Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-1 (2019) 388–393

# Extension of the do-mpc Development Framework to Real-time Simulation Studies

**Alexandru Tatulea-Codrean,** * **Clemens Lindscheid,** *
**Rafael Farrera-Saldana,** * **Sebastian Engell** *

* *Department of Dynamic Simulations and Control, Faculty of Bio-and Chemical Engineering, TU Dortmund, Dortmund, Germany (e-mail: alexandru.tatulea,clemens.lindscheid,rafael.farrera,sebastian.engell@tu-dortmund.de).*

**Abstract:** Nonlinear Model Predictive Control (NMPC) has become a serious option for industrial applications, with advances in software and algorithms making economics optimizing control suitable even for large scale systems. However, the industrial applications are posing a number of engineering challenges, related to the performance of NMPC in real plant environments. In this paper we address the issue of validating NMPC solutions in a real-time simulation environment, which mimics precisely the behavior of the controlled plant. We propose a validation strategy and describe the simulation framework that was developed in order to support the validation process. The software platform used for this purpose is based on the do-mpc framework, benefiting from its modularity and efficient implementation of NMPC algorithms. The focus of this work is the extension of the software environment to a real-application oriented platform where control scenarios can be simulated in real time. The necessity of the proposed validation strategy is demonstrated for a semi-batch polymerization example, where it is shown that feedback delays have a significant impact on the real-time performance. The necessary steps for a transition from simulation studies to the real application are discussed and a method for improving the NMPC performance is proposed based on the real-time simulation studies.



*Keywords:* NMPC, real-time systems, simulation framework, real-time validation

## 1. INTRODUCTION

Since the industrial breakthrough of MPC in the oil industry of the 70s, which was mainly triggered by the fact that MPC can handle multi-variable processes, MPC has become increasingly popular in the process industry (Qin and Badgwell (2003)), with several companies offering software and support for realizing MPC solutions for industrial processes. Nowadays, model predictive control is also used in fields beyond the process industry, such as power electronics (Vazquez et al. (2014)) or in embedded systems (Ferreau et al. (2016)). The vast majority of MPC applications are based on linear process models, assuming that linear models are sufficient for a good control performance and also that the response time of the underlying optimization algorithms does not cause stability or constraint violation problems.

Nevertheless, process non-linearity cannot always be disregarded, especially when the focus of control is the increase of plant profitability, while satisfying quality and operational constraints (economic NMPC) (Engell (2007)). These advantages of NMPC have been shown in several applications to pilot plants, such as a distillation column (Diehl et al. (2003)), the simulated moving bed process (Küpper and Engell (2007)) or the reactive distillation column (Hasskerl et al. (2018)). Furthermore, concerns about NMPC such as the occurrence of model-plant mismatch have been tackled in the academic literature, e.g. by the development of robustification methods (Lucia et al. (2014)). In support of the aforementioned research, state of the art NMPC can nowadays be implemented with a variety of tools and platforms, e.g. ACADO (Houska et al. (2011)), MUSCOD II (Diehl et al. (2001)), MLI (Frasch et al. (2012)), YANE (Pannek and Grüne (2017)), NMPC Tools (Amrith and Rawlings (2008)), OptoEcon Toolbox (Elixmann et al. (2014)) or do-mpc (Lucia et al. (2017)). While these tools provide state of the art NMPC algorithms and support, rapid development or modular two-layer MPC approaches, none of them directly address the real-application challenges that one faces in an optimization based control structure. Moreover, none of these tools have yet been reported to be used in industrial applications.

When inspecting the available industrial solutions for the implementation of linear MPC (e.g. ASPEN, ABB, Honeywell), one notices the focus on practical aspects. These software tools are modular and easy to configure, and most importantly, they support successive steps for the controller validation, which culminate in Hardware-in-the-Loop simulations, where the control logic and hardware can be tested against a simulated plant. They are designed for the later implementation stages of MPC. In contrast, the academic NMPC platforms focus on the early stages of design, i.e. modelling and optimization solutions. Although each of these tools offers very practical solutions and advantages, none of them addresses the necessary design steps when moving on from theoretical studies to practical applications. In order to ensure that the NMPC application will perform as required, the controller and the NMPC solution should be validated in a real-time simulation environment before they are applied to the real plant.

