Control Engineering Practice 60 (2017) 51–62

Contents lists available at ScienceDirect

Control Engineering Practice

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

Rapid development of modular and sustainable nonlinear model predictive control solutions

Sergio Lucia[a,b,*], Alexandru Tătulea-Codrean[b], Christian Schoppmeyer[b], Sebastian Engell[b]

[a] Institute for Automation Engineering, Otto-von-Guericke University Magdeburg, Universitätsplatz 2 (Building 07), 39106 Magdeburg, Germany
[b] Process Dynamics and Operations Group, Technische Universität Dortmund, Emil-Figge-Str 70, 44227 Dortmund, Germany

ARTICLE INFO

Keywords:
Nonlinear model predictive control
Robust control
Optimization
Process control
Development support
Numerical methods

ABSTRACT

While computational complexity is often not anymore an obstacle for the application of Nonlinear Model Predictive Control (NMPC), there are still important challenges that prevent NMPC from already being an industrial reality. This paper deals with a critical challenge: the lack of tools that facilitate the sustainable development of robust NMPC solutions. This paper proposes a modularization of the NMPC implementations that facilitates the comparison of different solutions and the transition from simulation to online application. The proposed platform supports the multi-stage robust NMPC approach to deal with uncertainty. Its benefits are demonstrated by experimental results for a laboratory plant.

1. Introduction

Model Predictive Control (MPC) is a popular control strategy that has been successfully applied especially in the process industries as reported e.g. in Qin and Badgwell (2003). The most important reason for this success is its ability to handle coupled multivariable systems with constraints. Its nonlinear variant, Nonlinear Model Predictive Control (NMPC), has been studied intensively by the research community and many simulation results have been published in the last years, including large-scale and highly nonlinear systems based on rigorous models (e.g. Huang, Zavala, & Biegler, 2009; Idris & Engell, 2012 or Toumi & Engell, 2004). Although some companies develop industrial NMPC implementations for some processes (see Cybernetica, 2014; IPCOS, 2014; Pluymers, Ludlage, Ariaans, & Van Brempt, 2008), its practical use is still in the early stages.

Some years ago, one of the main reasons that prevented NMPC from being applied in practice was the computation time needed to solve the resulting large-scale nonlinear programming problems. The progress made in the last years on optimization algorithms and on computational power has made it possible to significantly reduce the computation times needed to solve NMPC problems even to the microsecond range as shown in Houska, Ferreau, and Diehl (2011b). With computational power no longer being a problem in many cases, at least three major challenges remain as main obstacles for the application of NMPC to real processes: The high cost of the development of models, the lack of tools that provide a rapid and sustainable implementation of NMPC and the presence of significant uncertainty in the available models. The focus here lies on the two last challenges. Dealing with model errors in a systematic but not overly conservative manner also reduces the effort that is needed for the development of high-fidelity models.

Regarding the implementation of NMPC, many software tools have recently been developed in academia such as MUSCOD II (Diehl, Leineweber, & Schäfer, 2001), ACADO (Houska, Ferreau, & Diehl, 2011a), NMPC tools (Amrit & Rawlings, 2008), OptCon (Nagy, 2008) or the MPT Toolbox (Herceg, Kvasnica, Jones, & Morari, 2013) among others. These tools can solve different kinds of problems including NMPC formulations. It is very common however that different applications require different software tools, depending on their complexity. Currently most of these tools require an implementation of a model in a particular syntax and an interface to other necessary components such as a simulator or an observer. If the model is changed, the entire implementation has to be modified. If one wants to test a new tool, most of the code has to be rewritten. This lack of modularity results in complex and non-sustainable implementations, making it also difficult to compare the computational performance and solution qualities of different algorithms or to combine parts of different tools.

This paper extends the results presented in Lucia, Tatulea-Codrean, Schoppmeyer, and Engell (2014) by describing in more detail a new concept for the modularization of a NMPC implementation, dividing it into four main components: model and problem description, optimizer, observer and simulator. Such modular design makes it possible to

* Corresponding author at: Institute for Automation Engineering, Otto-von-Guericke University Magdeburg, Universitätsplatz 2 (Building 07), 39106 Magdeburg, Germany.
E-mail addresses: sergio.lucia@ovgu.de (S. Lucia), alexandru.tatulea-codrean@bci.tu-dortmund.de (A. Tătulea-Codrean), christian.schoppmeyer@bci.tu-dortmund.de (C. Schoppmeyer), s.engell@bci.tu-dortmund.de (S. Engell).

http://dx.doi.org/10.1016/j.conengprac.2016.12.009
Received 25 April 2016; Received in revised form 18 December 2016; Accepted 21 December 2016

compare different solutions (e.g. different discretizations or estimators) just by exchanging the corresponding module, but maintaining the rest of the solution. The effort of going from simulation to an online application is also reduced to the exchange of a simulator module by an application module. The platform supports the multi-stage NMPC (Lucia, Finkler, & Engell, 2013) approach to deal with uncertainties in a systematic way. Both contributions have been combined in the environment do-mpc to provide a framework for the rapid and sustainable development of standard and multi-stage NMPC solutions, which can be easily transferred to online applications, as illustrated by experimental results obtained for a laboratory reactor.

do-mpc provides a unique environment that can be useful to all kinds of users due to the flexibility of the implementation. The use of the CasADi (Andersson, Åkesson, & Diehl, 2012) tool set makes it possible to modify the NMPC formulation provided within do-mpc in a simple manner to include specific elements necessary for alternative formulations of the resulting optimal control problems (e.g., conditions for stability or robustness) rather than providing a black-box NMPC tool. Practitioners can benefit from the existence of templates to develop state-of-the-art implementations of both multi-stage and standard NMPC with low effort. The modularized implementation leads to sustainable NMPC solutions that can be reused when models are updated, or that can be easily transferred to the real system once the desired performance is achieved in simulations, as shown by the experimental results presented in this paper.

The remainder of this paper is structured as follows. Section 2 explains the concepts and the main components of the modular implementation of NMPC, as well as the do-mpc environment. The experimental setup is described in Section 3. The simulation and experimental results, together with a short review of multi-stage NMPC, are presented in Section 4. The paper is concluded in Section 5.

2. do-mpc: an environment for the rapid development of modular and sustainable NMPC solutions

One of the main outcomes of the European research project EMBOCON (Embedded Optimization for Resource Constrained Platforms) was the development of the open source software platform GEMS (Generic EMBOCON Minimal Supervisor) (Schoppmeyer, 2013).

The central idea of GEMS is to offer a set of general and standardized interfaces to simplify the process of developing and deploying a model-based control algorithm for a real system, and to offer a so-called supervisor that manages the flow of information between the different parts of the implementation. The implementation of a model-based control approach in GEMS is divided into four main components: the model, the optimizer, the observer and the simulation or real application (see inner blocks in Fig. 1). The exchange of information between the different modules is managed and logged by a supervisor. Using this conceptual idea, existing or newly developed algorithms for control, for simulation or for state estimation of a system can be implemented based on the GEMS interfaces. The interfaces are structured so as to encapsulate all the information needed by GEMS at run-time and they are implemented in plain C-code, which offers the possibility to integrate other tools and to expand the functionality with basic programming techniques.

This modularization idea has been extended and implemented in the environment do-mpc. do-mpc is a platform that uses the main ideas of GEMS to provide users with an easy, modular and robust way to realize sustainable implementations of NMPC, with a special focus on multi-stage NMPC. Within do-mpc, it is proposed to structure the four different modules in the following manner.

The *model and problem description* module contains the right-hand side of the ordinary differential equations or differential algebraic equations, including all model parameters and the definition of model states and inputs. Here also the control and estimation tasks (initial

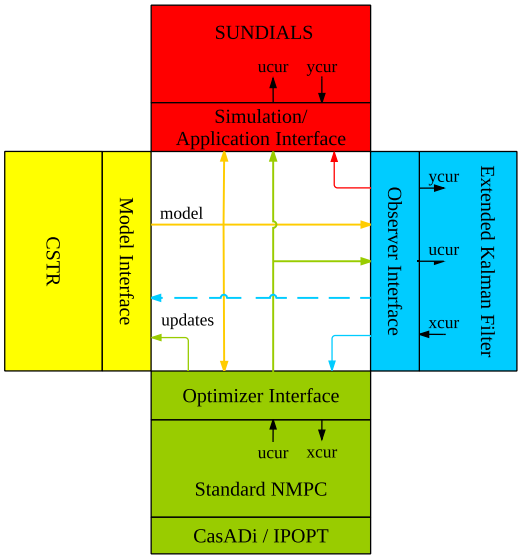

Fig. 1. Modular scheme of an NMPC implementation with four main blocks: Model, optimizer, observer and simulator. The arrows communicating different modules indicate the information exchange and possible updates performed between modules. A combination of the four modules forms a do-mpc configuration.

condition, objective, constraints) should be defined. do-mpc provides templates using the scripting language Python as a user-friendly environment for the representation of the information in the model and problem description module. It is not necessary to use this template, i.e., it is possible to define this information using any existing code or software as long as it passes the necessary information to the model interface so that the other modules can make use of it. It is also possible to define models of different complexity to be used with each one of the other modules. For example, one can use a simple model for the optimization, but a more detailed one for the simulation of the system.

The *optimizer* module contains the implementation of the solution method and of the optimizer for the NMPC problem. For example in the case of a simultaneous NMPC approach it contains the algorithm for the discretization of the dynamics so that a Nonlinear Programming Problem (NLP) is generated (using the information provided in the model and problem description module) and then passed to any available solver (as e.g. IPOPT Wächter & Biegler, 2006) or a self-implemented one. As a major contribution of do-mpc, a Python template that generates the NLP resulting from the robust multi-stage NMPC formulation (Lucia et al., 2013) is provided, or – if chosen by the user – the NLP resulting from standard NMPC formulations. This module also contains the controller parameters (sampling time, prediction horizon, etc.).

