open-loop time response of the dependent variable to a step change in each independent variable, while holding all other independent variables constant. For example, in Fig. 7, each curve represents the behavior of a dependent variable listed in the top row (AI-2020, AI-2021, AI-2022) for a step change in the corresponding independent variable listed in the left column (FIC-2001, FIC-2002, etc.).

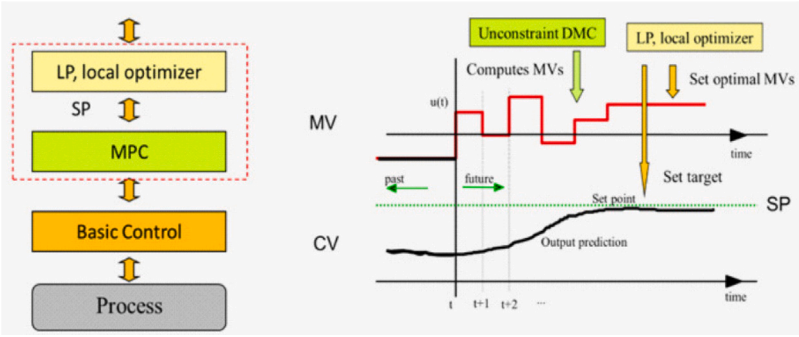


Fig. 6. (a): The two layers in Aspen DMC and the connection with the basic control and process; (b) The controller component predicts the future values of the CVs and MVs, considering the set-points calculated from the optimizer [10].

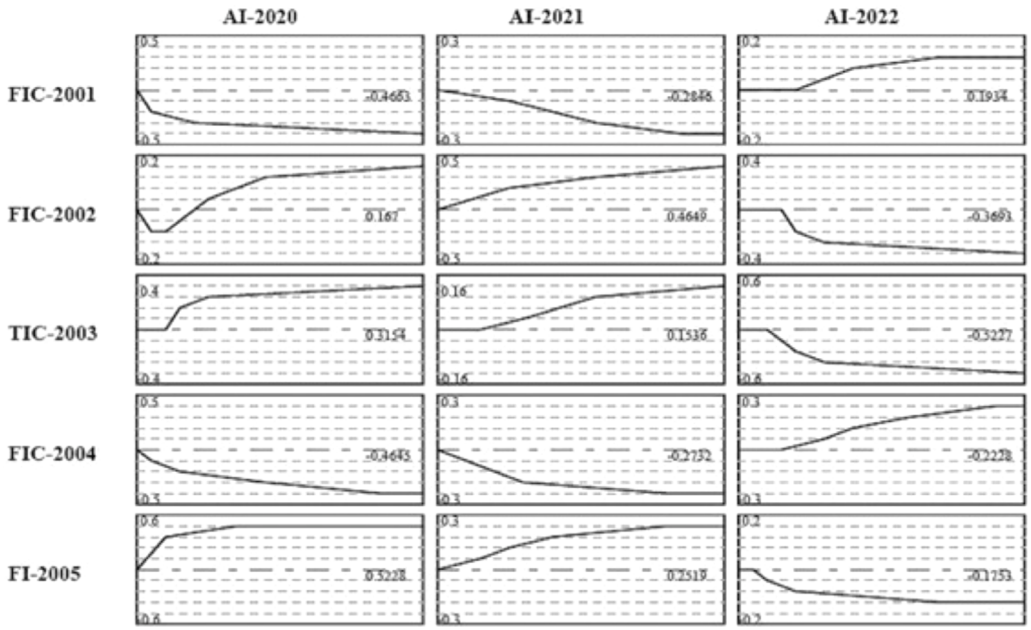


Fig. 7. Example of a FSR model for a complex fractionator [11].

2.3.2. Filter

The Filter uses the difference between the current measurement and the current prediction to calculate a bias (error) that is added to the prediction. In Aspen DMC, three different disturbance models are available for the FIR model: Full Feedback, First Order or Moving Average. Full Feedback uses the difference between prediction and measurement to calculate a bias to be applied to the prediction for the FSR model. First Order is similar to Full Feedback but only allows a fraction of the differences to be applied to the prediction. Moving Average uses an average of past values of the differences to determine the final difference to correct the prediction.

2.3.3. Optimizer

The optimizer uses the same linear model as the controller, but in steady state. The optimizer calculates the best economic steady-state target for the CVs and MVs, and the steady-state target for the CVs becomes the setpoint for the controller. The process constraints are also taken into account in the optimizer component. In fact, this is its main purpose: to compute, at each sampling time, targets that are feasible at the end of the prediction horizon according to the current operating conditions. The Aspen DMC also allows the input of external target information to be followed by the controller. This information is considered as an additional constraint in the optimizer.

2.3.4. Controller

The aim of the controller is to compute the move plan that minimizes the difference between the predicted value and the steady-state targets computed in the optimizer. The optimizer and controller are executed at a fixed time frequency. The optimizer calculates the setpoint for the controller, which uses the information of the prediction horizon and the control horizon to calculate the control strategy. The prediction horizon is the number of future control intervals that the controller must evaluate by prediction when optimizing the MVs. The control horizon is the section of the prediction horizon where the MVs are allowed to move. The controller component uses the dynamic prediction model with the current plant state to compute a control strategy for the control horizon using a numerical minimization algorithm. Only the first move of the control strategy is implemented, then at the next time of execution of the optimizer/controller, the process data is sampled again and the calculations are repeated starting from the new current state. Since the prediction horizon always moves forward, this strategy is called receding horizon control.

2.4. Dynamic Real Time Optimization (dRTO)

The objective of process optimization is to make the right operational conditions that minimize production costs and maximize profits while meeting safety, environmental and quality constraints. RTO is a set of algorithms and techniques that use real-time data to automatically calculate the optimal operating point of a process, taking into account economic criteria.

A traditional RTO solves a steady-state economic optimization problem using a rigorous steady-state model of the process updated using available historical process data. This layer computes setpoints that are implemented in the plant by an MPC layer according to the structure of Fig. 2 and typically have a frequency of hours due to the need for steady-state data. Some industrial processes with frequent disturbances, changing parameters and demand can make the use of traditional RTO questionable since the frequency of execution is very low. In this case, the idea of considering the dynamics of the process should be useful, and then dynamic RTO or economic MPC (eMPC) have been proposed [12]. In dRTO or eMPC, the dynamic information is incorporated into the optimization problem using a dynamic model instead of the steady state model.