In this paper we propose a tool which supports the development of real NMPC applications by faithful real-time simulations. The concept of *real-time* means here that we assume hard constraints on the timing of the MPC execution, guaranteeing

2405-8963 
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2019.06.093

that a function is executed within a fixed time interval. This requires an asynchronous software structure and the appropriate tools for carrying out the necessary simulation studies. We propose a new version of the do-mpc platform (Lucia et al., 2017), designed for an easy transition between the classical simulation studies in academia to real-time simulation studies, which are the two main development phases considered in our work. The extended do-mpc platform supports both of these phases, requiring only a small effort for the transition.

## 2. NMPC DESIGN METHOD AND REAL-TIME TESTING

### *2.1 From NMPC controller design to real-time implementation*

A typical procedure for validating a controller design is to follow a predefined validation sequence, which involves different testing environments (Abel and Bollig, 2006). The validation of the control concepts is usually performed in a so-called Model-in-the-Loop environment, where the model and the controller run in the same development software. The simulation of the controlled process is done in a synchronous fashion, where the process model and the controller are executed one after the other in a fixed sequence. One moves on from this phase only after the controller design was proven to be correct. Before the implementation of the controller at the process, an integration test is performed in a so-called Hardware-in-the-Loop environment. In this phase, the controller is tested against a real-time simulation of the process. The controller and the process run independently of each other in an asynchronous fashion.
The approach of dividing the design process in these two major phases, in which the real-time validation plays a substantial role, has been very successful in several engineering fields. Applications can be found especially in the automotive industry (Wilson et al. (1997), von Wiesel et al. (2016)), manufacturing (Smith and Peters (1998)), energy systems (Dufour and Belanger (2013), Vazquez et al. (2014)) and other distributed systems Gonzalez (2013). This approach can also be used for the development of NMPC solutions by separating the design steps into three main phases, as can be seen in Figure 1. Simulations are the main validation tool for the first two phases: preliminary validation of the design of control algorithms can be done in synchronous simulations (classical simulation studies), while the later validation steps must be done in asynchronous fashion (i.e. real-time simulation studies) to identify timing-related constraints of the software. The third and final stage is related to the real plant application and it consists of many engineering tasks that need to be carried out before the control structure from phase 2 can be applied to the process. This last phase is not discussed here, therefore the details are omitted. The following sequence of steps must be taken when designing an NMPC solution with the goal of a real plant application (see also Figure 1 - top):

1) The project must be defined, together with the main improvement that it should provide.
2) The process must be modelled, including models for the actuators and sensors used in the final implementation.
3) The NMPC must be formulated (constraints and cost-function) and the components of the NMPC structure have to be defined.
4) The NMPC solution has to be properly tested and tuned in synchronous simulations. Potential sources of uncertainty or plant-model mismatch should be investigated during this phase. It is recommended to design and validate more than one solution for each module of the NMPC loop (e.g different optimizers or estimators).
5) Real-time specifications must be defined for the NMPC modules and the potential sources for performance degradation have to be analyzed, so that their effects can be investigated in the following asynchronous simulations.
6) The NMPC solution must be split into logical and functional units, so that each unit is a stand-alone module. The functionality of the overall system must be defined.
7) The individual modules must be assembled in a real-time system, supervised by a central coordination unit. The performance of the overall asynchronous architecture must be investigated.
8) The performance of the controlled structure should be checked progressively, starting from the simplest available NMPC modules and continuing with the more demanding ones. The entire setup must be holistically validated.
9) The effects of uncertainty and model-plant mismatch should also be investigated in real-time simulations.
10) The effects of the NMPC tuning must be investigated in asynchronous operation, as changes could be required in order to satisfy the specifications.
11) The transfer to the real plant involves adjustments to the data communications, interfaces and safety routines. The functionality developed so far has to be further integrated with DCS or PLC solutions.
12) Plant validation and real application test runs.