The *observer* module contains an algorithm in which, given the measurements of the plant or simulation, the states of the system are calculated to initialize the optimizer. Standard observers such as the Extended Kalman Filter or the Moving Horizon Estimator can be implemented in a generalized manner so that the information of the other modules is used and there is no need to re-implement the observers for different models or optimizers.

The *simulation* module contains an integrator which can be self-coded or interfaced with existing software such as the SUNDIALS (Hindmarsh et al., 2005) toolbox. For a real test case, this module contains an interface to the Input–Output device that is used for the communication with the real plant. The implementation of the module is application specific and is able to send the calculated control inputs and receive the measurements from the real system.

do-mpc is built upon CasADi (Andersson et al., 2012) as a building block for the necessary modules and interfaces. CasADi is a tool for

automatic differentiation and dynamic optimization which manages the model and derivative information in an easy and efficient way. It is also possible to automatically generate C-code versions of the modules that have been defined within the Python templates. The auto-generated C-code is compiled to shared libraries, which are loaded at run-time by do-mpc. From the practical perspective this means that the exchange of the different modules (optimizers, estimators, etc.) can be done by just loading a different shared library at run-time. Note that the implementation of the modules using CasADi and the provided templates is only an option to minimize the coding effort, but any existing code can be used as long as it is interfaced with the standard interfaces.

An example of the do-mpc configuration with the four modules using an EKF as observer and the SUNDIALS integrator as simulator can be seen in Fig. 1.

The separate implementation of each module can result in a slightly larger implementation effort compared to a tailored NMPC implementation for a given application because each module should be independent of each other, except for the exchanged information. In addition, the modular implementation makes it difficult to re-use computations that are performed inside the modules, as it could be potentially done in a highly tailored implementation. However, the benefits of a modular implementation outweigh, in our opinion, the slightly larger effort needed. The main advantage of our solution is that running a new NMPC simulation reduces to setting up a do-mpc configuration (a module for each part), according to the given requirements. For example, if a new model implementation needs to be evaluated, only the model and problem description module has to be exchanged. It is thus possible to have libraries of models, optimizers, observers and simulators, with the help of which just by exchanging the desired module, a new NMPC implementation can be generated. This facilitates the comparison between different approaches and the sustainability of the implementations. In addition, the optimizer, observer, or simulator modules that have already been implemented can be directly reused when a new model and problem description module is defined, since their implementation is independent of the given model, but instead exchanges information following a set of standardized interfaces which are equal for all possible models. Another major advantage of the proposed modularization is the simplification of the deployment of the controller for the real system once the desired result is achieved in simulations. The deployment is reduced to the exchange of the simulation module with the module that manages the input–output communications with the plant. This transparent deployment operation simplifies the time consuming error detection and error tracking tasks that often occur when deploying a new NMPC solution.

The modular implementation makes it possible to use different configurations of do-mpc in parallel. As it is shown in Fig. 2, it is possible to run in parallel Configuration 1 and Configuration 2, which use the same model and application modules, but different optimizers and observer modules. A supervisor which has access to both configurations can decide which controller to actually use to control the real physical system. For example, one can use monitoring techniques based on multivariate statistics (AlGhazzawi & Lennox, 2009) or other monitoring approaches (Patwardhan, Shah, & Qi, 2002; Zagrobelny, Ji, & Rawlings, 2013) to decide if a controller is performing correctly. If it is not the case, a different configuration can be chosen. This capability of do-mpc can also be used to implement a recovery strategy that is triggered whenever one of the modules fails at the run-time. Additionally, a degree of redundancy can be introduced to cope with possible failures of one of the algorithms. do-mpc also features a basic data visualization tool which shows the time plots of any variable previously defined along with the predictions that the NMPC controller is computing. The visualization enhances the understanding of the controller performance and accelerates the design process of satisfactory NMPC solutions.

The structure of the tool and a graphical user interface have been developed using the Qt framework for C++ (Qt5.1, 2014) and Python. The platform was implemented based on CasADi. Fig. 3 shows the main components and the structure of the platform. do-mpc takes the information provided by the user in the templates (left part of the figure) and, after running it through its pre-implemented algorithms which rely on the CasADi API (top part of the figure), it automatically generates the four modules of the configuration architecture presented in Fig. 1.

Thus, completing the templates on the left part of Fig. 3 is the only necessary step to achieve an efficient robust NMPC implementation. While do-mpc offers this highly automated implementation procedure, it also supports the use of customized extensions. This means that any user can modify the predefined algorithms (top part) or can implement new ones using the CasADi API, which offers a convenient way to treat derivative information and includes interfaces to many existing software. Alternatively, users can implement the building blocks of the do-mpc configuration using external software (right part of the figure), as long as they are interfaced with the existing do-mpc structure. do-mpc also uses CasADi to automatically generate code that contains the different blocks of the do-mpc configuration which can be stored in a library of modules, or used for external applications.

In particular, do-mpc uses four main classes to define each one of the modules. Each class has different attributes that define all the necessary information that each type of module should have. For example, a model and problem description module should include differential and algebraic states, control inputs, differential and algebraic equations, uncertain parameters, and the description of the optimal control problem, which includes initial conditions, constraints and cost function. This process is automatically done (see the top part of Fig. 3) if the provided templates are used. Alternatively, other modules that are developed externally should match these attributes with an interface, as depicted on the right part of Fig. 3. After the instantiation of each one of the four modules, a configuration class is created using the four modules as inputs, forming the main object of the do-mpc implementation. The configuration object has three main methods to perform the simulation, estimation and optimization steps. In addition, it manages the data associated to each configuration, so that different configuration objects can be easily coupled with a supervisory and a data visualization layer as described in Fig. 2. The configuration class also contains methods for data management, data plotting, as well as a method to optionally provide reasonable initial guesses for the optimizer, which are obtained by simulating the model with a constant input and nominal values of the parameters.

More details about the implementation can be seen in the freely available code (do mpc, 2015).

do-mpc has been developed specifically for receding horizon implementations and it is therefore not suitable for the development of other optimal control strategies in which the information flow between the different modules (optimizer, simulator, observer) differs significantly from the flow of an MPC implementation. do-mpc requires that the user installs the software that he or she wants to use (CasADi, IPOPT, SUNDIALS, etc.). Within the freely available version of do-mpc, several model and problem description modules are provided, corresponding to typical examples of the chemical engineering field as well as different optimizer modules that include orthogonal collocation, single shooting and multiple shooting discretizations together with multi-stage or standard NMPC implementations. Any solver (such as IPOPT) that can be called via CasADi can be directly used also within do-mpc without any additional work. This includes, e.g., KNITRO, SNOPT, and WORHP. Other solvers can be coupled writing an interface. The optimizer modules are completely independent from the model and therefore they do not have to be edited when used for a new example. A simulator module using the integrators of the SUNDIALS toolbox is also provided. Users are invited to develop the additional modules e.g., implementing a generalized Moving Horizon Estimator module.

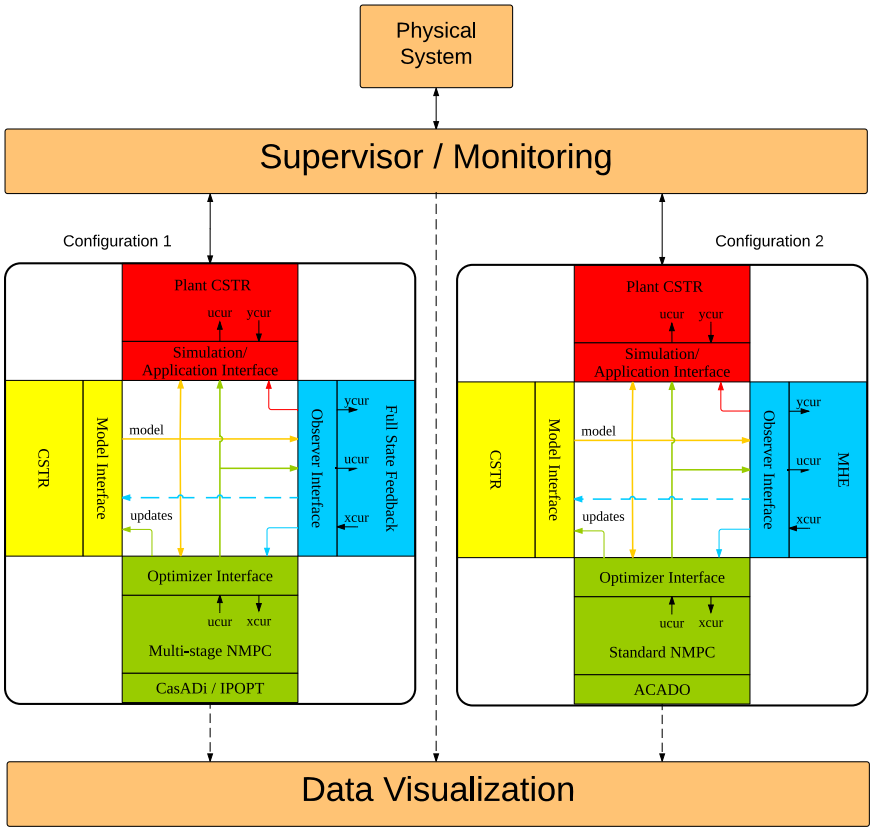

Fig. 2. Example of an NMPC implementation using DO-MPC with two configurations in parallel. A supervisory or monitoring layer can decide which of the configurations is actually used to control the real physical system based on different criteria.