3. Application to an industrial splitter

Superfractionators are equipments that separate mixtures of components with very low relative volatilities (values between 1.05 and 1.2) [13]. This equipment is usually located in the final stages of the process and therefore suffers the influence of all upstream disturbances. A very common case in the chemical industry is the separation of propane and propylene. Propylene is an important feedstock for the petrochemical industry and requires a high degree of purity. The need for a column with a large number of plates, high reflux values and a small temperature gradient means that this column has very slow dynamics. It takes more than 10 h to reach steady state, making it difficult to model, control and optimize.

The case study is a simulation of a real propane–propylene splitter at the Petronor refinery in Muskiz, northern Spain. The splitter object of this study aims to produce high purity propylene from a stream of propylene, propane and a small amount of impurities (C2–C4 hydrocarbons). The splitter has a total condenser, a partial kettle reboiler and 135 equilibrium stages. The process is controlled by a DMC controller to maintain the propylene concentration in the distillate within a range of $\geq$97.5% by manipulating the distillate flow, the amount of steam fed to the reboiler, and the pressure of the column (Fig. 8). The process also has regulatory controllers to maintain the level in the accumulator vessel and bottom at a set point by manipulating the reflux and bottoms flow rates respectively.

In this process, both propylene and propane are final products. The distillate below 97.5% of propylene is considered to be out of specification. The out-of-spec product can be sold, but at a much lower price. On the other hand, the price of the bottom product (propane) does not depend on the propylene concentration. However, a higher concentration of propylene in propane reduces the production of the more valuable product.

The two variables manipulated by the DMC are directly related to the main costs (steam) and profits (distillate production) of this plant. It therefore seems logical to add a Real Time Optimization (RTO) layer capable of calculating the process optimum (the setpoints for the DMC) using an objective function that takes into account the prices of these two variables. Typically, a steady state non-linear model based on first principles is used in an RTO, but in our case the size and complexity of such a model does not recommend its use. In addition, this plant has slow dynamics and suffers from a lot of upstream unmeasured disturbances, so our aim is to use the already developed linear model of the DMC for this purpose.

3.1. Virtual process modeling

In order to represent the splitter in the VE, it is necessary to build a model able to predict the dynamic behavior of the column. There are several ways to describe a distillation column using a dynamic model. In general, these models use conservation laws such as mass, energy, and momentum, and time-dependent constitutive equations that define the relationship between intensive variables and extensive variables (as equations of state and equilibrium equations). These equations form a system of differential–algebraic equations (DAE). The mathematical model takes into account the following simplifying hypothesis:

- Constant pressure drop.
- The feed consists of four components: propylene, propane, isobutane and ethane.
- The column is thermally insulated.

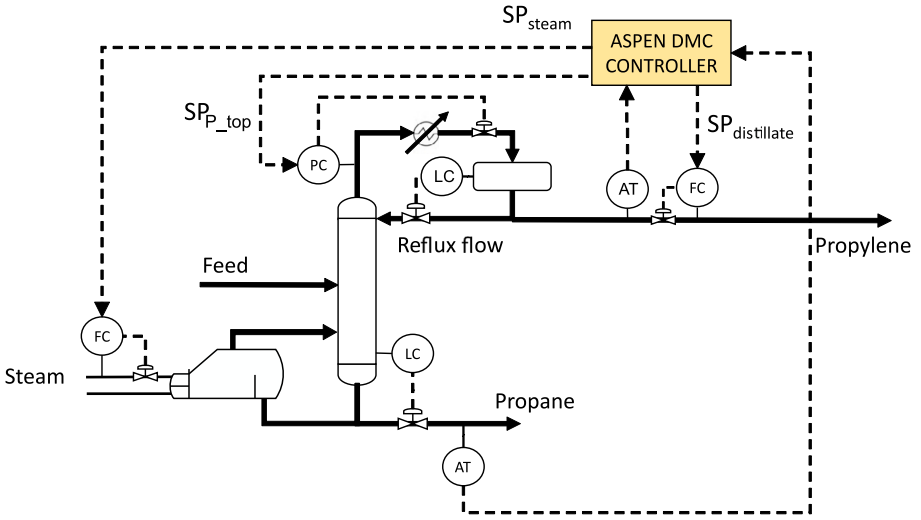


Fig. 8. Flow diagram of the propylene–propane splitter.

The model also considers that the condenser allows subcooling. The equations of the dynamic model were presented in [14]. The described nonlinear dynamic model was simulated using EcosimPro [7]. The model has 12 090 equations and it was solved using the IDAS_SPARSE integration solver [15]. Once the model was built, model validation was performed.

3.1.1. Process model validation

To validate the model, some of the historical process data were compared with the simulation of the Deck OPC UA connected to the Aspen DMC controller. To link the process simulation to the other applications in the VE, the EcosimPro model was converted to an OPC UA deck and connected to the Aspen DMC controller. This simulation used input data from approximately 5 consecutive days (one data per hour during November 2021). The comparison between the simulation and the historical data is shown in Fig. 9. All the results presented were normalized due to enterprise confidentially reasons. It can be seen that the dynamic behavior of the curves is quite similar, presenting a small offset between them. Probably due to noise in the signal from the instruments, the process data also shows a behavior with more peaks. The rigorous dynamic model proposed for the splitter has a dynamic response and gains similar to the real process data. In conclusion, the present model is considered good enough for the purpose of mimicking the real plant.

3.2. Controller

Aspen DMC3 installations have an architecture as shown in Fig. 10. The architecture consists of the following components:

1. Aspen DMC3 online applications server(s): the machine(s) that hosts the active operation of deployed Aspen DMC3 applications. The server receives data from the Aspen CIM-IO server and hosts security assignments.
2. IO sources(s): the machine(s) that provide an interface to the DCS (Distributed Control System). Here we use the Aspen CIM-IO Server.
3. DMC3 Builder client(s): machine(s) where Aspen DMC3 Builder is installed. These workstations are used by the engineering staff to develop, update, test and deploy modeling and controller systems.
4. Aspen APC Web Interface host: server that host the web site APC Web Interface.
5. Aspen APC Web Interface clients with web browser access: machine(s) from which clients can access the APC Web Interface, allowing the engineering staff to monitor the status and manipulate the activities of the deployed control systems.