### *2.2 Real-time software environments*

In any conventional NMPC solution, we can identify 3 main interacting elements: the plant (or a real-time simulator), the state (and possibly parameter) estimation scheme and the NMPC controller. The main difference between synchronous simulations which are mainly used in academia and the asynchronous simulations of NMPC algorithms is the sequential, respectively parallel execution of the NMPC modules. In the sequential approach one step is simulated and only then a state estimation is performed using the new measurements from the simulation. Finally the optimization is performed using the newly estimated states. The new control moves are passed to the process simulation, where a next step is simulated. In this procedure everything takes place in one thread, the simulation having to wait for the optimizer to finish the calculation of the control moves.
An industrial production unit, however, is continuously running during the calculation of the control moves and the state estimation. When NMPC is based on complex process models, the computation time for the control moves increases and timing-related issues can occur, which can result in significant performance losses. The theoretical results from synchronous simulation studies do not provide information about the asynchronous behavior of the controlled system. Therefore real time testing is necessary and this must be done in an environment that treats the NMPC application as a real-time interconnected set of systems. Additionally, it should be mentioned that often the NMPC controller runs on a different hardware than the automation of the real plant so that additionally the communication between the plant and the NMPC solution has to be taken into account.
To perform asynchronous simulations in a parallel fashion, all modules are initialized first and then executed in parallel, each in a separate process and with its own timer. All modules follow the same execution pattern. They read the current data from

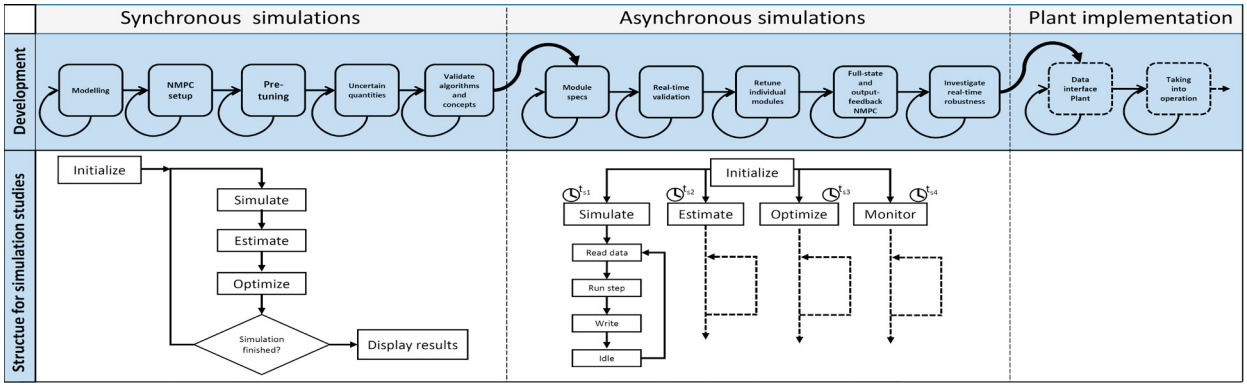


Fig. 1. Development phases of NMPC applications and underlying software models.

a centralized source, perform one step (simulation, estimation, optimization) and write their respective results back. After an idle time, the procedure is repeated again and the whole procedure continues until the execution is terminated by the user. In this configuration, the process simulation is timed usually much faster than the estimation module and the optimization module. In the real application, the sampling time of the Programmable Logic Controller (PLC) or the Distributed Control System (DCS) is also usually much faster than the computation time of the optimal control moves. It should be mentioned that in this concept the estimation and the optimization are separated, which could be coupled in one module with the same sampling time. However, in NMPC applications the optimization step often takes far longer than the estimation step and a shorter sampling time for the estimation is usually better for the performance of the estimator (Hasskerl et al. (2017)).

## 3. THE PROPOSED REAL-TIME STRUCTURE

One can identify the common properties that any real-time simulation environment should have in order to facilitate the design and evaluation of distributed interacting real-time systems:

- The environment must provide tools for the definition of distributed processing and logical units, which form the sub-systems of the overall system.
- The simulation platform must be able to integrate different kinds of modules, with different dynamics and implemented in different programming languages.
- The sub-systems must be able to communicate with each other and with a central coordinator via a specified communication pattern, with clearly defined interfaces and data structures.
- Each sub-system must be functionally independent of the others, updating its own internal states and outputs at the frequency needed for its internal dynamics and compatible with the changes of the inputs specified by other modules.