3. Continuous stirred tank reactor example

The modular implementation described in the previous section is demonstrated in the rest of this paper for a real lab-scale setup. The plant under consideration (see Fig. 4) is located at the Group of Process Dynamics and Operations at TU Dortmund. It consists of a continuous stirred tank reactor (CSTR) which is equipped with a jacket. The thermal behavior of the CSTR is realized in hardware while the kinetics of the chemical reaction are realized in software, providing flexibility to simulate all kinds of reactions without having to care about plant safety too much. A schematic representation of the plant is shown in Fig. 5.

The jacket is fed with water, which is supplied from a thermostat,

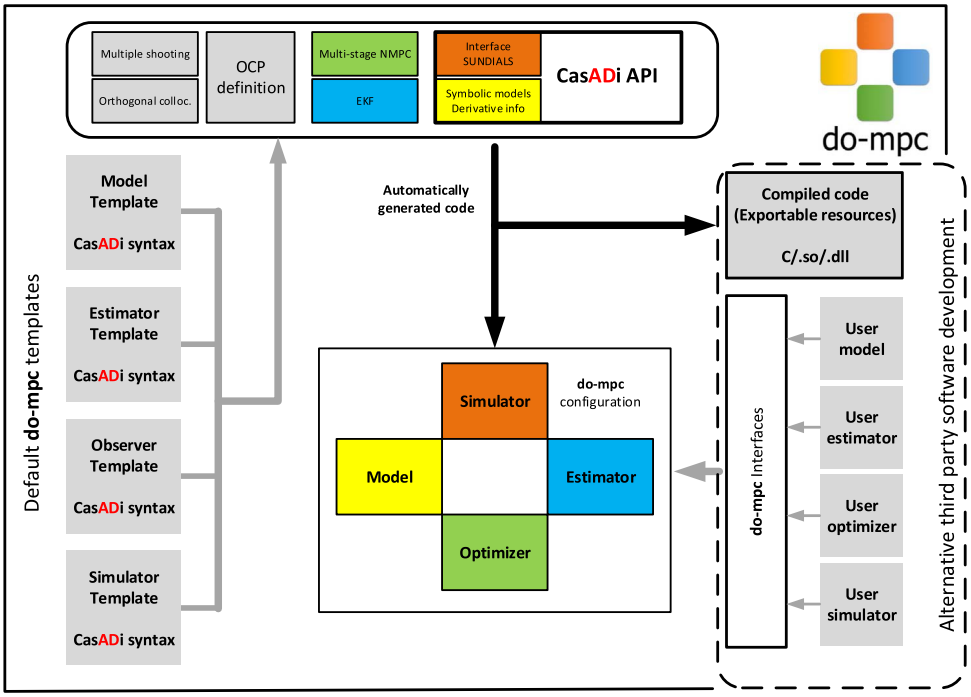

Fig. 3. An overview of the do-mpc platform. Templates, main building blocks and information exchange are presented.

Fig. 4. Experimental setup of the thermal part of the CSTR located at TU Dortmund.

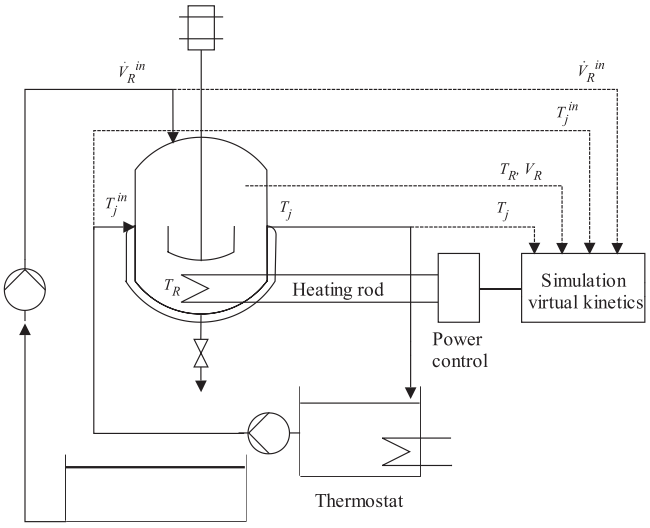

Fig. 5. Schematic representation of the CSTR under consideration with virtual reaction kinetics.

which can heat or cool the water if necessary, modifying the temperature at the inlet of the jacket T_j^{in}. The reactor is operated in a semi-batch mode, the input flow ($\dot{V}_{\text{R}}^{\text{in}}$) can be adjusted using a pump, and there is no outflow. The heat generated by the simulated chemical reaction is introduced using a heating rod which produces the heat that an exothermic reaction taking place inside the reactor would produce.

In the chemical model that is considered here, it is assumed that two reactants A and B react to form product C according to the following reaction scheme:

$$\text{A} + \text{B} \rightarrow \text{C}. \tag{1}$$

From energy, mass and component balances, the differential equations that describe the dynamics of the system were derived:

$$\dot{V}_{\text{R}} = \dot{V}_{\text{R}}^{\text{in}}, \tag{2a}$$

$$\dot{T}_{\text{R}} = \frac{\dot{V}_{\text{R}}^{\text{in}}}{V_{\text{R}}}(T_{\text{R}}^{\text{in}} - T_{\text{R}}) - \frac{\alpha A (T_{\text{R}} - T_{\text{J}})}{\rho V_{\text{R}} c_{\text{p}}} - \frac{k c_{\text{A}} c_{\text{B}} \Delta H_{\text{R}}}{\rho c_{\text{p}}}, \tag{2b}$$

$$\dot{T}_{\text{J}} = \frac{\dot{V}_{\text{J}}^{\text{in}}}{V_{\text{J}}}(T_{\text{J}}^{\text{in}} - T_{\text{J}}) + \frac{\alpha A (T_{\text{R}} - T_{\text{J}})}{\rho V_{\text{J}} c_{\text{p}}}, \tag{2c}$$

$$\dot{c}_{\text{A}} = -\frac{\dot{V}_{\text{R}}^{\text{in}}}{V_{\text{R}}} c_{\text{A}} - k c_{\text{A}} c_{\text{B}}, \tag{2d}$$

$$\dot{c}_{\text{B}} = -\frac{\dot{V}_{\text{R}}^{\text{in}}}{V_{\text{R}}}(c_{\text{b}}^{\text{in}} - c_{\text{b}}) - k c_{\text{A}} c_{\text{B}}, \tag{2e}$$

$$\dot{c}_{\text{C}} = -\frac{\dot{V}_{\text{R}}^{\text{in}}}{V_{\text{R}}} c_{\text{C}} + k c_{\text{A}} c_{\text{B}}, \tag{2f}$$

$$\dot{T}_{\text{J}}^{\text{in}} = \frac{\overline{T}_{\text{J}}^{\text{in}} - T_{\text{J}}^{\text{in}}}{\tau_t}. \tag{2g}$$

The model includes a mass balance for the volume of the reactor V_{R} where $\dot{V}_{\text{R}}^{\text{in}}$ is the input flow, which is a control input of the system. The energy balances in (2b) and in (2c) describe the dynamics of the temperature of the reactor T_{R} and of the temperature of the jacket T_{J}, which is assumed to be perfectly mixed. The parameter α is the heat transfer coefficient and A is the heat transfer area, which can be calculated as a function of the radius of the reactor r and of the volume of water in the reactor V_{R} as $A = \pi r^2 + 2\frac{V_{\text{R}}}{r}$. The specific heat and the density of the water used in the experiment are denoted by c_{p} and ρ. $\dot{V}_{\text{J}}^{\text{in}}$ is the flow rate into the jacket, which is considered constant, and T_{R}^{in} denotes its temperature. V_{J} is the volume of the cooling fluid inside the jacket. The component balances for the substances A, B, C are described in (2d)–(2f), where c_i denotes the concentration of component i. The reaction constant is denoted by k and c_{b}^{in} is the concentration of component B in the inflow $\dot{V}_{\text{R}}^{\text{in}}$. In the considered setup the inflow only contains the substance B. The dynamics of the thermostat are approximated by a first order system with a time constant τ_t (see Eq. (2g)). The thermostat has different dynamics depending on whether it is cooling or heating the water. For this reason the time constant τ_t varies depending on the temperature of the water at the inlet of the jacket:

$$\tau_t = \begin{cases} \tau_t^{\text{cool}}, & T_{\text{J}}^{\text{in}} \geq \overline{T}_{\text{J}}^{\text{in}}, \\ \tau_t^{\text{heat}}, & T_{\text{J}}^{\text{in}} < \overline{T}_{\text{J}}^{\text{in}}. \end{cases} \tag{3}$$

The set-point of the thermostat ($\overline{T}_{\text{J}}^{\text{in}}$) is the second control input of the system. The values of all the parameters of the model can be found in Table 1.

The initial conditions of the states together with the constraints on the states are stated in Table 2. The constraints for the control inputs are shown in Table 3.

All the parameters have been obtained from experiments. The

Table 1
Parameter values of the CSTR.

Parameter	Value	Unit
$\dot{V}_{\text{J}}^{\text{in}}$	$9{\cdot}10^{-6}$	$\text{m}^3\,\text{s}^{-1}$
T_{R}^{in}	299.15	K
α	0.1484	$\text{kW}\,\text{m}^{-2}$
r	0.092	m
ρ	1000	$\text{kg}\,\text{m}^{-3}$
c_p	4.2	$\text{kJ}\,\text{kg}^{-1}\,\text{K}^{-1}$
V_{J}	0.00222	m^3
k	$1.339{\cdot}10^{-6}$	$\text{m}^3\,\text{mol}^{-1}\,\text{s}^{-1}$
ΔH	−50	$\text{kJ}\,\text{mol}^{-1}$
c_{b}^{in}	$4{\cdot}10^{6}$	$\text{mol}\,\text{m}^{-3}$
τ_t^{heat}	200	s^{-1}
τ_t^{cool}	950	s^{-1}

Table 2
Initial conditions and state constraints.