In order to use the Aspen DMC in the virtual plant, Fig. 10 structure was installed on a PC running Windows Server 2019. We then used the same controller application file from the real plant, changing the IO (input/output) source and IO tag of the original plant for the OPC UA deck in the deployment phase of application development in Aspen DMC3 Builder, see Fig. 11.

Once the connections were tested, the controller was deployed to the online application server. The APC Web Interface (Fig. 12) is then used to switch on the DMC controller and to monitor its actions.

3.3. Optimization

As mentioned above, the main objective of the case study is to test the incorporation of a dRTO layer prior to its implementation in the propane–propylene splitter. The dRTO used is the one presented in [16–18]. In these papers, an eMPC is considered to replace

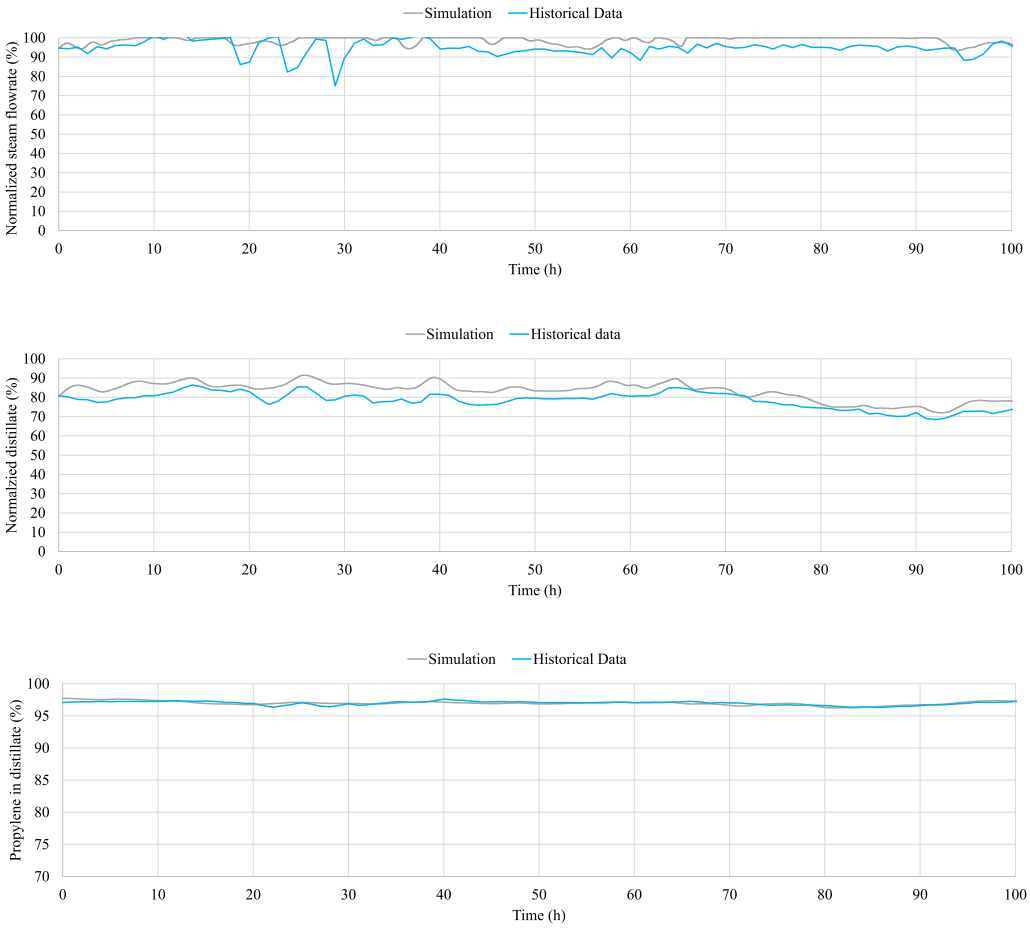


Fig. 9. Results for the closed loop validation.

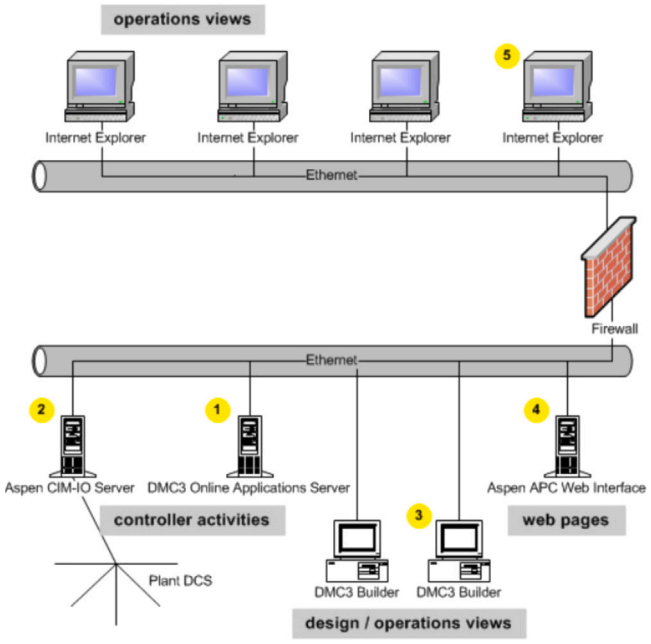


Fig. 10. Architecture installation of Aspen DMC3 [11].

