

Easy-to-use MPC tool for controlling chemical processes in a rigorous simulation environment[★]

Marco Vaccari^{*} Riccardo Bacci di Capaci^{*,**} Alberto Busoni^{*}
Gabriele Pannocchia^{*}

^{*} *Department of Civil and Industrial Engineering, University of Pisa, Italy*

^{**} *Corresponding author: riccardo.bacci@unipi.it*

Abstract: Rigorous process simulation has become a tool that academic and industrial environments are exploiting, mainly to extract information useful for maximizing profit. As a matter of fact, the detailed thermodynamic models contained in commercial or open-source software are able to represent the behavior of a chemical process far better than a linearized model. On the other hand, designing customized model predictive controllers (MPC) has proven to enhance process performance over traditional control architectures. Therefore, in this paper, we present the interaction of an easy-to-use MPC algorithm developed in Python with the rigorous simulator UniSim Design[®]. The communication exploits the UniSim Design[®] spreadsheets as the variables database to be read/written by Python, by stopping or not the simulation before every control action. The software communication has been properly developed so to maintain the flexibility of the original MPC code and to exploit different controller designs. Two different test cases are presented to show the effectiveness of the proposed methodology: a simple two-phase separator and a more complex debutanizer column. System identification is used to build the controller's linear models, various MPC designs differing in considering disturbances as measurable have been analyzed and satisfactory results are obtained.

Keywords: Process modeling; process control; rigorous simulation; system identification; HYSYS/Unisim; Python

1. INTRODUCTION

Advanced process simulation provides the ability to model complex chemical systems by using accurate thermodynamic models to analyze their behavior (Vaccari et al., 2023, 2020). Research accomplishments in process simulation, conceptual design, process control, operations, and optimization has led to useful methodologies and software tools for industry (Grossmann and Harjunkski, 2019). For such reasons, simulation-based optimization is also taking an important role in the evolution towards Industry 4.0 by enabling the development of detailed *digital twins* (Alrabghi, 2018).

On the other hand, custom designs of advanced controllers, such as model predictive control (MPC), have shown improvements over traditional architectures in different fields (Bacci di Capaci et al., 2019), even for rigorously simulated processes (Halager et al., 2021). A recent comparative study (Bartolome and Van Gerven, 2022) demonstrated various ways to connect custom MPC algorithms and simulated processes in HYSYS[®] using popular languages such as Microsoft Excel (VBA), MATLAB, Python, and Unity (C#). Tuan et al. (2016) applied the control of a depropanizer column in HYSYS[®] enabling the communication with MATLAB; Shin et al. (2020) in the same framework controlled a linear surrogate model built via an artificial neural network. The interaction MATLAB-HYSYS[®] has been widely used in different works to develop specific control actions (Ahmadgurabi et al., 2010; Mitchell et al., 2017). Other examples of MPC controllers applied to

chemical systems rigorously simulated used an Excel-VBA-HYSYS[®] link to improve the performances of two natural gas liquefaction processes (Fazlollahi et al., 2016), while Oravec et al. (2017) used Profit Design Studio software to control a depropanizer unit in UniSim Design[®].

Given this framework, the present paper aims at developing a communication link between an MPC algorithm developed in Python (Vaccari and Pannocchia, 2016) with the commercial process simulation software UniSim Design[®]. This solution enhances the possibility of testing the developed MPC code and, at the same time, allows one to study custom MPC algorithms on complex chemical systems that are often difficult to linearize. The rest of the paper is organized as follows: Section 2 describes the tools and methodologies used for the aforementioned control and communication; Section 3 shows the application to two case studies, a phase separator, and a debutanizer column; finally, Section 4 concludes the work.

2. TOOLS AND METHODS

2.1 MPC code

The code in which the MPC algorithm is designed and simulated has been developed in Python language, under the name of `MPCcode` (see Vaccari and Pannocchia (2016) for implementation details). The MPC framework is general and versatile, giving the possibility to easily customize the controller options, and it is adaptable to problems in different industrial fields, e.g., process control, and robotics. Various options are available to define the different blocks of the standard MPC algorithms: state estimation, target, and optimal control problems. Different

[★] This research was supported by Horizon Europe through the funding program "FrontSeat, Project 101079342"

MPC simulations can be easily studied with `MPCcode`, including linear and non-linear, as well as discrete or continuous time formulations. In addition to the standard tracking MPC problems with a quadratic objective function, there is also the possibility to implement any (linear or nonlinear) function to represent other MPC structures, e.g., to consider economic MPC. In all cases, there is the opportunity to activate the “offset-free” design option, that is, to implement a linear or non-linear disturbance model, and an associated augmented observer (Pannocchia, 2015). The `MPCcode` is structured in modules for easy customization of the MPC algorithm desired, and for a complete system simulation, also the controlled process is to be simulated in the code. Nevertheless, as anticipated, the use of rigorous process simulation tools can exploit the behavior of complex models of a chemical process that a surrogate model in Python must simplify, but that can be important for the control action to be calculated. Therefore combining the flexibility of the `MPCcode` with the rigorous models of simulation tools can represent a plus for researchers but also for industrial realities.