State	Init. cond.	Min.	Max.	Unit
V_R	0.0035	0	0.01	m^3
T_R	50.0	48.0	52.0	°C
T_J	50.0	0	100.0	°C
c_A	2000.0	0	Inf	mol m^{-3}
c_B	0	0	Inf	mol m^{-3}
c_C	0	0	Inf	mol m^{-3}
T_J^{in}	50.0	0	100	°C

Table 3
Bounds on the manipulated variables.

Control	Min.	Max.	Unit
$\dot{V}_R^{in}$	0	$9{\cdot}10^{-6}$	$m^3\ s^{-1}$
$\bar{T}_J^{in}$	30	80	°C

parameters that determine the virtual reaction are the reaction constant k and the reaction enthalpy ΔH. These parameters are chosen so that it is possible to emulate the virtual reaction with the equipment available. The heat that would be generated by the reaction $Q_{rea} = \Delta H\ k\ c_A c_B V_R$ is generated by the heating rod during the experiment.

The control task consists in maximizing the amount of product C obtained ($n_C = V_R c_C$), while satisfying the state and input constraints defined in Tables 2 and 3.

4. Design of NMPC controllers and experimental results

This section presents the design steps and experimental results obtained with standard and multi-stage NMPC for the CSTR presented above.

4.1. Model validation

The first step when applying a model-based technique to control a system is to check the quality of the available model and its prediction capabilities. With the help of this information, it has to be decided whether the available model is predicting the behavior of the system well enough or whether some improvements are needed. For this purpose, an experiment was performed in which a trajectory for the control inputs was given and the temperature measurements of the experimental setup are compared with the ones obtained by simulating the model (2). From the given initial condition the model is simulated in parallel and the values of the concentrations are used to calculate the heat of reaction Q_{rea} which is generated using the heating rod that emulates the chemical reaction.

The results obtained are presented in Fig. 6. The simulation reproduces the results obtained experimentally well and it was decided that the model is accurate enough to be used for nonlinear model predictive control. Small deviations are expected due to the assumptions made (perfect mixing in the jacket), the approximation of the thermostat dynamics by a first order system, not modeled effects (heat transfer to the environment), small inaccuracies in some of the reactor parameters and sensor noise.

4.2. Handling uncertainty: multi-stage NMPC

It is considered that the reaction parameters k and ΔH are uncertain and can vary ±25% with respect to the nominal values reported in Table 1. These uncertainties strongly affect the dynamics of the system and it is therefore necessary to employ a robust NMPC approach to achieve a satisfactory performance, including robust

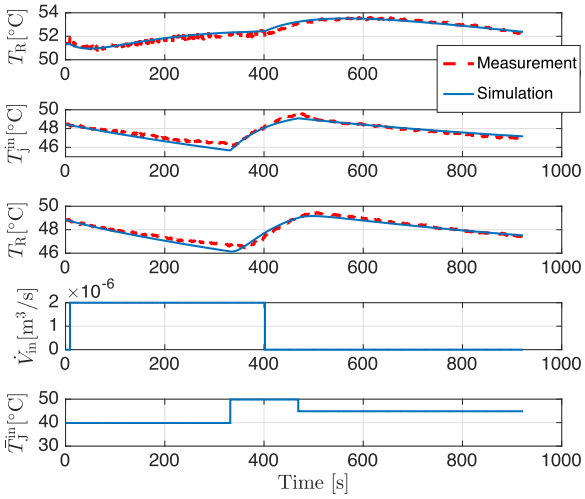

Fig. 6. Temperature of the reactor, temperature at the inlet of the jacket and temperature of the jacket for a given trajectory of control inputs. The solid lines represent simulation results and the dashed lines experimental results.

constraint satisfaction. Here multi-stage NMPC (Lucia et al., 2013; Lucia, Andersson, Brandt, Diehl, & Engell, 2014) is used, which can be efficiently implemented using the do-mpc framework. Multi-stage NMPC is a robust NMPC approach that is based on describing the evolution of the system under uncertainty by a scenario tree (see Fig. 7). Each branching at a node represents the effect of an unknown uncertain influence (disturbance or model error) together with the chosen control input. Due to the presence of feedback, future control inputs can depend on the previous values of the uncertainty. In a multi-stage formulation, the optimal values of the first stage variables (here and now decisions, those that are actually implemented) are computed while taking into account the possible adaptation of the future inputs which can be different along the branches of the tree (recourse actions). Because it takes the presence of feedback into account, multi-stage NMPC exhibits a lower degree of conservativeness compared to other approaches, such as open-loop min–max NMPC (Campo & Morari, 1987; Lee & Yu, 1997). The use of a scenario tree for MPC was suggested already in Scokaert and Mayne (1998) and some results for multi-stage linear MPC have been reported in Muñoz de la Peña, Bemporad, and Alamo (2005) and in Bernardini and Bemporad (2009). In order to represent the real-time decision problem correctly, the

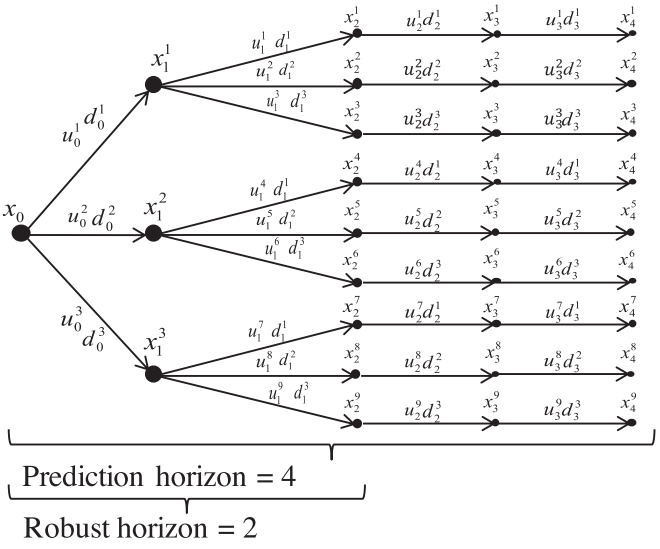

Fig. 7. Scenario tree representation of the evolution of the process for multi-stage NMPC with robust horizon.

control inputs cannot anticipate the values of the uncertainty that are realized after the corresponding decision point. This is enforced by the so-called non-anticipativity (or causality) constraints that require all the control inputs that branch at the same node to be equal.

A simple strategy to deal with the exponential growth of the tree with the prediction horizon is to assume that the uncertainty remains constant after a certain point in time (called the robust horizon), as illustrated in Fig. 7. This simplification has been shown to provide very good results in practice as can be seen, e.g. in Lucia et al. (2013) and Lucia, Andersson, et al. (2014). The scenario tree setting assumes a discrete-time formulation of an uncertain nonlinear dynamic system described by $\boldsymbol{f}$: $\mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$ that can be written as:

$$\boldsymbol{x}_{k+1}^{j} = \boldsymbol{f}(\boldsymbol{x}_{k}^{p(j)}, \boldsymbol{u}_{k}^{j}, \boldsymbol{d}_{k}^{r(j)}), \tag{4}$$

where $\boldsymbol{x}_{k+1}^{j} \in \mathbb{R}^{n_x}$ denotes the state at stage k+1 and $\boldsymbol{x}_{k}^{p(j)}$ denotes its parent state, $n_{\boldsymbol{x}}$ being the number of states. The control input vector is denoted as $\boldsymbol{u}_{k}^{j} \in \mathbb{R}^{n_u}$, $n_{\boldsymbol{u}}$ is the number of control inputs. The realization r of the uncertainty at stage k, which is a function of the position j in the scenario tree at each stage, is denoted as $\boldsymbol{d}_{k}^{r(j)} \in \mathbb{R}^{n_d}$, where $n_{\boldsymbol{d}}$ is the dimension of the uncertainty vector (for example in Fig. 7, $\boldsymbol{x}_{2}^{7} = \boldsymbol{f}(\boldsymbol{x}_{1}^{3}, \boldsymbol{u}_{1}^{7}, \boldsymbol{d}_{1}^{1})$). For simplicity of the presentation, it is considered that the tree has the same number of branches at all nodes, given by $\boldsymbol{d}_{k}^{r(j)} \in \{\boldsymbol{d}_{k}^{1}, \boldsymbol{d}_{k}^{2}, \ldots, \boldsymbol{d}_{k}^{s}\}$ at stage k for s different possible values of the uncertainty. In order to make the notation clear, the index set of all occurring indices (j, k) is denoted by I. Each path from the root node $\boldsymbol{x}_0$ to a leaf node $\boldsymbol{x}_{N_P}^{i}$ is called a scenario and is denoted by S_i. The number of scenarios (or leaf nodes) is denoted by N and N_P is the prediction horizon. The set of states that belong to scenario i is defined as X_i and the set of control inputs that belong to scenario i is defined as U_i.