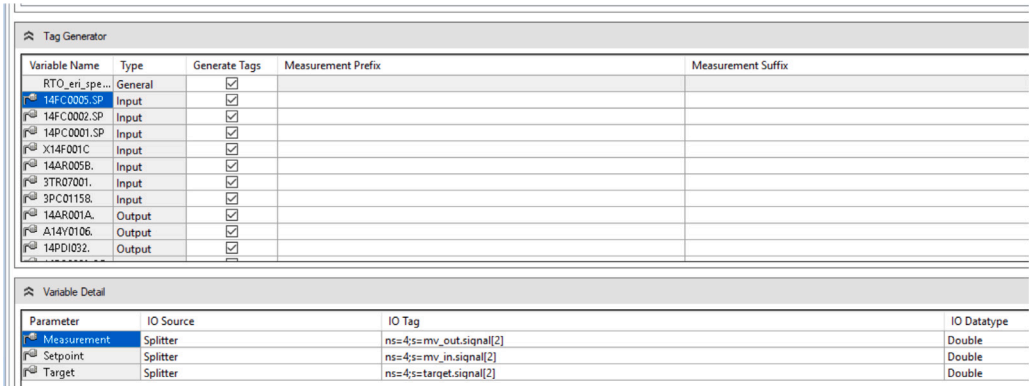


Fig. 11. Connection between deck and Aspen DMC.

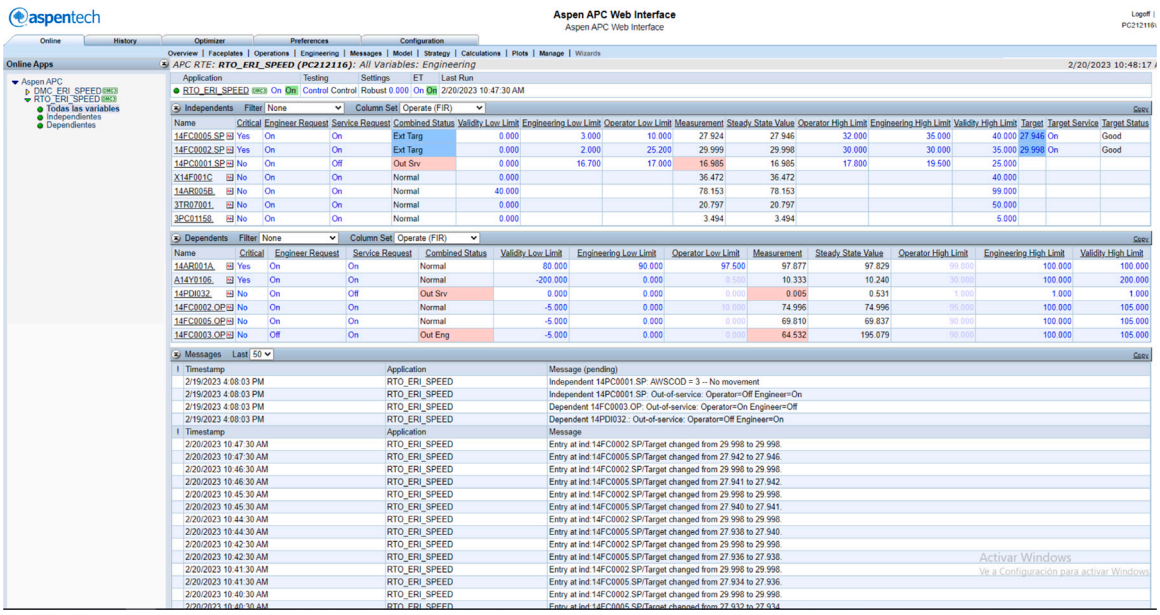


Fig. 12. Aspen APC Web Interface.

the optimization and control layers into a single layer. However, for the present industrial case, it is important to consider the current control structure present in the plant and try to minimize the required changes. It is also important that the optimization layer can be quickly disconnected to return to the previous plant control structure for safety reasons. Thus, the controller layer is kept and an optimization layer is added, following the two-layer architecture of Fig. 2. The optimization problem solved in the dRTO layer will be the same problem of an eMPC, using an economic objective function and the same dynamic linear model of the DMC. The dRTO layer is implemented using a function in Matlab (using fmincon optimization solver). The Matlab function is called by the RTM algorithm in Python and calculates the new setpoints for the Aspen DMC controller. Since the real plant does not have an RTO layer (which is what this case study is intended to evaluate), another change to the DMC application file was required. The current "target type" of the MVs was changed from "none" (the SPs are calculated from the local optimizer) to type IRV, see Fig. 13. The IRV mode allows external targets to be sent to the Aspen DMC and these values are considered as constraints in the optimizer as explained in Section 2.3.3.

The dRTO layer considers the profit from selling propylene and propane minus the cost of producing steam. The product prices depend on whether the propylene specification is met in the distillate (≥97.5% molar), i.e., the price of off-specification distillate is reduced according to the concentration of propylene. All the distillate production is sent to a propylene spherical tank. When the tank is full, the final composition is measured and if it is within specification (≥97.5% molar propylene), the product can be sold. If the quality composition is not achieved, the product must be reprocessed, which significantly increases the production costs. To avoid this reprocessing, the dRTO problem will consider reducing the price of the off-spec product to compensate for the loss of

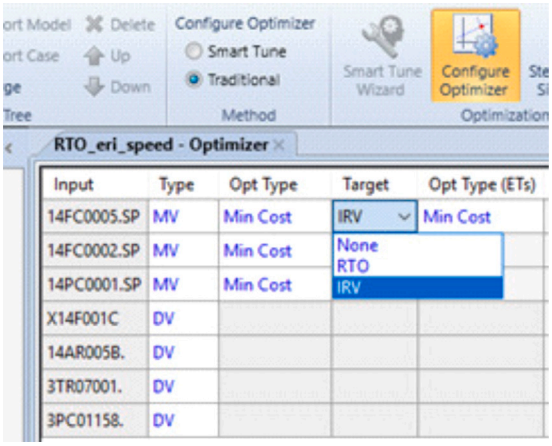


Fig. 13. Changing the current target type of MVs.