2.2 UniSim Design - Python communication

The communication between the process simulator and the Python environment has been carried out via a COM interface. The “Python for Windows Extensions” package, known as `pywin32`, allows one to easily access the Windows Component Object Model (COM) - a binary-interface standard for software components, and control Microsoft applications via the `win32com` library. It is used to enable inter-process communication in a large range of programming languages by exploiting a language-neutral object implementation to be used in environments different from where they were created. In this paper, the process simulator used for the communication with the `MPCcode` is UniSim Design® and it is considered as an *automation* server which provides services to be explored by the Python client (Mounaam et al., 2020). Automating UniSim Design® gives the possibility to run several simulations in a much shorter time, by allowing also easier data collection and processing. Moreover, for our purposes, advanced control algorithms can be directly applied to the process simulation. The communication requires a few code lines in Python and a simulation file open and active to access its objects. All these operations can be carried out individually, but in our case, they have been included in a script function in `MPCcode`. When the communication is established, several UniSim Design® objects can be accessed, e.g. unit operations and their parameters, streams, controllers and spreadsheets, but also simulation tools as the *Integrator* and its functionalities. The *Integrator* is used to set and run the dynamic simulations, and its options can be modified directly from Python, e.g., time can be started and stopped, the integration step can be modified, and real-time mode can be activated. Nevertheless, to simplify the interface and facilitate modifications, a spreadsheet in the simulation environment is used for communication. Accessing via `MPCcode` this spreadsheet, whose cells are connected to the process parameters of interest, gives the possibility to read simulation data and to write new values iteratively along with the dynamic simulation in UniSim Design®. Process outputs are recorded in Python from UniSim Design® and MPC outputs, that is, process inputs, are sent back to the simulator, recursively at any evaluation step. When working with MIMO processes, many inputs and/or outputs are required, therefore, suitable functions have been developed to read/set entire columns in the simulator spreadsheet. Following the structure of `MPCcode`, all these steps are

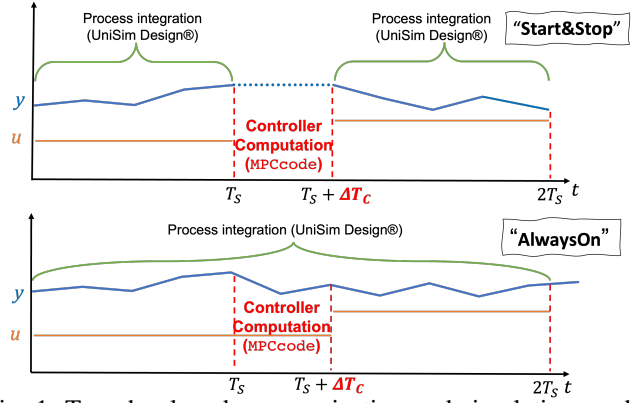


Fig. 1. Two developed communication and simulation modes between UniSim Design® and `MPCcode`.

included in a separate file named *Unisim.Communication*. In general, communication with UniSim Design® spreadsheet can be used both in steady-state and in dynamic mode; nevertheless, MPC needs a dynamic simulation since this is the only one that can give a transient response to the control action calculated.

2.3 Communication and simulation modes

The *Integrator* in UniSim Design® can be active or not during the communication, that is, while `MPCcode` is running. Two different approaches are thus tested in this work as represented in Figure 1. The first is the “Start&Stop” mode in which the *Integrator*, that is, the process simulation, is stopped during MPC elaborations and then restarted for the established period of time (ΔT_S) only once MPC calculations have been imported, i.e., the operator point of view is inside the MPC algorithm. On the other hand, when the dynamic simulation and the *Integrator* are always active during the whole communication process (“AlwaysOn” mode), the operator point of view is within the simulation that is going on also while the MPC calculation is performed. In this scenario, MPC suffers from time delay due to the computational cost of the algorithm (ΔT_C), as in real industrial applications. As in Figure 1, in both modes the output y is registered always every sample time (T_S), the controller action u is calculated for y_{T_S} , and its computation time is always equal to ΔT_C , that is, the controller action is sent back to UniSim Design® at $T_S + \Delta T_C$. In “Start&Stop” mode $y_{T_S} = y_{T_S + \Delta T_C}$, thus representing the ideal situation in which $\Delta T_C = 0$, while for the “AlwaysOn” mode the action computed based on y_{T_S} is applied to a system that has already reached $y_{T_S + \Delta T_C}$, causing the time delay mentioned above. In addition, the absolute time recorded in the UniSim Design® *Integrator* is always shifted by a quantity equal to the integration step with respect to the one in Python. This mismatch is easily solvable taking into account the step of integration of UniSim Design® when recording values in `MPCcode`: during the integrator calculation, process values indeed do not change.