The optimization problem that results from the multi-stage formulation can be written as:

$$\min_{x_k^j, \boldsymbol{u}_k^j \forall (j,k) \in I} \quad \sum_{i=1}^{N} \omega_i J_i(X_i, U_i), \tag{5a}$$

$$\text{subject to:} \quad \boldsymbol{x}_{k+1}^{j} = \boldsymbol{f}(\boldsymbol{x}_{k}^{p(j)}, \boldsymbol{u}_{k}^{j}, \boldsymbol{d}_{k}^{r(j)}), \quad \forall\ (j, k+1) \in I, \tag{5b}$$

$$\boldsymbol{g}(\boldsymbol{x}_{k+1}^{j}, \boldsymbol{u}_{k}^{j}) \le 0, \quad \forall\ (j, k+1) \in I, \tag{5c}$$

$$\boldsymbol{u}_{k}^{j} = \boldsymbol{u}_{k}^{l} \text{ if } \boldsymbol{x}_{k}^{p(j)} = \boldsymbol{x}_{k}^{p(l)} \quad \forall\ (j, k), (l, k) \in I, \tag{5d}$$

where $\boldsymbol{g}$: $\mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$ represents general and possibly nonlinear constraints on the states and the inputs of the control problem evaluated at each node of the tree. The number of constraints is determined by $n_{\boldsymbol{g}}$. The cost of each scenario S_i is denoted by J_i: $\mathbb{R}^{n_x \times N_P+1} \times \mathbb{R}^{n_u \times N_P} \to \mathbb{R}$ defined as:

$$J_i(X_i, U_i) = \sum_{k=0}^{N_P-1} L(\boldsymbol{x}_{k+1}^{j}, \boldsymbol{u}_{k}^{j}), \quad \forall\ \boldsymbol{x}_{k+1}^{j} \in X_i, \quad \boldsymbol{u}_{k}^{j} \in U_i, \tag{6}$$

where L: $\mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is the stage cost, which represents a general cost function. The cost of each scenario S_i is weighted by ω_i, which can represent the degree of likelihood or the importance of each scenario. The non-anticipativity constraints in (5d) enforce that the decisions $\boldsymbol{u}_{k}^{j}$ with the same parent node $\boldsymbol{x}_{k}^{p(j)}$ are the same.

Following the implementation guidelines based on the templates of do-mpc (do mpc, 2015), the only added complexity necessary to formulate a multi-stage NMPC approach is to define the bounds of the uncertainties. A scenario tree is then automatically created considering the combinations of the extreme values and the nominal values of the uncertainties. Of course, any other scenario tree can be alternatively used. This leads to a simple and efficient robust NMPC implementation that is not currently available in any other tool.

4.3. Controller formulation and implementation details

In the resulting implementation of the NMPC controller, the following optimization problem is solved at each sampling time:

$$\min_{x_k^j, \boldsymbol{u}_k^j \forall (j,k) \in I} \quad -\sum_{i=1}^{N} \omega_i \sum_{k=0}^{N_P-1} V_{\mathrm{R},k}^{j} c_{\mathrm{C},k}^{j} + r_1 \Delta \dot{V}_{\mathrm{R}}^{\mathrm{in}2} + r_2 \Delta \overline{T}_{\mathrm{J}}^{\mathrm{in}2} + \mu \epsilon_k^{j2},$$

$$\forall\ \boldsymbol{x}_{k}^{j} \in X_i, \boldsymbol{u}_{k}^{j} \in U_i \tag{7a}$$

$$\text{subject to:} \quad \boldsymbol{x}_{k+1}^{j} = \boldsymbol{f}(\boldsymbol{x}_{k}^{p(j)}, \boldsymbol{u}_{k}^{j}, \boldsymbol{d}_{k}^{r(j)}), \quad \forall\ (j, k+1) \in I, \tag{7b}$$

$$0 \le V_{\mathrm{R},k}^{j} \le 0.01, \quad \forall\ (j, k) \in I, \tag{7c}$$

$$48 \le T_{\mathrm{R},k}^{j} + \epsilon_k^{j} \le 52, \quad \forall\ (j, k) \in I, \tag{7d}$$

$$0 \le T_{\mathrm{R},k}^{j} \le 100, \quad \forall\ (j, k) \in I, \tag{7e}$$

$$0 \le T_{\mathrm{J},k}^{\mathrm{in},j} \le 100, \quad \forall\ (j, k) \in I, \tag{7f}$$

$$0 \le c_{\mathrm{A},k}^{j}, \quad \forall\ (j, k) \in I, \tag{7g}$$

$$0 \le c_{\mathrm{B},k}^{j}, \quad \forall\ (j, k) \in I, \tag{7h}$$

$$0 \le c_{\mathrm{C},k}^{j}, \quad \forall\ (j, k) \in I, \tag{7i}$$

$$0 \le c_{\mathrm{B},k}^{\mathrm{acc},j} \le 2000, \quad \forall\ (j, k) \in I, \tag{7j}$$

$$0 \le \dot{V}_{\mathrm{R},k}^{\mathrm{in},j} \le 9 \cdot 10^{-6}, \quad \forall\ (j, k) \in I, \tag{7k}$$

$$30 \le \overline{T}_{\mathrm{J},k}^{\mathrm{in},j} \le 80, \quad \forall\ (j, k) \in I, \tag{7l}$$

$$\boldsymbol{u}_{k}^{j} = \boldsymbol{u}_{k}^{l} \text{ if } \boldsymbol{x}_{k}^{p(j)} = \boldsymbol{x}_{k}^{p(l)} \quad \forall\ (j, k), (l, k) \in I, \tag{7m}$$

where $\boldsymbol{x}_{k}^{j}$ contains all the states at each node of the scenario tree

$$\boldsymbol{x}_{k}^{j} = [V_{\mathrm{R},k}^{j}, T_{\mathrm{R},k}^{j}, c_{\mathrm{A},k}^{j}, c_{\mathrm{B},k}^{j}, c_{\mathrm{C},k}^{j}, T_{\mathrm{J},k}^{\mathrm{in},j}]^T$$

and $\boldsymbol{u}_{k}^{j}$ contains all the control inputs at each node

$$\boldsymbol{u}_{k}^{j} = [\dot{V}_{\mathrm{R},k}^{\mathrm{in},j}, \overline{T}_{\mathrm{J},k}^{\mathrm{in},j}]^T.$$

Since the optimization problem is solved using real measurements, the hard constraint on the temperature of the reactor is relaxed and implemented as a soft constraint by adding the parameter ϵ_k^j at each point in the prediction horizon as shown in (7d) and then penalized in the cost function (7a) using a large penalty weight $\mu = 10^6$. This is done to avoid infeasible optimization problems. An ℓ_1 penalty term could be used instead of the ℓ_2 term chosen here. Since the optimal solution will operate at the constraint, it is likely that due to measurement errors or plant model mismatches, at some point in time the measurement lies outside of the constraint and therefore the resulting optimization problem becomes infeasible if no additional measures are taken. This can jeopardize the performance of the control scheme by giving unsuitable commands to the real plant if no backup strategy is available when an infeasible optimization problem is encountered. Furthermore, an additional constraint (7j) is added to limit the amount of component B that can be fed into the reactor $c_{\mathrm{B}}^{\mathrm{acc}} = \int_0^{\infty} c_{\mathrm{b}}^{\mathrm{in}} \dot{V}_{\mathrm{R}}^{\mathrm{in}}\, \mathrm{d}t$. If this constraint is not added the batch would end with a large amount of component B in the reactor and this is not desired. To avoid oscillatory behavior of the control inputs, penalty terms on the control moves are introduced as indicated in (7a) using the tuning parameters $r_1 = 5 \cdot 10^7$ and $r_2 = 0.001$.

The implementation of standard and multi-stage NMPC was realized using do-mpc, both for the simulation and for the experimental results. The communication between the plant and the computer is done via USB using the data acquisition system Labjack U12 except for the thermostat, which uses serial port (RS-232) communication. To perform the experiments, the system is first driven to the initial condition described in Table 2. Then the NMPC controller is started and at the same time a simulation of the system is run in parallel using as control inputs those values computed by the NMPC. Every second, the values of the concentrations obtained by the parallel simulation are used to calculate the heat of reaction Q_{rea} and the corresponding command is given to the heating rod. This emulates the

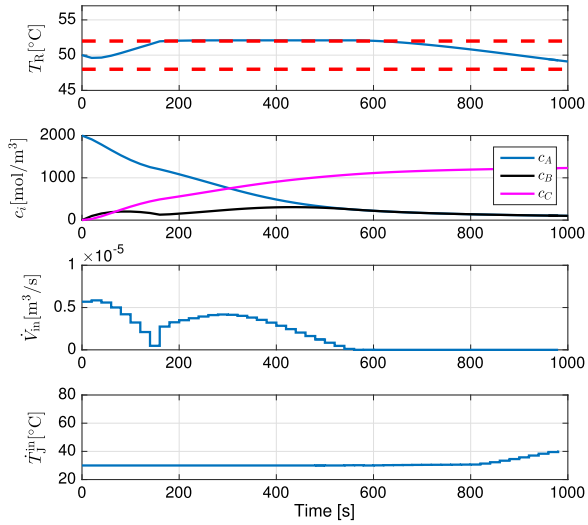

Fig. 8. Reactor temperature, concentrations, input flow and setpoint of the thermostat for simulation of standard NMPC with no uncertainty on the reaction parameters.

chemical reaction that is assumed to take place in the reactor. The values of the concentrations obtained by the simulation are corrupted by white noise ($\sigma = 0.1$ mol m^{-3}) and used as measurements for the NMPC controller. It is assumed that all states can be measured: the temperatures and the volume of the reactor are directly measured using sensors and the concentrations are obtained from the parallel simulation of the reaction after the addition of measurement noise. The same probabilities ω_i are chosen for all the scenarios. The sampling time of the controller is $t_{step} = 20$ s and the prediction horizon is N_P=15 steps. For the multi-stage results, a scenario tree is generated considering as scenarios the combinations of the maximum, minimum and nominal values of the uncertain reaction parameters k and ΔH. It is considered that the parameters can vary by $\pm 25\%$ with respect to the nominal values reported in Table 1. A robust horizon of N_R=1 is chosen, which results in a total of N=9 scenarios. For all the simulation and experimental results, orthogonal collocation on finite elements is used for the discretization of the nonlinear ODEs.