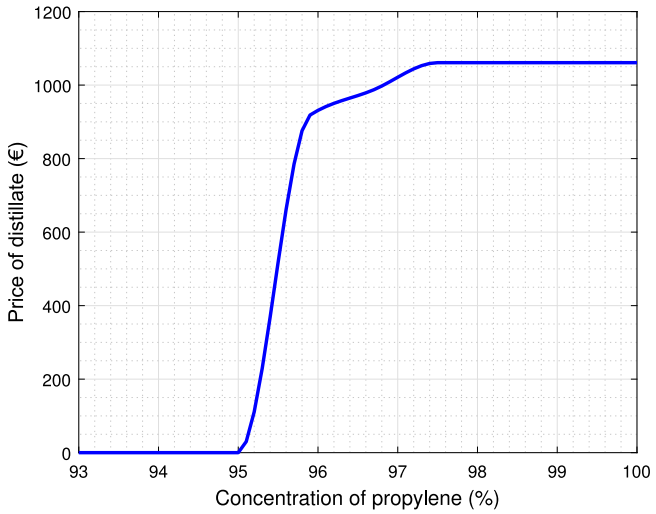


Fig. 14. Price of distillate depends on the concentration of propylene.

profit in producing less distillate of higher purity. For example, if 96.5% propylene distillate is produced in a given time period, the same amount of higher purity propylene distillate (98.5%) must be produced to meet the specification in the tank. Thus, a vector of composition and prices has been constructed, and using the Modified Akima piecewise cubit hermit interpolation (makima function in Matlab®), the distillate price changes with the composition as shown in Fig. 14.

The dynamic optimization problem for the splitter could be written as the problem stated in Eq. (1).

$$\begin{aligned} \max_{x_D} \phi_{splitter} &= p_{C_3H_6}(x_D) \times F_D + p_{C_3H_8} \times F_B - c_W \times F_W \\ \text{s.t.} \quad & \text{DMC dynamic linear model} \\ & x_D \geq 97.5 \end{aligned} \tag{1}$$

where F_D is the distillate flow, F_B is the bottoms flow, F_W is the steam flow, $p_{C_3H_6}$ is the price of propylene, which is a function of the concentration of propylene in the distillate x_D, $p_{C_3H_6}(x_D)$. $p_{C_3H_8}$ is the price of propane and c_W the cost of steam.

3.4. Database InfluxDB

InfluxDB is an open source time series database (TSDB) developed by InfluxData. It is used for storing and retrieving time series data in areas such as operational monitoring, application metrics, Internet of Things (IoT) sensor data, and real-time analytics [19].

The process variables of interest are stored in InfluxDB database and the software allows visualization as shown in Fig. 15.

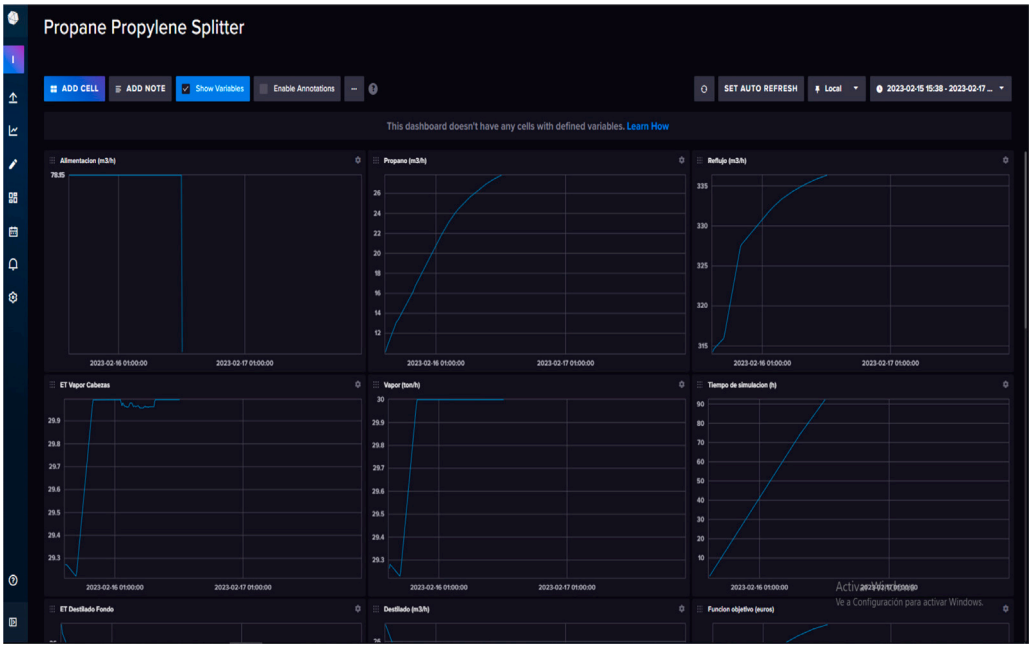


Fig. 15. Process variables that are stored in InfluxDB.

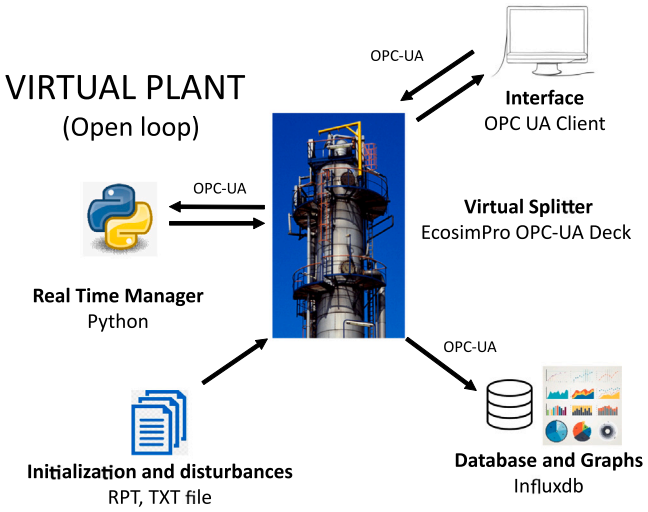


Fig. 16. Virtual Plant in open loop.

3.5. Virtual Plant simulation

The developed Virtual Plant can be used in 4 different architectures. The first one is shown in Fig. 16. In this architecture, the splitter is not connected to the DMC controller, so its MVs (distillate flow and steam flow) could be changed manually inside the RTM code in Python or any other OPC UA client.