Based on the requirements presented above, we have expanded the existing *do-mpc* platform for rapid development of NMPC solutions presented in (Lucia et al., 2017) and we have designed a simulation environment which addresses the main aspects of a real-time system. In particular, the timing constraints defined for each of the sub-modules are challenging for the software implementation. The main difference is the way the NMPC modules are executed, i.e. based on parallel software execution and external communication. Parallel software execution means that the modules perform the calculations without influencing each other except at the communication points, external communication means that the modules exchange data with an external software. Figure 2 depicts the structure of the real-time platform, side-by-side with that of the original *do-mpc*. In the following, we explain the functioning of the platform and the main components.

The simulation environment is implemented in Python and only makes use of open source software. It is seamlessly integrated in *do-mpc*. The models used and the algorithms for optimization and estimation are available in *do-mpc*. In order to use them in the real-time platform, the modules are functionally separated from each other and additional functionality is added so that the modules can be executed as stand-alone program units. Doing so, one achieves true parallelism, because the units can be run as different computing processes, or even on different machines. The plant can later be connected, as a physically separated entity, to the real-time platform. After a global initialization, the modules start their execution, each of them having its own prescribed cycle time. The starting and stopping of the modules can be controlled by a central or by a distributed logic or by direct user input. One important additional functionality that needs to be added is the operation logic. Typically, the simulator is started first, determining when the estimator can be started. A central coordinator then decides when the optimizer module can be started based on a timing criterion or on a performance indicator of the estimator.

Since the different modules run independently of each other, the communication between them must be managed by an external module. In the process industry, the Open Platform Communications (OPC) (Mahnke et al. (2009)) is commonly used as a communication standard. For the real-time environment the OPC Unified Architecture (OPC-UA) was chosen, since it is the newest protocol from the OPC Foundation and it provides machine to machine communication without being dependent on Microsoft Windows COM and DCOM technologies. An OPC-UA server acts as the central communication entity and the different modules (clients) can write and read the current data from this OPC-UA server. As this server is typically implemented on the process side of the plant, the communication should be restricted to the exchange of important external information as e.g. optimal control moves, current measurements and estimates as well as status signals. Further data that needs to be logged for debugging purposes is stored in a database, in this case a MongoDB instance. We chose this noSQL database format because it is document oriented, efficient and fast, allowing us to store all outputs of the optimization (e.g. optimizer

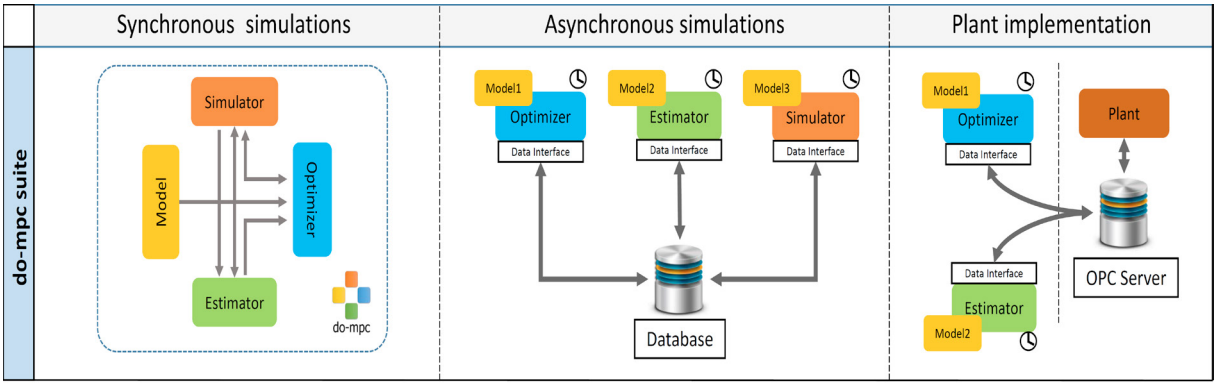


Fig. 2. Three development phases and simulation possibilities with the *do-mpc* platform.

predictions and failure reports, useful for debugging optimizer crashes).