4.4. Results

Fig. 8 shows the simulation results of standard NMPC applied to the CSTR without uncertainties, i.e. the values of the reaction parameters k and ΔH are the same in the model and in the emulation of the reaction heat. The feed ($\dot{V}_R^{in}$) is adjusted such that the constraints on the temperature of the reactor are satisfied. The NMPC keeps the setpoint of the thermostat ($\overline{T}_J^{in}$) to the minimum to be able to feed more and thus to maximize the amount of component C produced.

One of the main advantages of the NMPC implementation realized within do-mpc is the ease with which one can switch between different modules to compare different implementations. For example, if the user wants to compare the solution shown in Fig. 8 with the solution obtained with a different cost function, the only necessary step is to exchange the model and problem description module by another one with the same model but a different cost function as illustrated in Fig. 9. The result of such an exchange can be seen in Fig. 10, where a quadratic tracking term is added to the cost function defined in (7a) so that the temperature of the reactor T_R tracks the temperature $T_R^{track} = 50$ °C.

The various model and optimization parameters were first tuned until a satisfactory simulation result is reached, as shown in Fig. 8. After this design stage, and because of the modular implementation of do-mpc, the only necessary step to obtain experimental results is to exchange the simulator module (that uses an integrator – in this case SUNDIALS) with the module that contains the interface with the real plant. This is illustrated in Fig. 11, which shows the do-mpc configuration used to obtain the simulation results reported in Fig. 8 and the experimental results that are shown in Fig. 12. This approach allows the user to perform quick *software in the loop* tests with do-mpc. The process of transferring the simulation results to reality becomes very simple and transparent, and possible implementation errors can be tracked easily. By comparing the simulation results in Fig. 8 and the experimental results in Fig. 12, it can be concluded that the experimental standard NMPC of the CSTR has an excellent performance and reproduces the simulation results accurately.

If standard NMPC is applied when the plant parameters differ from the nominal values considered in the model, violations of the constraint on the reactor temperature T_R occur as it can be seen in the simulation results shown in Fig. 13 (left) and in the experimental results shown in Fig. 13 (right). In this case, the kinetic parameters k and ΔH, which are used to calculate the heat of reaction Q_{rea}, have values that are 25% higher than their nominal values. The only necessary step to obtain the experimental results is to exchange the simulation module by the module that contains the interface with the physical system.

The violations of the constraints can be avoided if multi-stage NMPC is used. This can be observed in Fig. 14, where the results for multi-stage NMPC of the CSTR are shown. The left plot shows the simulation results and the right plot shows the results obtained experimentally. Again, the simulation results match the experimental results very well. Less product C is produced when compared to the use of standard NMPC shown in Fig. 13 because the feed is reduced to avoid the violations of the constraints.

Further simulation (left) and experimental (right) results are shown in Fig. 15 for the case when uncertain parameters have their nominal value and in Fig. 16, which shows the results obtained when the uncertain parameters are chosen to be 25% smaller than their nominal value. It can be seen that multi-stage NMPC satisfies the constraints for all the scenarios also in the experiments.

The use of multi-stage NMPC with a simple scenario tree that contains the extreme values of the uncertainty leads to a robust control of the plant, also for those values of the uncertainty that are not explicitly included in the scenario tree. It is clear that this is not guaranteed for general nonlinear systems, but in practice constraints are satisfied very often even for values that are not included in the scenario tree (Lucia, Andersson, et al. (2014); Lucia et al., 2013; Lucia & Paulen, 2014). This can be seen in Fig. 17, where simulation results for different values of the parameters are plotted. The reactor temperature (left) and the inflow to the reactor are plotted for combinations of the parameters that vary simultaneously within the range of ± 25%, so that a three-dimensional plot is obtained. Other combinations (e.g. increasing k and decreasing ΔH) lead to the same qualitative results. It can be seen that the constraints on the temperature of the reactor are satisfied for all possible values of the uncertainty when using multi-stage NMPC with a robust horizon of 1.

The average computation time needed for the solution of each optimization problem was 0.073 s for standard NMPC and 0.65 s for multi-stage NMPC. Very similar computation times were observed between the simulation and the experimental results. Since this time is small in comparison to the sampling time ($t_{step} = 20$ s), no additional measures are taken to counteract the effect of the computation delay. For other cases where the delay is significant, it can be taken into account by simulating the system for the expected computation time and then by using this state as initial condition of the NMPC controller. More advanced techniques to cope with this problem include the use of the real-time iteration algorithm described in Diehl, Bock, and Schlöder (2005).

Multi-stage NMPC provides a very flexible robust NMPC method, which can be combined with other approaches to enhance its capabilities and that can also be integrated easily within do-mpc. For example, if a rigorous guarantee for robust constraint satisfaction is needed for all the cases of the uncertainty (also for those values that are not

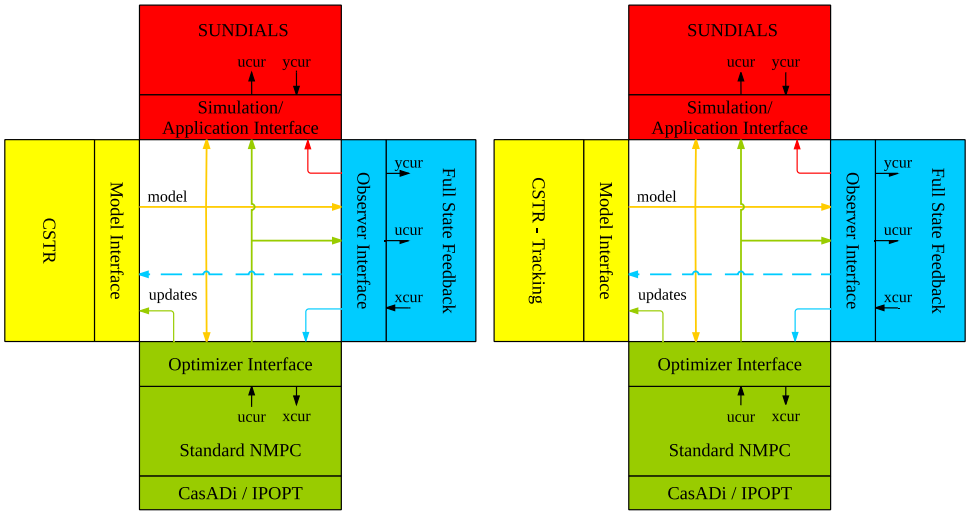

Fig. 9. do-mpc configurations used to obtain the simulation results with the economic cost defined in (7a) and the simulation results with an additional tracking term of standard NMPC applied to the CSTR under consideration.

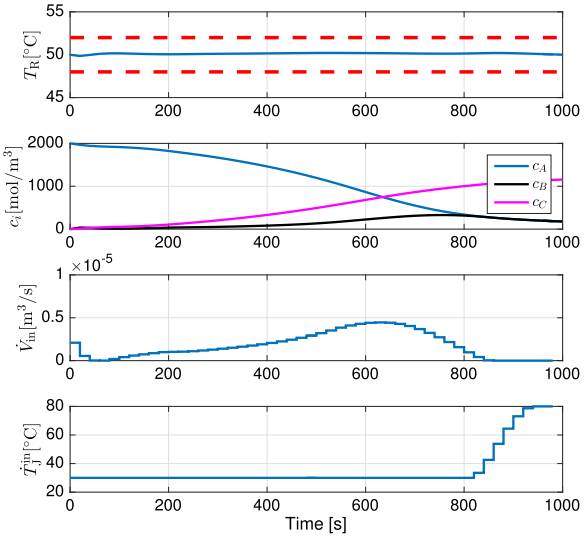

Fig. 10. Reactor temperature, concentrations, input flow and setpoint of the thermostat for simulation of standard NMPC with an additional tracking term and no uncertainty on the reaction parameters.

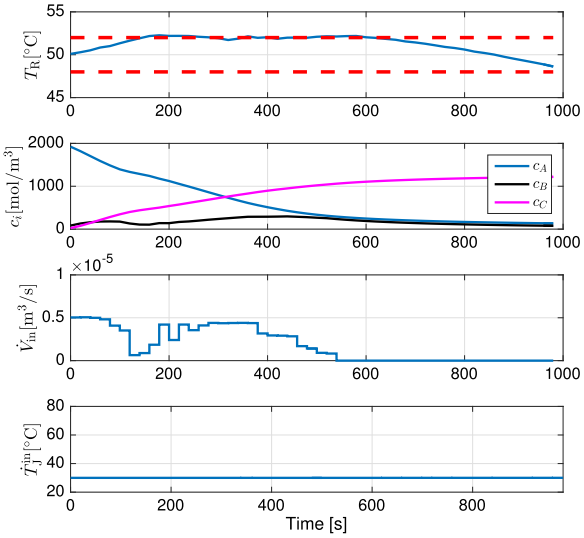

Fig. 12. Reactor temperature, concentrations, input flow and setpoint of the thermostat, experimental results for standard NMPC with no uncertainty in the reaction parameters.