The second possible architecture is shown in Fig. 17. In this architecture, the splitter has the same configuration as in the real plant: in closed loop with DMC controller. In this layout, the MVs moves are calculated by the DMC (see Section 2.3.3), which sends the values to the virtual plant via OPC UA.

The third architecture, Fig. 18, considers that the eMPC sends the moves directly to the plant, i.e., the RTO and MPC layers are combined into a single layer. This layout could be interesting to make experiments faster than with the Aspen DMC controller.

Finally, the last possible architecture is presented in Fig. 19. This is the architecture of interest to test the expected benefits of using an upper supervisory layer based on the dRTO paradigm in the real process.

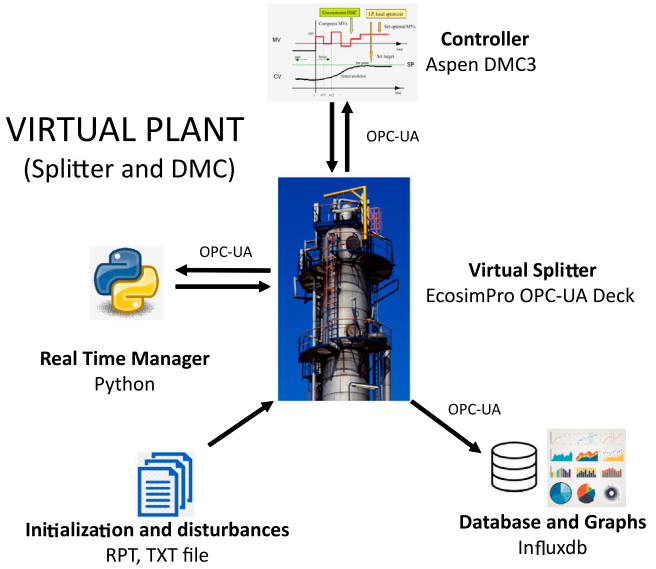


Fig. 17. Virtual Plant in closed loop using Aspen DMC.

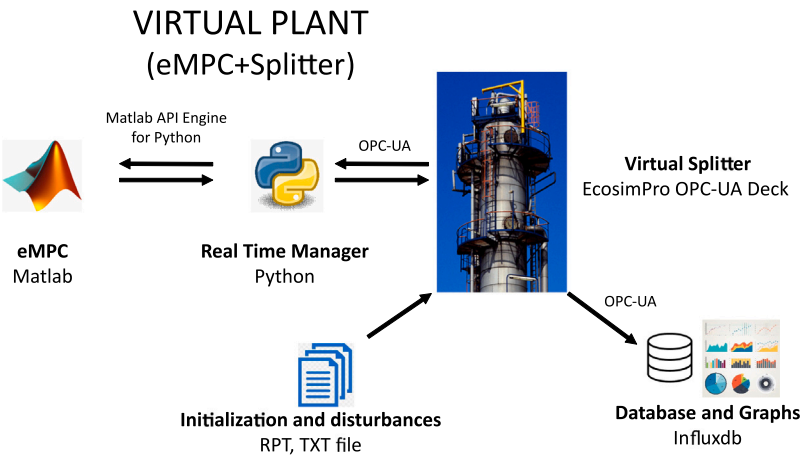


Fig. 18. Virtual Plant in closed loop using eMPC.

The DMC installed at the Petronor refinery has a discrete step response model with a settling time of more than 15 h. All the results presented below were performed on a PC running Windows Server 2019, with a 24-core i9 processor at 3 GHz and 128 GB of memory.

The execution of any of the four configurations of the Virtual Plant is roughly done in the following steps:

1. Run the Deck OPC UA Server
2. Start Influxdb to collect data
3. If using DMC: Launch the DMC3 application using Aspen APC Web Interface
4. Set the CINT, TIME, TSTOP, speedup factor in the RTM Python code
5. Execute the RTM algorithm

Fig. 20 shows the results of the cost function of Eq. (1) for the current architecture of the real plant, i.e., the DMC controller optimizer calculates the optimum operating point (gray line) and the case where a dRTO is installed above the DMC controller (blue line). Fig. 20 shows that the dRTO improves the economic performance of the plant. Both simulations were performed initiating from the same steady-state operation point. This best cost performance during the transient is due to the fact that the dRTO is able to meet the quality constraint for most of the operating time (Fig. 21). To confirm this, the total profit was calculated and presented in Table 2, where an increase of more than 2% can be seen after the implementation of the dRTO.

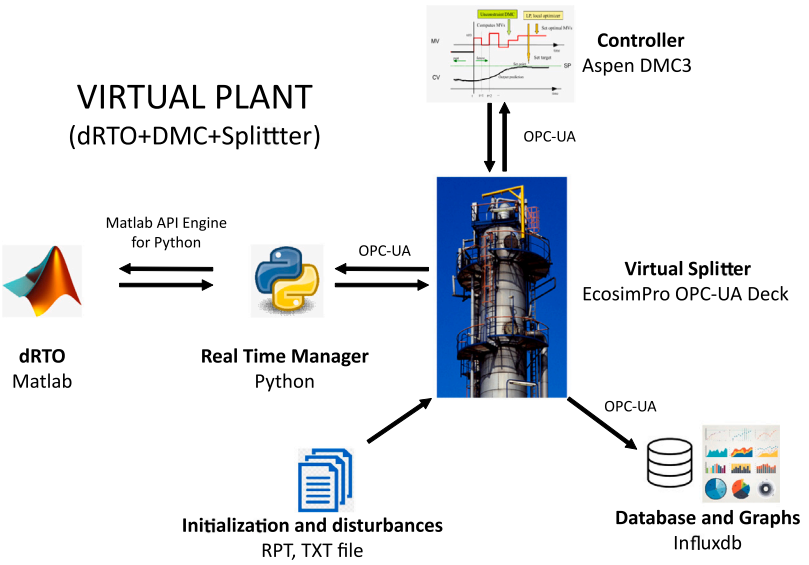


Fig. 19. Virtual Plant in closed loop with Aspen DMC and optimization layer.