The communication principle is very simple. Each module writes its current data to the server, and the server-side software takes care of storing the data. When a module needs to read data, it will always read the latest available data. For example, the estimator runs at a given frequency and takes the latest process measurements, updating the state estimates in the required fields. Since an estimator often runs at faster rates than the optimizer, it means that the estimates will be updated several times before the optimizer reads them. When an optimization cycle starts, the optimizer module reads the latest available estimates and starts the optimization from the current point. This mode of operation is an accurate representation of the operation at the real plant and it does not require further synchronization. The plant will always run with the latest available inputs, which only change at the rate of the execution of the optimizer.

## 4. ILLUSTRATION: A SEMI-BATCH POLYMERIZATION REACTOR CONTROL PROBLEM

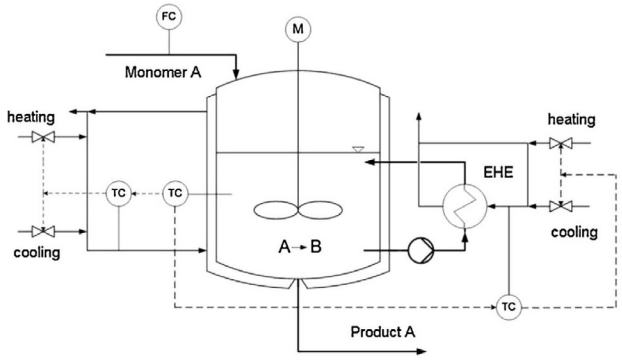


Fig. 3. P & ID of the semi-fed polymerization batch reactor

We present the results obtained with the proposed simulation platform for a semi-batch polymerization reactor depicted in Figure 3. The reactor is equipped with a motor for stirring the contents, a cooling jacket and an external heat exchanger. Monomer is fed into the reactor, where it turns into polymer via an exothermic reaction. The main control challenge is to ensure the polymer quality by keeping the reactor temperature within tight bounds, while maximizing the productivity. In the interest of space we do not reproduce the process model and parameters here, they are given in Lucia et al. (2014).

The resulting process model consists of ten ordinary differential equations (ODEs) and the state vector can be written as $\underline{x} = [m_P, m_A, m_W, T_R, T_S, T_M, T_{EK}, T_{AWT}, T_{adiab}, m_A^{acc}]^T$. These equations represent the mass balances for water, monomer and product hold-ups ($m_W$, $m_A$, $m_P$) and energy balances for the reactor ($T_R$), the vessel ($T_S$), the jacket ($T_M$), the mixture in the external heat exchanger ($T_{EK}$), the coolant leaving the external heat exchanger ($T_{AWT}$) and the adiabatic temperature $T_{adiab}$. The total mass of monomer fed into the reactor is modelled by the state $m_A^{acc}$. The available control inputs are the feed flow $\dot{m}_F$, the coolant temperature at the inlet of the jacket $T_M^{IN}$ and the coolant temperature at the inlet of the external heat exchanger $T_{AWT}^{IN}$. The properties of the resulting polymer are strongly influenced by the temperature at which the process takes place. It is therefore desired to operate the plant within a range of $\pm 2°C$ around a set-point value $T_{R,set} = 90°C$. Additionally, an important safety constraint that must be respected is the adiabatic temperature, which is a virtual temperature that would be reached if all the monomer suddenly reacted. This value must be below $109°C$ throughout the operation.

The control problem being solved here was discussed in detail by Lucia et al. (2014). From an optimization point of view, it is important to remember that we are implementing a simultaneous approach, where the prediction and the optimization parts are performed at the same time by the optimizer module. The nonlinear process model and the previously discussed quality specifications are implemented as constraints on the predictions. Further physical limitations are imposed on the state variables, while the inputs are bounded by operational specifications. The control objective reflects three aspects of optimization: maximization of productivity, reference tracking and input movement penalization. As it will be discussed in the results sections, soft constraints were added for the real-time study.