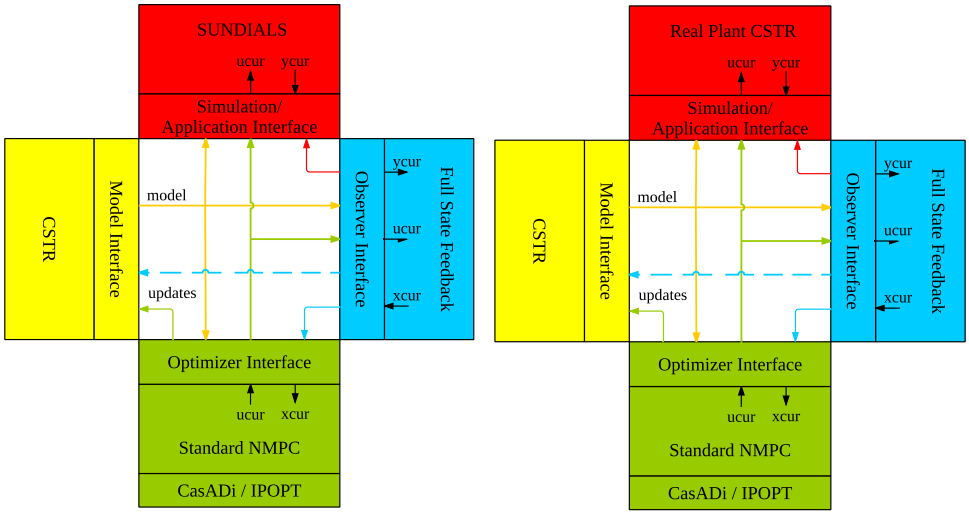

Fig. 11. do-mpc configurations used to obtain the simulation results (left) and the experimental results (right) of standard NMPC applied to the CSTR under consideration.

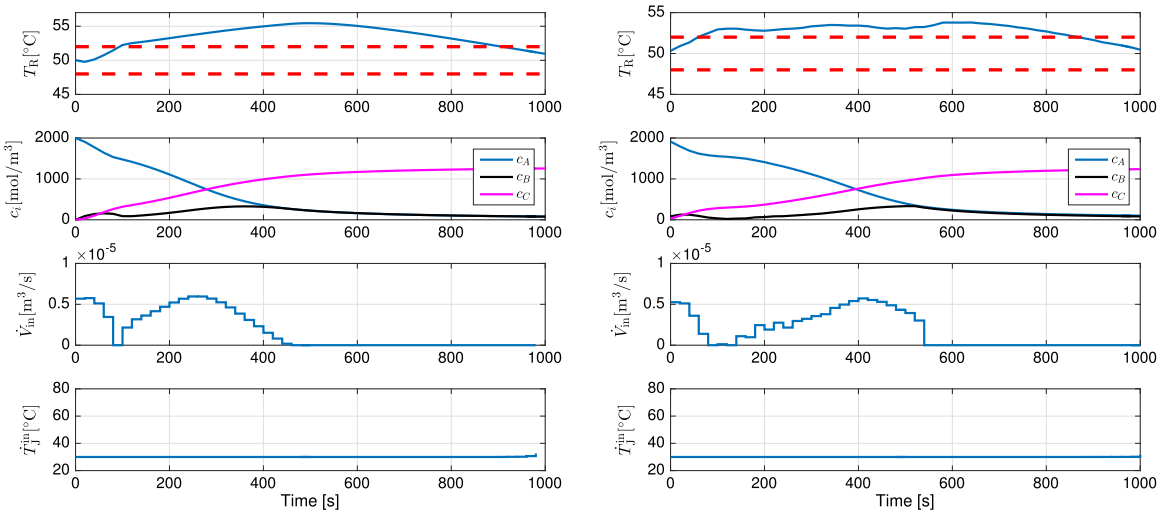

Fig. 13. Reactor temperature, concentrations, input flow and setpoint of the thermostat for standard NMPC when the uncertain parameters are 25% larger than their nominal values. The left plot shows simulation results and the right plot shows experimental results.

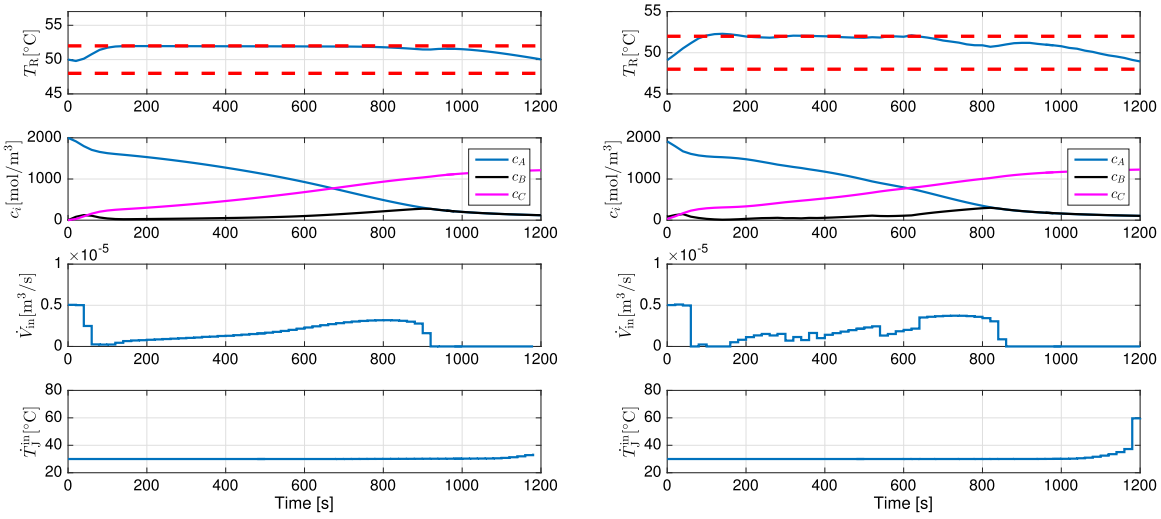

Fig. 14. Reactor temperature, concentrations, input flow and setpoint of the thermostat for multi-stage NMPC when the uncertain parameters are 25% larger than their nominal values. The left plot shows simulation results and the right plot shows experimental results.

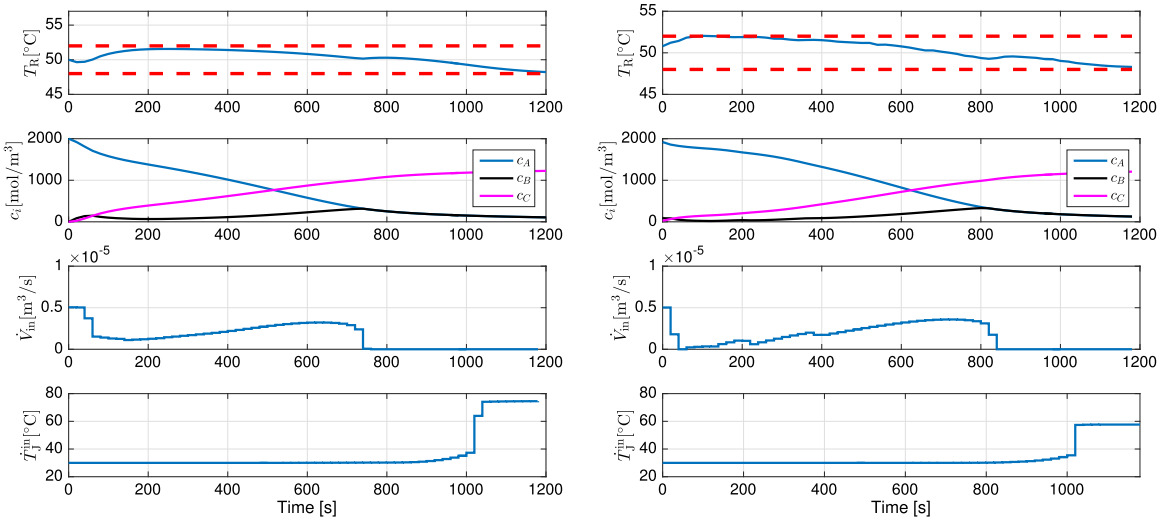

Fig. 15. Reactor temperature, concentrations, input flow and setpoint of the thermostat for multi-stage NMPC with no uncertainty on the reaction parameters. The left plot shows simulation results and the right plot shows experimental results.

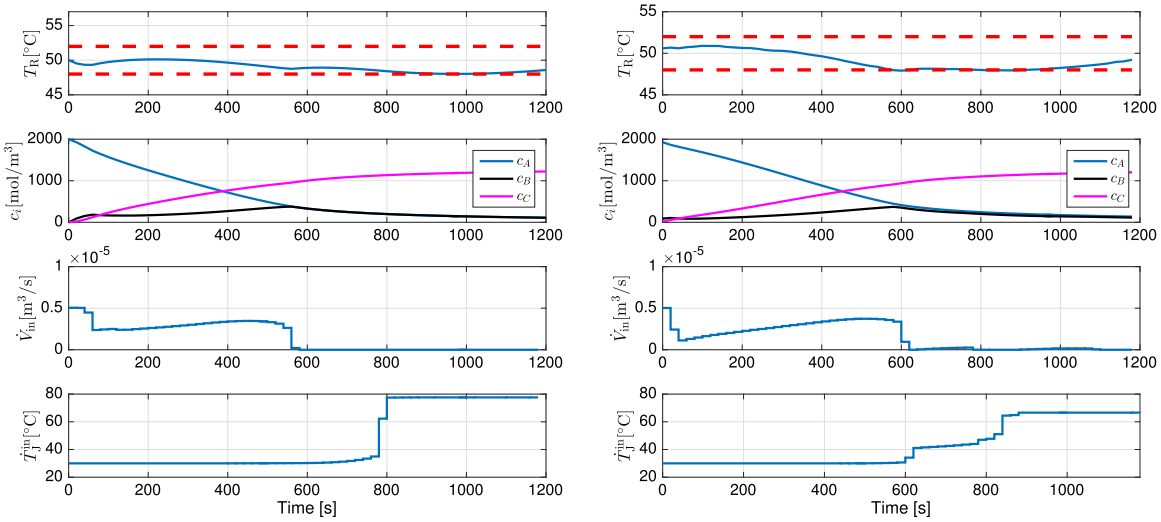

Fig. 16. Reactor temperature, concentrations, input flow and setpoint of the thermostat for multi-stage NMPC when the uncertain parameters are 25% smaller than their nominal values. The left plot shows simulation results and the right plot shows experimental results.