3. CASE STUDIES

The developed communication tools are first tested on a simple two-phase separator. The second case study (a debutanizer column) is instead used to compare different MPC designs.

3.1 Two-phase separator

The simulated process comprises the separation of light components of a hydrocarbon mixture in a two-phase separator. A

Table 1. Features of feed stream (Brambilla, 2014).

Conditions			
Pressure [kPa]	150	Vapor fraction	0.406
Temperature [°C]	70	Mass flow [kg/h]	1000
Composition (mole fractions)			
Ethane	0.02	Propane	0.10
n-Butane	0.15	n-Pentane	0.15
n-Hexane	0.15	n-Heptane	0.19
n-Octane	0.24		

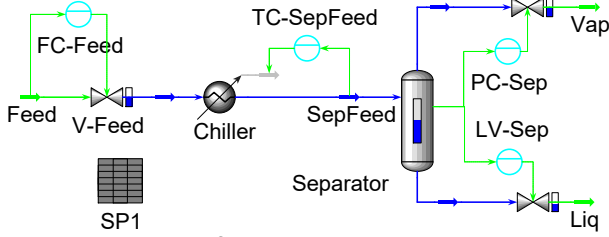


Fig. 2. Unisim Design® PFD of the two-phase separator.

linear MPC is derived based on a SISO model identified by simple step tests. The process aim is to separate the lightest fractions, i.e., from n-butane to ethane (Figure 2). Before the vessel, the feed is cooled to condensate heavy components and obtain the desired composition at the separator vapor outlet, that is, a total molar fraction of light components equal to 90%. The feed composition is reported in Table 1. At first, the dynamic simulation is carried out by using material streams, a chiller and a separator; the Peng-Robinson fluid package is adopted. A variation in the separator feed temperature results in a variation of the vapor outlet, both in composition and in flow rate. Valves are located on the process boundaries streams to calculate the pressure profile of the entire process; valves and units are designed with suitable dimensions, as they strongly affect process dynamics. Four PI regulators with fixed tuning parameters ($K_c = 1$, $\tau_i = 1\text{min}$) are adopted:

- (1) **FC-Feed**: provides disturbance rejection and set-point tracking ($SP = 1000\text{ kg/h}$); the control variable is the mass flow rate while valve position is the manipulated one;
- (2) **TC-SepFeed**: manipulates the chiller energy stream to track the set-point on feed temperature (20 °C);
- (3) **PC-Sep**: manipulates the valve position on the vapor outlet to keep the pressure at its set-point (125 kPa);
- (4) **LV-Sep**: regulates the valve position of the liquid outlet to keep the vessel level at its set-point (50%).

The MPC solution is implemented to manipulate the set-point of TC-SepFeed to follow a purity specification, that is a maximum n-hexane mass fraction in the vapor outlet of 0.035. The spreadsheet SP1 is employed to exchange all major process parameters and PI set-points. The input is the TC-SepFeed set-point; the output is the mass fraction of n-hexane in the vapor outlet (Vap in Figure 2). A first order plus time delay transfer function is adopted as SISO model for the system. A non-measurable disturbance is added to the inlet stream temperature and simulated as a random walk variation from its steady-state value. The transfer function is obtained with a graphic approach from the step response. In particular, the evaluated parameters are the static gain $K = \frac{\Delta y}{\Delta x} = 0.001705$ and the time constant $\tau = 19.93\text{ min}$, subject to $y_{\theta+\tau} = 0.63y_{ss}$, where $\theta = 2T_I$, that is, two integration time steps in UniSim Design® are considered as time delay. The model is then converted into a discrete-time state-space formulation, augmenting states number from 1 to 3 to include time delay, and implemented in the MPC algorithm

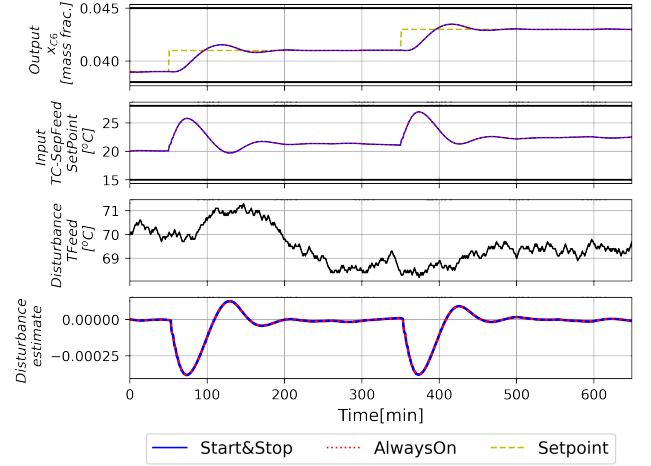


Fig. 3. Separator case study: output, input, disturbance.