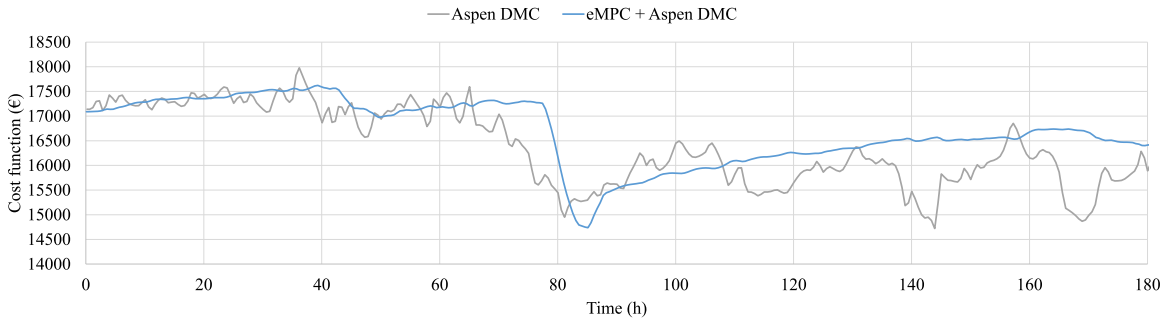


Fig. 20. Cost function.

Table 2
Total profit for each architecture.

Description	Total profit (€)
Aspen DMC	3.023×10^6
dRTO + Aspen DMC	3.095×10^6

The difference in performance in the distillate of the dRTO is justified by the move plan of the MVs shown in Figs. 22 and 23. The dashed black lines represent the constraints for these MVs. The lower and upper bounds of the MVs have been normalized for confidentiality reasons. The dRTO reduces the time of off-spec propylene production by sending the best values for steam and distillate to the Aspen controller at the right time.

As can be seen from the results, the virtual environment developed allows the evaluation of adding an optimization layer to the real process. The obtained result is valid because the simulation uses a phenomenological model as a testbed, which also implies a zero experimental effort. The implementation of the supervisory layer in the real process would be relatively simple, since the developed tool uses similar communication standards as the plant and uses the same variables (type, configuration, etc.) as the real system.

4. Conclusions

We have shown the advantages of developing a virtual environment of a real industrial plant by simply changing the real process by simulation while maintaining the same control architecture. A real case study has been presented, where a propane–propylene splitter of the Petronor refinery in Muskiz, Northern Spain, has been modeled in EcosimPro software and converted into an OPC UA Server. The OPC UA Server allows interaction by reading/writing variables of the model in the same way as if it were a real

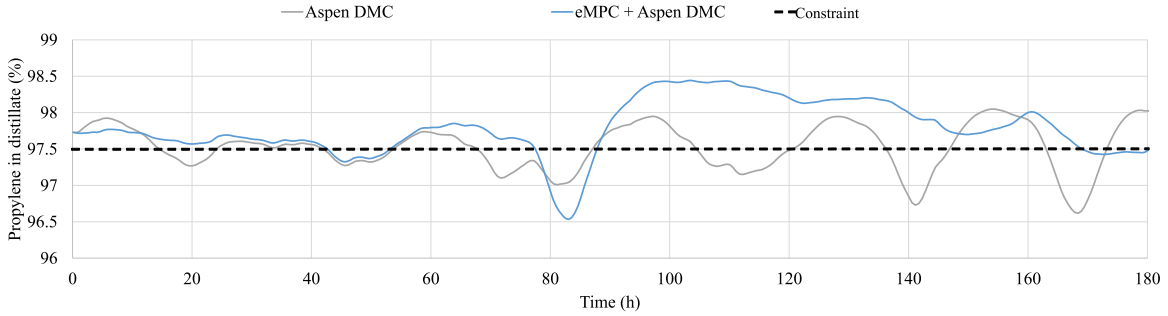


Fig. 21. Quality constraint.

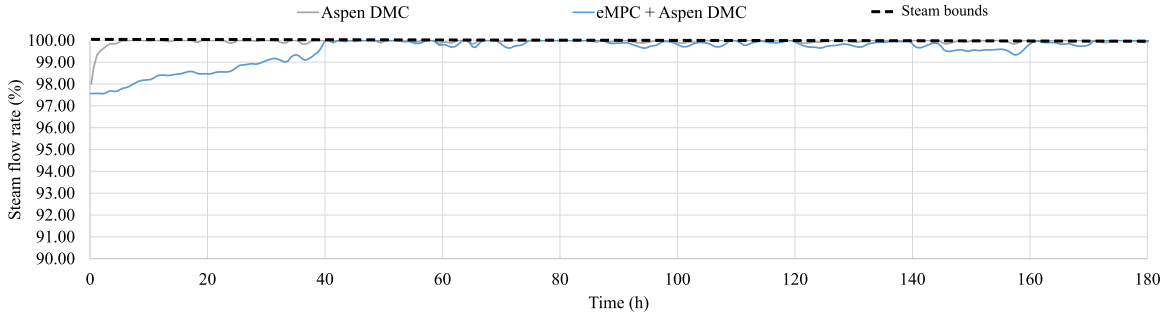


Fig. 22. Steam flowrate.

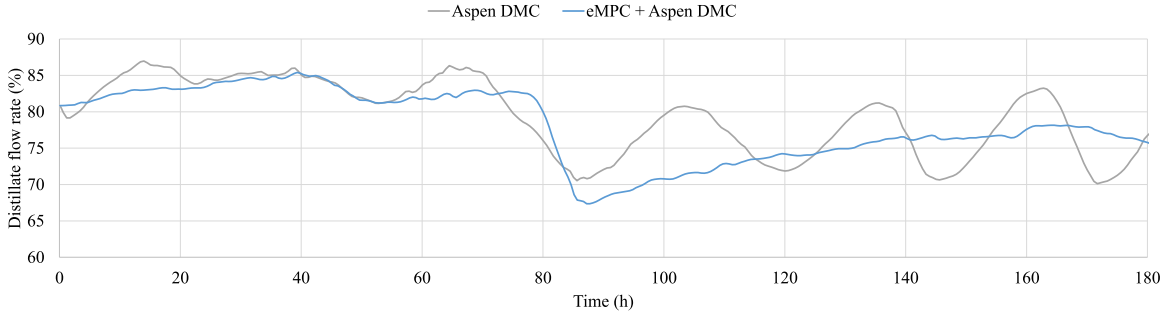


Fig. 23. Distillate flowrate.

system, with the added advantage of speeding up the simulation. The architecture has been completed with the control (DMC) and optimization (dRTO) layers mimicking the same configuration as in the real plant.