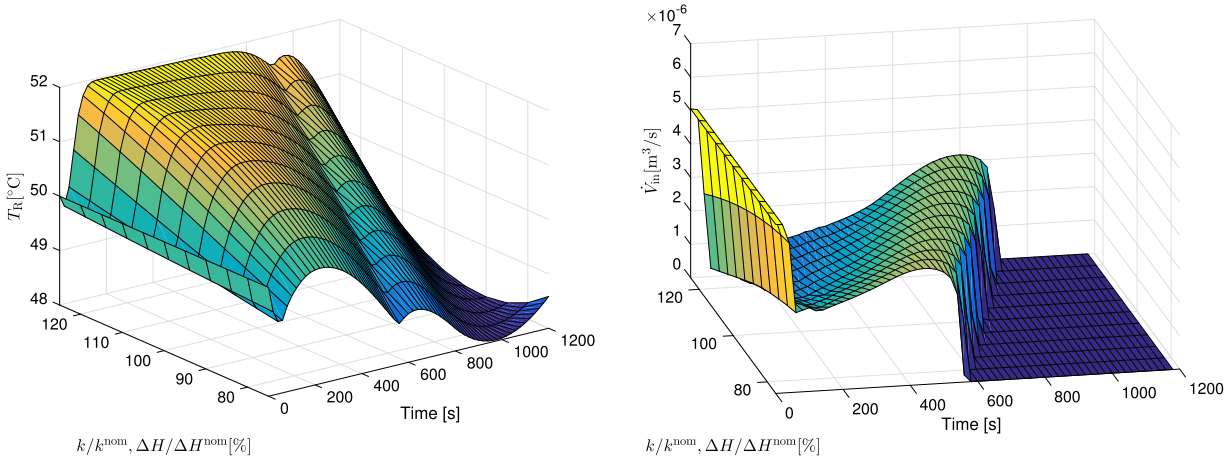

Fig. 17. Simulation results obtained using multi-stage NMPC for different values of the uncertainty on the reaction parameters. The left plot shows the reactor temperature and the right plot shows the input flow. The constraints are satisfied for all the possible values of the uncertainty.

included in the tree), a tree of reachable sets can be built as shown in Lucia, Paulen, and Engell (2014). Furthermore, if the uncertainty is unknown but constant (e.g. an unknown time-invariant parameter), the information that can be gained by the measurements can be used to estimate the uncertainty and a new scenario tree can be generated based on the estimated value and on the confidence intervals that are obtained given noisy measurements as reported in Lucia and Paulen (2014).

For the cases where the estimation or measurement errors are significant, they can be taken into account explicitly in the scenario tree by e.g. the sampled innovations approach presented in Subramanian, Lucia, and Engell (2014) or in Subramanian, Lucia, and Engell (2016).

5. Conclusion

This paper deals with two of the most important obstacles for the widespread use of nonlinear model predictive control in industry: the difficulties to rapidly obtain sustainable NMPC implementations, and handling model uncertainties. A new concept for the modular implementation of NMPC solutions is presented based on four independent modules: model and problem description, optimizer, observer and simulator. The main idea is that any self-implemented or existing algorithm can be used for each module and that they interact via standardized interfaces. If one wants to test a new model or a new optimization algorithm, the only necessary step is to exchange the corresponding module. do-mpc is presented as a realization of such a modular NMPC scheme in which an efficient implementation of multi-stage NMPC is included in order to deal with uncertainty in a systematic manner, an approach which has been shown to provide very good results in practice.

The presented ideas are demonstrated with experimental results, which show that using such a modular implementation it is possible to develop fast and sustainable NMPC solutions which are real-time feasible and take uncertainty explicitly into account.

Acknowledgment

The authors would like to thank Daniel Haßkerl and Sakthi Thangavel for the help with the experimental setup and Joel Andersson for fruitful discussions on the implementation. The research leading to these results has received funding from the European Commission under grant agreement number 291458 (MOBOCON) and from the Deutsche Forschungsgemeinschaft under grant agreement number EN 152/39-1.

References

AlGhazzawi, A., & Lennox, B. (2009). Model predictive control monitoring using multivariate statistics. *Journal of Process Control*, 19, 314–327.

Amrit, R. & Rawlings, J. (2008). Nonlinear model predictive control tools (NMPC tools). Online report. URL: ⟨http://jbrwww.che.wisc.edu/software/mpctools/mpc-tools-doc.pdf⟩.

Andersson, J., Åkesson, J., & Diehl, M. (2012). CasADi – A symbolic package for automatic differentiation and optimal control. , in: Forth, S., Hovland, P., Phipps, E., Utke, J., & Walther, A. (Eds.). (2012). Recent advances in algorithmic differentiation . Berlin: Springer, 297–307.

Bernardini, D. & Bemporad, A. (2009). Scenario-based model predictive control of stochastic constrained linear systems. In *Proceedings of the 48th IEEE conference on decision and control* (pp. 6333–6338).

Campo, P. & Morari, M. (1987). Robust model predictive control. In *Proceedings of the American control conference* (pp. 1021–1026).

Cybernetica (2014). Cybernetica AS. ⟨http://www.cybernetica.no⟩.

Diehl, M., Bock, H., & Schlöder, J. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43, 1714–1736.

Diehl, M., Leineweber, D., & Schäfer, A. (2001). MUSCOD-II Users Manual. IWR Preprint 2001-25.

Herceg, M., Kvasnica, M., Jones, C., & Morari, M. (2013). Multi-parametric toolbox 3.0. In *Proceedings of the European control conference* (pp. 502–510).

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E. et al. (2005). SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31, 363–396.

Houska, B., Ferreau, H., & Diehl, M. (2011). ACADO Toolkit – An open source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32, 298–312.

Houska, B., Ferreau, H., & Diehl, M. (2011). An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47, 2279–2285.

Huang, R., Zavala, V., & Biegler, L. (2009). Advanced step nonlinear model predictive control for air separation units. *Journal of Process Control*, 19, 678–685.

Idris, E., & Engell, S. (2012). Economics-based NMPC strategies for the operation and control of a continuous catalytic distillation process. *Journal of Process Control*, 22, 1832–1843.

IPCOS (2014). IPCOS NV. ⟨http://www.ipcos.com⟩.

Lee, J. H., & Yu, Z. H. (1997). Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica*, 33, 763–781.

Lucia, S., Andersson, J., Brandt, H., Diehl, M., & Engell, S. (2014). Handling uncertainty in economic nonlinear model predictive control: A comparative case-study. *Journal of Process Control*, 24, 1247–1259.

Lucia, S., Finkler, T., & Engell, S. (2013). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23, 1306–1319.

Lucia, S. & Paulen, R. (2014). Robust nonlinear model predictive control with reduction of uncertainty via robust optimal experiment design. In *Proceedings of the 19th IFAC World congress* (pp. 1904–1909).

Lucia, S., Paulen, R., & Engell, S. (2014). Multi-stage nonlinear model predictive control with verified robust constraint satisfaction. In *Proceedings of the 54th IEEE conference on decision and control* (pp. 2816–2821).

Lucia, S., Tatulea-Codrean, A., Schoppmeyer, C., & Engell, S. (2014). An environment for the efficient testing and implementation of robust NMPC. In *Proceedings of the 2014 IEEE multi-conference on systems and control* (pp. 1843–1848).

do mpc (2015). do-mpc: An easy, modular and efficient implementation of robust nonlinear model predictive control. ⟨https://github.com/do-mpc/do-mpc⟩.

Nagy, Z. (2008). OptCon – An efficient tool for rapid prototyping of nonlinear model predictive control applications. In *Proceedings of the AIChE annual meeting* (pp. 16–21).

Patwardhan, R. S., Shah, S. L., & Qi, K. Z. (2002). Assessing the performance of model predictive controllers. *Canadian Journal of Chemical Engineering*, 80, 954–966.

Muñoz de la Peña, D., Bemporad, A., & Alamo, T. (2005). Stochastic programming applied to model predictive control. In *Proceedings of the 44th IEEE conference on decision and control* (pp. 1361–1366).

Pluymers, B., Ludlage, J., Ariaans, L., & Van Brempt, W. (2008). An industrial implementation of a generic NMPC controller with application to a batch process. In *Proceedings of the 17th IFAC World congress* (pp. 982–987).

Qin, S. J., & Badgwell, T. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11, 733–764.

Qt5.1 (2014). Qt5.1 framework. ⟨http://www.qt-project.org⟩.

Schoppmeyer, C. (2013). GEMS – Generic EMBOCON minimal supervisor. ⟨https://github.com/EMBOCONcs/GEMS⟩.

Scokaert, P., & Mayne, D. (1998). Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43, 1136–1142.

Subramanian, S., Lucia, S., & Engell, S. (2014). Economic multi-stage output nonlinear model predictive control. In *Proceedings of the 2014 IEEE multi-conference on systems and control* (pp. 1837–1842).

Subramanian, S., Lucia, S., & Engell, S. (2016). A non-conservative robust output feedback mpc for constrained linear systems. In *Proceedings of the 55th conference on decision and control* (pp. 2333–2338).

Toumi, A., & Engell, S. (2004). Optimization-based control of a reactive simulated moving bed process for glucose isomerization. *Chemical Engineering Science*, 59, 3777–3792.

Wächter, A., & Biegler, L. (2006). On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106, 25–57.

Zagrobelny, N., Ji, L., & Rawlings, J. (2013). Quis custodiet ipsos custodes? *Annual Reviews in Control*, 37, 260–270.