with the following parameters: 650 time samples for the whole simulation, a prediction horizon of 50 samples, and a sampling time $T_s = 10\text{ s}$ in the communication. An output disturbance linear model is adopted to obtain offset-free control, i.e., $B_d = 0$ and $C_d = 1$, and Kalman filter is used as a state estimator. Output set-point changes are scheduled over time; while input and state set-points are fixed. Finally, bound constraints and tuning for steady-state and dynamic optimization weight matrices are properly defined. Both “Start&Stop” and “AlwaysOn” modes are tested and compared over the same operating conditions. The MPC proves to be effective and a new steady state is reached after every set-point change despite the disturbance. Figure 3 shows the time trend of input, output, actual disturbance, and disturbance estimate. Results obtained by the two communication modes are comparable since time trends are almost perfectly superimposed; e.g., the RMSE between the two outputs, normalized over the average value for the “Start&Stop” mode, is $1.79 \cdot 10^{-3}$. As expected, when the computational time ($\Delta T_c \approx 13\text{ ms}$) of the MPC algorithm is much smaller than the communication sampling time, the delay effect given by the online use of MPC can be successfully neglected. We stress that this simple case could be easily managed by well-tuned PID controllers and it is here used just to test the communication setup. Further confirmations are to be obtained with larger scale simulations, since for very complex processes, the computational time is higher and the delay effect more impacting.

3.2 Debutanizer column

The second example considered is an industrial debutanizer distillation column with 40 trays. With reference to Figures 4a and 4b, the feed composition is reported together with the other main stream parameters at the steady-state condition in Table 2. The process aim is to split the feed to obtain a top product with low levels of pentane and heavier components and a bottom product with low levels of butane and lighter components. In particular, the impurity targets are a maximum molar composition of n-pentane in the distillate ($x_{C5,D}$) and of n-butane in the bottom ($x_{C4,B}$), respectively, both equal to 0.005. These specifications are the main indicators of good process performance, as in LNG distillation column trains (Luyben, 2013). It is worth underlining that top and bottom temperatures are chosen to require the use of traditional utilities, that is, tower cold water and medium-pressure steam; for this reason, operative pressure is imposed accordingly. Five PI controllers

Table 2. Debutanizer: steady-state conditions to maintain.

Feature	D	B	Light	Heavy	Feed
Mass Flow [kg/h]	2596	13404	6500	9500	16000
Temperature [°C]	48.55	191.7	100	100	100
Pressure [kPa]	1210	1238	1280	1280	1280
Composition [mole frac]					
Ethane (x_{C2})	0.0747	0.0000	0.0412	0.0000	0.0200
Propane (x_{C3})	0.3735	0.0000	0.2061	0.0000	0.1000
n-Butane (x_{C4})	0.5469	0.0049	0.3092	0.0000	0.1500
n-Pentane (x_{C5})	0.0048	0.2030	0.0844	0.2117	0.1500
n-Hexane (x_{C6})	0.0000	0.2048	0.1210	0.1773	0.1500
n-Heptane (x_{C7})	0.0000	0.2594	0.1218	0.2541	0.1900
n-Octane (x_{C8})	0.0000	0.3277	0.1160	0.3567	0.2400

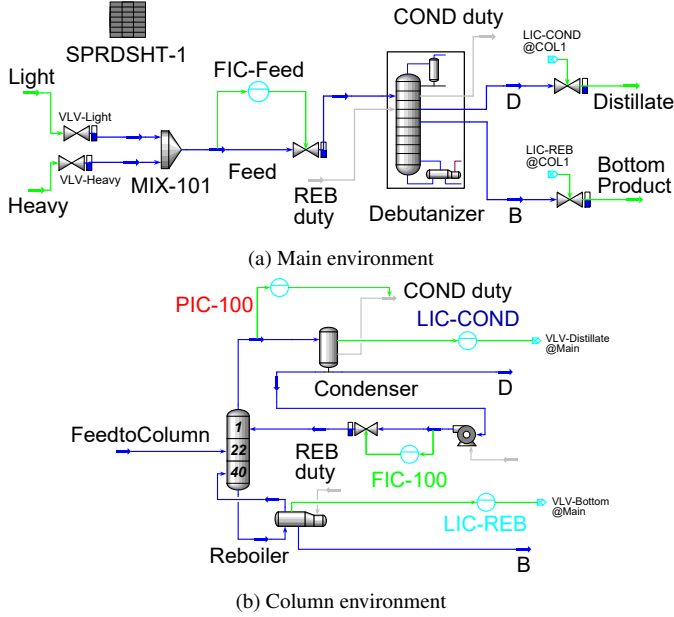


Fig. 4. Dynamic simulation of debutanizer case study.