The typical goal for such a process is to increase the polymer yield, while also satisfying product quality constraints via controlling the reactor temperature. A trade-off exists between feeding more monomer and keeping the adiabatic temperature within bounds, which becomes challenging especially when operating close to the nonlinear constraints.

The synchronous simulation results depicted in Figure 4 serve as a starting point for the evaluation of the real-time implementation, which will be discussed in the next section. The economic optimization criterion results in the process being driven to the constraints of operation. In the first phase (up to around 2 minutes) the feed is increased until the adiabatic temperature constraint is reached (top plot). The feed is then adjusted so that the adiabatic temperature is maintained until around minute

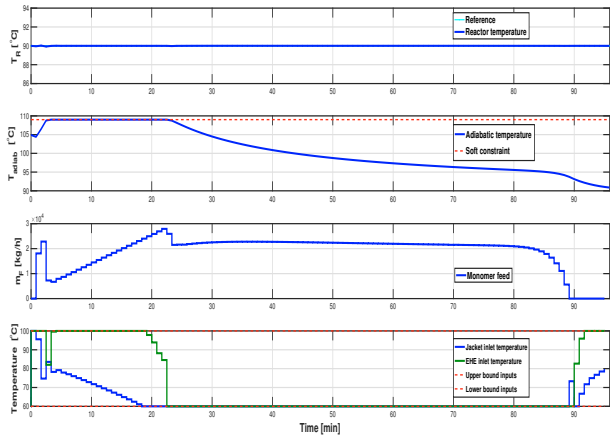


Fig. 4. Synchronous simulation results of the semi-batch reactor

23, when the maximum cooling power is reached. Both the external heat exchanger inlet temperature and the jacket inlet temperature are set to their lower bound of $60°C$. At around 89 minutes the reactor is full and the feed is stopped. Then the holding phase lasts until around 95 minutes when 20680 kg of polymer have been produced and the batch is finished.

## 5. REAL-TIME SIMULATION RESULTS AND DISCUSSION

The transition from synchronous to asynchronous simulations with the proposed *do-mpc* framework is easily possible, based on the procedure sketched in Figure 2. The modules used for the simulations are the same ones that were used for the previous results, but are now equipped with a timing function and communication interfaces, as explained in Section 3. The execution rate of each of the modules was set as follows: the process simulator runs every second, the estimator runs every 2.0 seconds and the optimizer every 50.0 seconds. These values are chosen based on the previous case study, considering typical PLC or DCS sample frequencies in the process industries, which range between 1-10 Hz.

The issue with this kind of optimization scenarios is the way in which the optimizer behaves when the process reaches a point at or very close to the limits of feasibility. In a real process, one cannot accept the control moves that are computed with failures of the optimization routine. We have observed an average time delay for this simulation scenario $\Delta T_{MPC} = 6$ seconds, which is the sum of the computation times for the simulation, estimation and optimization modules and includes also small data communication delays. The combined presence of time delays and hard constraints is not feasible for a real-time implementation. Following common industrial practice, we have substituted the hard constraint on the adiabatic temperature with a soft constraint implementation. The soft constraint was placed at $T_{soft} = 108.5°C$ and a maximum violation of $1.5°C$ was permitted in order to avoid optimization failures. The result of the real-time simulation can be seen in figure 5. As expected, the controller has difficulty balancing the trade-off between the maximization of the economic profit and meeting the soft constraint. Due to the feedback delays present in the real-time operation, this optimization result is not the best operating scenario for the plant, as the batch time increases by 2.9 minutes and the controlled variables become oscillatory in the first 20 minutes of operation.