It is important to note that the presented architecture can be adapted to any other process by changing the model in EcosimPro and the configurations and model in Aspen DMC. Other controller applications or programming languages can also be used if the OPC UA protocol is supported.

CRediT authorship contribution statement

Erika Oliveira-Silva: Methodology, Software, Investigation, Visualization, Validation, Writing – original draft, Writing – review & editing. **Jesús M. Zamarreño:** Conceptualization, Methodology, Software, Investigation, Visualization, Writing – original draft, Writing – review & editing. **Cesar de Prada:** Conceptualization, Methodology, Supervision, Funding acquisition, Project Administration, Writing – review & editing. **Daniel Navia:** Supervision, Writing – review & editing. **Sergio Marmol:** Methodology, Software, Validation, Writing – review & editing. **Rafael Gonzalez:** Methodology, Validation, Writing – review & editing.

Data availability

The data that has been used is confidential.

Acknowledgment

This work was supported by the Regional Government of Castilla y León and the EU-FEDER (CL-EI-2021-07, UIC 233), as well as with project PID2021-123654OB-C31 (a-CIDit), from MCIN/AEI /10.13039/501100011033/FEDER, UE. The first author thanks the European Social Fund and the "Consejería de Educación de la Junta de Castilla y León". All the authors would also like to thank the management of Petronor for their commitment and help.

References

[1] OPC Foundation, 2013, URL: http://www.opcfoundation.org/. Online; accessed 25-February-2013.
[2] J.M. Zamarreño, L.F. Acebes, R. Alvés, OPC-based real time simulator: architecture and practical example, in: 41st SIMS Simulation Conference (SIMS'2000), Technical University of Denmark, DTU, Lyngby, Denmark, 2000.
[3] J.M. Zamarreño, Entorno de comunicaciones OPC para ecosim, in: Primeras Jornadas de Usuarios de EcosimPro (2001), UNED, Madrid, Spain, 2001.
[4] L.F. Acebes, R. Alves, A. Merino, C. de Prada, Un entorno de modelado inteligente y simulación distribuida de plantas de proceso, Rev. Iberoam. Autom. Inform. Ind. (RIAII) 1 (2) (2004) 42–48, URL: http://recyt.fecyt.es/index.php/RIAII/article/view/10611/7418.
[5] R. Alvés, J.E. Normey-Rico, A. Merino, L.F. Acebes, C. de Prada, OPC based distributed real time simulation of complex continuous processes, Simul. Model. Pract. Theory 13 (7) (2005) 525–549, http://dx.doi.org/10.1016/j.simpat.2005.01.005, URL: http://www.sciencedirect.com/science/article/pii/S1569190X05000171.
[6] J.M. Zamarreño, R. Mazaeda, J.A. Caminero, A.J. Rivero, J.C. Arroyo, A new plug-in for the creation of OPC servers based on EcosimPro© simulation software, Simul. Model. Pract. Theory 40 (2014) 86–94, http://dx.doi.org/10.1016/j.simpat.2013.09.003.
[7] EcosimPro, 2013, URL: http://www.ecosimpro.com/. Online; accessed 25-February-2013.
[8] Python, 2022, https://www.python.org/. Online; accessed 27-July-2022.
[9] E.F. Camacho, C. Bordons, Model Predictive Control, Springer London, 2007, http://dx.doi.org/10.1007/978-0-85729-398-5.
[10] C. de Prada, D. Sarabia, G. Gutierrez, E. Gomez, S. Marmol, M. Sola, C. Pascual, R. Gonzalez, Integration of RTO and MPC in the hydrogen network of a petrol refinery, Processes 5 (2017) 1–20, http://dx.doi.org/10.3390/pr5010003.
[11] Aspen Technology Inc, Aspen DMC3, 2021, https://www.aspentech.com/en/products/msc/aspen-dmc3. Online; accessed 21-February-2023.
[12] M. Ellis, H. Durand, P.D. Christofides, A tutorial review of economic model predictive control methods, J. Process Control 24 (8) (2014) 1156–1178, http://dx.doi.org/10.1016/j.jprocont.2014.03.010.
[13] W.L. Luyben, Practical Distillation Control, Springer, 2012.
[14] E. Oliveira-Silva, C. de Prada, D. Navia, Simulation platform of an industrial propylene-propane splitter integrated to advanced process control for real time optimization experiments, IFAC-PapersOnLine 55 (7) (2022) 673–678, http://dx.doi.org/10.1016/j.ifacol.2022.07.521.
[15] A.C. Hindmarsh, P.N. Brown, K.E. Grant, S.L. Lee, R. Serban, D.E. Shumaker, C.S. Woodward, SUNDIALS, ACM Trans. Math. Softw. 31 (3) (2005) 363–396, http://dx.doi.org/10.1145/1089014.1089020.
[16] E. Oliveira-Silva, C. de Prada, D. Navia, Economic MPC with modifier adaptation using transient measurements, in: 31st European Symposium on Computer Aided Process Engineering, Elsevier, 2021, pp. 1253–1258, http://dx.doi.org/10.1016/b978-0-323-88506-5.50193-5.
[17] E. Oliveira-Silva, C. de Prada, D. Navia, Dynamic optimization integrating modifier adaptation using transient measurements, Comput. Chem. Eng. 149 (2021) 107282, http://dx.doi.org/10.1016/j.compchemeng.2021.107282.
[18] E. Oliveira-Silva, C. de Prada, D. Montes, D. Navia, Economic MPC with modifier adaptation using transient measurements, Comput. Chem. Eng. (2023) 108205, http://dx.doi.org/10.1016/j.compchemeng.2023.108205.
[19] InfluxData Inc, Influxdb, 2021, https://www.influxdata.com/. Online; accessed 21-February-2023.