compose the standard control architecture:

- (1) **FIC-Feed**: provides disturbance rejection and set-point tracking for the column feed flow rate (F_F).
- (2) **PIC-100**: manipulates the cold duty on the condenser to keep column pressure at its set-point.
- (3) **LIC-COND**: regulates the valve position of the distillate stream to keep the condenser liquid level at set-point.
- (4) **LIC-REB**: regulates the valve position of the bottom product to keep the reboiler liquid level at its set-point.
- (5) **FIC-100**: provides disturbance rejection and set-point tracking on the reflux mass flow rate.

The various controllers were tuned both manually and with the help of the autotuning function of UniSim Design[®], after a suitable detuning to avoid possible instabilities. Figure 4a shows how the column feed is obtained by mixing two auxiliary streams, a light and a heavy component one (Table 2). This configuration is used to easily manipulate the feed composition (F_C); e.g., a higher flow rate of the light component stream would result in a lighter composition of the column feed.

System identification and validation. To apply the MPC, a suitable process model is required; system identification is thus performed. Data are collected with “Start&Stop” communication mode. The following parameters are set: a communication sampling time $T_S = 1$ minute; 720 minutes of simulation for the identification stage and 360 for the validation. The main variables classified as manipulated, controlled, and

Table 3. Model variables and their initial condition.

Manipulated	I.C.	Controlled	I.C.
Reflux mass flow SP (R_{SP}) [kg/h]	3630	$x_{C5,D}$ [mole frac.]	0.0048
Pressure SP (P_{SP}) [kPa]	1200	$x_{C4,B}$ [mole frac.]	0.0049
Reboiler duty (R_P) [kW]	1486	Distillate valve [%]	50
Disturbances		Bottom valve [%]	50
Feed mass flow rate (F_F) [kg/h]	16000	Reflux valve [%]	50
Feed composition (F_C) [%]	50	Condenser valve [%]	50

disturbances, are shown in Table 3, together with their initial conditions (I.C.), which represent a steady-state point for the system simulation. The reflux mass flow set-point is the first manipulated, with a key role in the column top equilibrium: the higher the reflux, the higher its cooling effect in the first stage. Moreover, it has an economical implication, because higher reflux, means higher thermal power to be provided to the condenser and reboiler. The column pressure set-point is the second manipulated, since it strongly affects the equilibrium equations along the column. The third manipulated is the reboiler duty: thermal power at the reboiler strongly affects bottom composition. Disturbances, on F_F and F_C , are introduced by varying the set-point of FIC-Feed and the valve opening of the auxiliary light stream (VLV-Light in Figure 4a), respectively. First, two controlled variables are $x_{C5,D}$ and $x_{C4,B}$; as anticipated, their target is to be always below 0.005. The other four controlled variables are the opening percentage of four key control valves, i.e., on distillate and bottom product, on the reflux, and on the condenser, that is, on the cold utility. Valves must operate in the range 5 – 95% to avoid saturation conditions. A suitable data campaign was performed to collect input and output data. To be informative enough, manipulated inputs were modeled as GBN signals, with a switching probability $\delta = 0.1$, and different amplitudes: ± 30 kg/h for R_{SP} , ± 5 kPa for P_{SP} , ± 5 kW for R_P . Every input was firstly designed independently to register the same impact on process outputs. Two disturbances were modeled as random walk signals, with a standard deviation $\sigma = 5$ for F_F and $\sigma = 0.15$ for F_C . Outputs were verified to have the same variation range in the identification and validation sets.

Model identification was achieved by using the SIPPY package (Armenise et al., 2018). All data were centered using their initial value; different models and methods were tested: 2 input-output models (ARX, ARMAX), and state-space models with 4 different subspace methods (N4SID, MOESP, CVA, PARSIM-K). Subspace methods have been applied in innovation form, so that, a Kalman filter matrix is adopted. The Akaike information criterion (AIC) was used to find the best model order, in terms of number of states, in the range $[6 \div 20]$. All methods produced extremely accurate results and outputs trends were almost superimposed. Further details are here omitted for the sake of brevity. Validation was then performed on a different data set, using state-space models in their process form, while input-output models were in predictor form. Model performance was assessed by the explained variance, defined as:

$$EV = 1 - \frac{\frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2}{\sigma_y^2} \quad (1)$$

where N is the number of time samples, \hat{y} is the model output, $\sigma_y^2 = \frac{1}{N} \sum_{i=1}^N (\bar{y} - y_i)^2$ is the variance of the actual output y , with respect to its mean value \bar{y} . Mean values of EV obtained for the validation dataset are reported in Table 4. Note that values of EV higher than 0.90 were obtained for almost all the methods for all outputs. Finally, the 17-states state-space model obtained

Table 4. Model performance in validation.