There are several methods to overcome the feedback delay. Real-time iterations Diehl et al. (2002) are one possible method, where in each step the NMPC problem is firstly set-up using the previous values and then the current estimate is embedded, thus reducing the feedback delay. This method was applied to this case study in (Lindscheid et al., 2016). An alternative solution is to estimate the feedback delay and compute a new initial value for the optimizer, which approximates the process behavior during the calculation of the new control moves. By combining this approach with the use of a soft constraint at $T_{soft} = 109°C$ and a very violation high penalty, the operation of the reactor was improved (see Figure 6). Although it is very similar to the synchronous simulation result, this case still suffers from the presence of a small plant-model mismatch, as reflected in the different feeding and cooling profiles. The achieved batch time in this case is 97.3 minutes, which can be attributed to the inherent time delays and the imperfect control moves. We can conclude that even for this simple example with an ideal estimator and a perfect plant model it is important to investigate

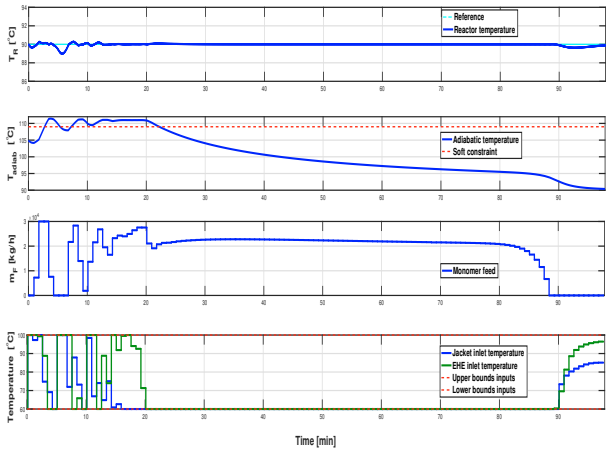


Fig. 5. Initial simulation results with the real-time platform: soft constraint and the effect of time delays

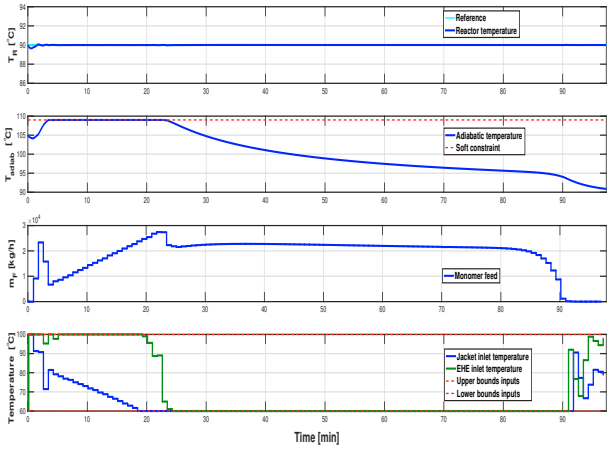


Fig. 6. Final simulation results with the real-time platform: desirable implementation with time delay anticipation

the behavior of the controller in a real-time environment before the application at a real plant.

## 6. CONCLUSION

We have described a procedure for the development of Nonlinear MPC solutions for real processes and we have introduced a platform for real-time simulations which enables the rapid development and validation of the NMPC applications taking into account the effects of the computation times of the different elements. This platform is an extension of *do-mpc* and it is designed in a way that facilitates the transition from standard simulations to asynchronous real-time studies by integrating both simulation modes in an easily configurable software structure. The parallel simulation is based on industry standard communication protocols, facilitating the integration with real processes. The necessity for real-time simulations is highlighted in the case study of a polymerization reactor where it was found that the real-time application raises additional control challenges that do not appear in a classical simulation study. In particular, it was shown that time delays are a potential source of constraint violations and performance losses of the controlled plant, which need to be addressed in the optimization algorithms and the software implementation. We have presented a possible solution to the problem and we will implement the proposed structure at a real laboratory plant.

## REFERENCES


Abel, D. and Bollig, A. (2006). *Rapid Control Prototyping: Methoden und Anwendungen*. Springer.

Amrith, R. and Rawlings, J.B. (2008). Nonlinear model predictive control tools (nmpc tools). URL `https://che.wisc.edu/software/mpctools`.

Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12, 577–585.

Diehl, M., Findeisen, R., and Schwarzkopf, S. (2003). An efficient algorithm for nonlinear model predictive control of large-scale systems part ii: Experimental evaluation for a distillation column. *Automatisierungstechnik*, 1(51).

Diehl, M., Leineweber, D.B., and Schäfer, A. (2001). *MUSCOD-II Users' Manual*. Interdisciplinary Center for Scientific Computing (IWR), Heidelberg.