	ARX	ARMAX	PARSIM-K	CVA	MOESP	N4SID
\overline{EV}	0.9641	0.9642	0.9025	0.9468	0.9156	0.9387

with the CVA method was selected since it gave the best results in terms of mean explained variance and model stability.

MPC design. Once identified and validated, the process model is implemented into the MPCcode and the communication mode adopted is “Start&Stop”. The general model considered for the MPC is as follows:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + B_d d_k + B_m d_{m,k} \\ d_{k+1} = d_k \\ y_k = Cx_k + C_d d_k \end{cases} \quad (2)$$

in which $d \in \mathbb{R}^6$ is the so-called *disturbance*, $x \in \mathbb{R}^{17}$ are the states, $u \in \mathbb{R}^3$ the inputs, and $y \in \mathbb{R}^6$ the controlled outputs. Measurable disturbances (d_m) can be included in the MPC model to improve the prediction of the system behavior and the control performance. Thus, the different MPC formulations tested to investigate the effect of measurable disturbances are:

- MPC-0: no measurable disturbances;
- MPC-1: only feed flow rate (F_F) is measurable;
- MPC-2: F_F and feed concentration (F_C) are measurable.

The disturbances studied are two step changes introduced simultaneously at $t = 50$ min for 700 minutes on F_F (+100 kg/h) and on F_C (+1% on the opening of light stream valve). An output disturbance model is adopted, that is, $B_d = 0$ and $C_d = I$, which allows offset-free behavior despite the process/model mismatch. Initial values for the MPC simulation are those in Table 3. As initial and linearization values for the states, a null vector and $C^{-1}y_0$ have been chosen. A Kalman filter has been adopted in the estimator module. The aim of the controller is to reject the disturbances and maintain as much as possible the process at the initial steady state. Obviously, this is not possible in case of random disturbances, since, a new target is evaluated at every sampling time from the stationary module. Input and output constraints were chosen as a compromise between output specification to be fulfilled and acceptable input variation range to impose. To avoid computational caveats (unfeasibility issues) in the optimal control problem, the upper bounds of the impurities were imposed higher than corresponding steady-state ones. Standard quadratic design is used for both optimization modules. The tuning matrices are as follows:

- output steady-state, $Q_{ss} = \text{diag}(10^2, 10^2, 0, 0, 0, 0)$. Being the impurities control the main objective, the only non-zero elements are the first two. Indeed, for the four valves position, just the respect of the bounds is considered.
- input steady-state, $R_{ss} = 0$: a null matrix is used since input set-point tracking is not considered a priority.
- states dynamic, $Q = C^T Q_y C + 10^{-6}I$, where $Q_y = \text{diag}(10^6, 10^6, 0, 0, 0, 0)$ is the output dynamic one. In this way, Q is a full matrix; the identity term is added to avoid malconditioning problems deriving from null terms of Q_y .
- input variation dynamic, $S = \text{diag}(10^{-4}, 10^{-4}, 10^{-4})$, values are small, since priority is given to output targets.

Result discussion. The mean computation time ($\overline{\Delta T_c}$) to evaluate the optimal control input and deliver it back into UniSim Design® is $\simeq 83$ ms, with a minimum value of $\simeq 70$ ms and a worst case of $\simeq 143$ ms; simulations are performed on a Windows 10 Pro x64, CPU 2.9 GHz i7, 16GB DDR3. Figure 5 shows the trends resulting from the application of the three

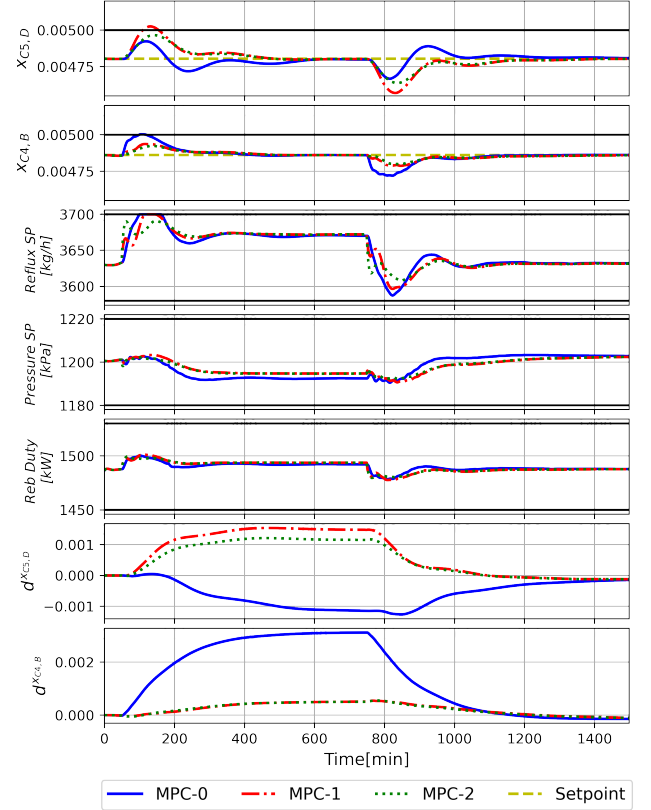


Fig. 5. Variables time trend for the debutanizer: first, two controlled $x_{C5,D}$ and $x_{C4,B}$ (top), the three manipulated (middle), and the disturbance estimates on $x_{C5,D}$ and $x_{C4,B}$ (bottom). The black horizontal lines are the bound constraints.

MPC formulations studied. The only one that does not violate or activate any constraints is MPC-2. It can be seen that $x_{C4,B}$ does not respect the upper bound (at $t \simeq 120$ min) for MPC-0 meaning that the nature and the intensity of the disturbances are capable to worsen the controller performance. On the other hand, also $x_{C5,D}$ overcomes the upper threshold when only F_F is considered measurable (MPC-1). Explanations for this behavior could rely on a not accurate model dynamics for the disturbance on F_F . As a matter of fact, the second-last panel of Figure 5 shows that $d^{x_{C5,D}}$ for MPC-1 and MPC-2 has a completely different trend with respect to the one for MPC-0, because it has to assume complete different values to balance the measurable disturbance being overestimated. When only F_F is considered measurable (MPC-1), $d^{x_{C5,D}}$ increases even more and causes the controller performances to worsen until failing the output upper bound. For a numerical comparison, the following cost function has been used as a metric of error:

$$V_{TOT} = \sum_k^{N_{sim}} \|y_k - y_{sp}\|_{Q_y}^2 + \sum_k^{N_{sim}-1} \|u_{k+1} - u_k\|_S^2 \quad (3)$$

This is a minimized expression that takes into account the controlled output differences with the set-point (y_{sp}) and the input variations. To study how every term of Eq. 3 changes with the different controllers, the corresponding square norm is reported in Table 5 as $V(i)$, where i is a specific output or input. V_{TOT} provides a general idea of the performance of the three MPCs; this metric is expected to get smaller if measurable disturbance dynamics are considered. MPC-2 is indeed the best solution; nevertheless, the result for MPC-1 is almost double that of MPC-2 and even worse than MPC-0. The reason seems to be poor modeling of F_F which may depend

Table 5. Numerical comparison of three studied MPC.

	MPC-0	MPC-1	MPC-2
$V(x_{C5,D})$	$3.04 \cdot 10^{-6}$	$8.45 \cdot 10^{-6}$	$5.00 \cdot 10^{-6}$
$V(x_{C4,B})$	$3.60 \cdot 10^{-6}$	$1.11 \cdot 10^{-6}$	$0.79 \cdot 10^{-6}$
$V(R_{SP})$	296.24	483.85	841.57
$V(P_{SP})$	6.22	6.42	8.46
$V(R_P)$	19.42	44.07	36.16
V_{TOT}	6.64	9.56	5.81

on the identification method adopted, but also on a possibly misleading data collection campaign. Further analysis will be carefully developed in future works to improve such results by also testing MPC-1 behavior when considering the effect of the two active disturbances separately.

4. CONCLUSIONS

In this paper, a complete virtual procedure for the investigation of a custom MPC algorithm developed in Python is presented. The controlled process is dynamically simulated via a rigorous thermodynamic model in a well-established commercial software: UniSim Design®. The communication between the two software has been carried out successfully with two different modes: “Start&Stop” and “AlwaysOn”. This allows the development of custom MPC algorithms and their testing on highly complex controlled processes. The framework was tested on two cases (SISO and MIMO). For the MIMO process, a model has been identified and validated using system identification on the basis of simulation data. Different MPC formulations have been investigated by considering or not the measurable disturbance within the controller design. As results, including the disturbance dynamics in MPC can improve substantially its performance. As more measurable disturbances are included in the MPC predictions, the performances are better, but the type and level of the disturbances may affect these improvements. Next steps will include system identification performed in closed-loop mode, that is, with cascade controllers on top and bottom impurities. Another aspect to be investigated is the computational burden of a large rigorous simulation to be controlled by using the “AlwaysOn” communication mode, eventually evaluating the limiting factors. In such cases, possible advantages could rely on separating control and prediction horizons (e.g. with a moving block strategy). Moreover, different types of process disturbances (e.g., random walk) and enhanced predictive controllers will be tested; in particular, both economic and nonlinear MPC are to be applied to evaluate the effectiveness on complex processes of algorithms recently proposed in the literature (Vaccari et al., 2021).