Dufour, C. and Belanger, J. (2013). On the use of real-time simulation technology in smart grid research and development. *IEEE Transactions on Industry Applications*.

Elixmann, D., Puschke, J., Scheu, H., Schneider, R., Wolf, I., and Marquardt, W. (2014). A software environement for economic nmpc and dynamic real-time optimization of chemical processes. *Automatisierungstechnik*, 62(2).

Engell, S. (2007). Feedback control for optimal process operation. *Journal of Process Control*, 17, 203–219.

Ferreau, H., Almer, S., Peyrl, F., Jerez, J., and Domahidi, A. (2016). Survey of industrial applications of embedded model predictive control. In *2016 European Control Conference (ECC)*.

Frasch, J.V., Wirsching, L., Sager, S., and Bock, H.G. (2012). Mixed-level iteration schemes for nonlinear model predictive control. *IFAC Proceedings Volumes*, 45(17), 138–144.

Gonzalez, F.G. (2013). Real-time simulation and control of large scale distributed discrete event systems. In D.B. C. J.J. Paredis C. Bishop (ed.), *2013 Conference on Systems Engineering Research*, volume 16, 177–186.

Hasskerl, D., Lindscheid, C., Subramanian, S., Diewald, P., Tatulea-Codrean, A., and Engell, S. (2018). Economics optimizing control of a multi-product reactive distillation process under model uncertainty. *Computers & Chemical Engineering*. In press, corrected proof.

Hasskerl, D., Subramanian, S., Hashemi, R., Arshad, M., and Engell, S. (2017). State estimation using a multi-rate particle filter for a reactive distillation column described by a dae model. In *25th Mediterranean Conference on Control and Automation*.

Houska, B., Ferreau, H.J., and Diehl, M. (2011). Acado toolkit - an open source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3), 298–312.

Küpper, A. and Engell, S. (2007). *Non-linear model predictive control of the Hashimoto simulated moving bed process*, chapter Assessment and Future Directions of Nonlinear Model Predictive Control, 473–483. Springer.

Lindscheid, C., Hasskerl, D., Meyer, A., Potschka, A., Bock, H.G., and Engell, S. (2016). Parallelization of modes of the multi-level iteration scheme for nonlinear model-predictive control of an industrial process. In *IEEE Conference on Control Applications*.

Lucia, S., Andersson, J.A.E., Brandt, H., Diehl, M., and Engell, S. (2014). Handling uncertainty in economic nonlinear model predictivecontrol: A comparative case study. *Journal of Process Control*, 24.

Lucia, S., Tatulea-Codrean, A., Schoppmeyer, C., and Engell, S. (2017). Rapid development of nmpc solutions. *Computer Engineering Practice*. URL `https://github.com/do-mpc/do-mpc`.

Mahnke, W., Leitner, S.H., and Damm, M. (2009). *OPC Unified Architecture*. Springer Verlag.

Pannek, J. and Grüne, L. (2017). *Nonlinear model predictive control: Theory and algorithms*. Springer.

Qin, S.J. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*.

Smith, J.S. and Peters, B.A. (1998). Simulation as a decision-making tool for real-time control of flexible manufacturing systems. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, volume 1, 586–590 vol.1.

Vazquez, S., Leon, J.I., Franquelo, L.G., Rodriguez, J., Young, H.A., Marquez, A., and Zanchetta, P. (2014). Model predictive control: A review of its applications in power electronics. *IEEE Industrial Electronics Magazine*.

von Wiesel, D., Talon, V., Thomas, V., Grangier, B., Lansky, L., and Uchanski, M. (2016). Linking model predictive control (mpc) and system simulation tools to support automotive system architecture choices. In *8th European Congress on Embedded Real Time Software and Systems,*.

Wilson, I., Cox, C., French, I., and Fletcher, I. (1997). Real-time simulation is a necessity in the automotive industry. In *IEE Colloquium on System Control Integration and Rapid Prototyping in the Automotive Industry (Digest No. 1997/388)*, 5/1–5/3. doi:10.1049/ic:19971343.