REFERENCES

- Ahmadgurabi, R.S., Nekoui, M.A., and Salahshoor, K. (2010). Design and implementation of an adaptive predictive controller for a nonlinear dynamic industrial plant using HYSYS and MATLAB simulation packages. In *ICCAS 2010*, 227–230. IEEE.
- Arabghy, A. (2018). Simulation based optimization frameworks as key enablers for the transformation to industry 4.0. In *Proceedings of International Conference on Computers and Industrial Engineering, CIE*, volume 2018-December.
- Armenise, G., Vaccari, M., Bacci di Capaci, R., and Pannocchia, G. (2018). An open-source system identification package for multivariable processes. In *2018 UKACC 12th Int. Conference on Control (CONTROL)*, 152–157. IEEE.
- Bacci di Capaci, R., Vaccari, M., Scali, C., and Pannocchia, G. (2019). Enhancing MPC formulations by identification and estimation of valve stiction. *J. Process Control*, 81, 31–39.
- Bartolome, P.S. and Van Gerven, T. (2022). A comparative study on Aspen HYSYS interconnection methodologies. *Comput Chem Eng.* 162, 107785.
- Brambilla, A. (2014). *Distillation control and optimization: operation fundamentals through software control*. McGraw-Hill Education.
- Fazlollahi, F., Bown, A., Saeidi, S., Ebrahimzadeh, E., and Baxter, L.L. (2016). Transient natural gas liquefaction process comparison—dynamic heat exchanger under transient changes in flow. *Appl. Therm. Eng.*, 109, 775–788.
- Grossmann, I.E. and Harjunkski, I. (2019). Process systems engineering: Academic and industrial perspectives. *Comput Chem Eng.* 126, 474–484.
- Halager, N.S., Bayer, C., Kirkpatrick, R., Gernaey, K.V., Huusom, J.K., and Udugama, I.A. (2021). Modelling and control of an integrated high purity methanol distillation configuration. *Chem. Eng. Process.*, 169, 108640.
- Luyben, W.L. (2013). Control of a train of distillation columns for the separation of natural gas liquid. *Ind. Eng. Chem. Res.*, 52(31), 10741–10753.
- Mitchell, M., Udugama, I., Currie, J., and Yu, W. (2017). Software integration for online dynamic simulation applications. In *2017 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, 360–364. IEEE.
- Mounaam, A., Chhiti, Y., Souissi, A., Salouhi, M., Harmen, Y., and Khouakhi, M.E. (2020). Simulation and optimization of an industrial sulfuric acid plant with contact process using Python-UniSim Design. In *SIMULTECH 2020*, 64–89. Springer.
- Oravec, J., Bakošová, M., and Artzová, P. (2017). Advanced process control design for a distillation column using UniSim Design. In *2017 21st International Conference on Process Control (PC)*, 303–308. IEEE.
- Pannocchia, G. (2015). Offset-free tracking MPC: A tutorial review and comparison of different formulations. In *ECC*, 527–532.
- Shin, Y., Smith, R., and Hwang, S. (2020). Development of model predictive control system using an artificial neural network: A case study with a distillation column. *J. Clean. Prod.*, 277, 124124.
- Tuan, T.T., Tufa, L.D., Mutalib, M.I.A., and Abdallah, A.F.M. (2016). Control of depropanizer in dynamic HYSYS simulation using MPC in MATLAB-Simulink. *Procedia Eng.*, 148, 1104–1111.
- Vaccari, M., Bonvin, D., Pelagagge, F., and Pannocchia, G. (2021). Offset-free economic MPC based on modifier adaptation: Investigation of several gradient-estimation techniques. *Processes*, 9(5), 901.
- Vaccari, M. and Pannocchia, G. (2016). MPCcode. <https://github.com/CPCLAB-UNIP/MPC-code/wiki>. [Online; accessed 27-March-2023].
- Vaccari, M., Pannocchia, G., Tognotti, L., and Paci, M. (2023). Rigorous simulation of geothermal power plants to evaluate environmental performance of alternative configurations. *Renew. Energy*, 207, 471–483.
- Vaccari, M., Pannocchia, G., Tognotti, L., Paci, M., and Bonciani, R. (2020). A rigorous simulation model of geothermal power plants for emission control. *Appl. Energy*, 263, 114563.