Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

# Teaching Nonlinear Model Predictive Control with MATLAB/Simulink and an Internal Combustion Engine Test Bench

**Martin Keller** [*] **Dennis Ritter** [*] **Lukas Schmitt** [*]
**Severin Hänggi** [**] **Christopher Onder** [**] **Dirk Abel** [*]
**Thivaharan Albin** [*,**,***]

[*] *Institute of Automatic Control, RWTH Aachen University, Aachen, Germany (e-mail: M.Keller@irt.rwth-aachen.de, D.Ritter@irt.rwth-aachen.de, L.Schmitt@irt.rwth-aachen.de, D.Abel@irt.rwth-aachen.de)*
[**] *Institute for Dynamic Systems and Control, ETH Zurich, Zurich, Switzerland (e-mail: shaenggi@idsc.mavt.ethz.ch, onder@idsc.mavt.ethz.ch)*
[***] *Embotech AG, Zurich, Switzerland (e-mail: albin@embotech.com)*

**Abstract:** Model Predictive Control (MPC) is used for more and more applications in an industrial context. The applications are characterized by increasing complexity while the available computation time is getting smaller and smaller. MPC is the most important advanced control technique with even increasing importance. Hence, this topic should be covered in control lectures during the academic studies in order to prepare students for their future work. For the successful implementation of MPC algorithms, knowledge from multiple disciplines is crucial and needs to be taught. Besides teaching knowledge in classical control theory, especially fundamentals in the fields of modeling, simulation and numerical optimization are required for understanding MPC. Programming skills are inevitable to apply the concept in real-world applications.
This paper presents a concept for teaching MPC from the theory to the application to real-world systems. Details about the lectures covering the relevant topics are given. In the hands-on exercises, students implement their own linear as well as nonlinear MPC in MATLAB/Simulink. As example application in the exercises, the air path of a turbocharged diesel engine with high pressure exhaust gas recirculation is investigated. At the end of the semester, students can test their developed controllers on a real diesel engine test bench and compete against each other for the best control performance.

*Keywords:* Nonlinear Model Predictive Control, Modeling, Optimization, Engine Control

## 1. INTRODUCTION

Model Predictive Control (MPC) concepts are very popular in academia as well as industrial applications. They explicitly use a mathematical description of the system to be controlled to calculate the control inputs. Nowadays, MPC concepts are used more and more widely for controlling complex systems. The advantages of MPC arise in a systematic synthesis of the control algorithms and a possible higher control quality with less need for heuristic tuning. Most applications are continuously further developed, such that the requirements on the closed-loop controller are increasing. Classical control approaches, such as PID control, require a lot of controller tuning to cope with the respective systems. The internal combustion engine is a good example for the ongoing research and development. Until the 1960s engines where controlled purely mechanically. Recent gasoline engines are equipped with 15 to 25 sensors and about 6 manipulated variables. The same holds for diesel engines consisting of about 20 sensors and up to 9 manipulated variables (Isermann (2014)). To cope with strict emission legislations even more complex actuators are used such as multiple injections per cycle (Tschanz et al. (2014), Guzzella and Onder (2010)). For these systems, the design of conventional controllers based on PID control and look-up tables is suboptimal concerning performance and very time consuming due to extensive test-bench measurements for controller tuning. If only one component is changed or replaced, the controllers need to be retuned and validated which is again quite costly.

Nevertheless, there are some advantages of conventional PID controllers. They are tunable without much system knowledge and have low computational demands for processing the algorithm. In addition, the controller can be designed without a system model. With more and more powerful computers and electronic control units (ECU), the computational demand of optimization-based algorithms can be handled.

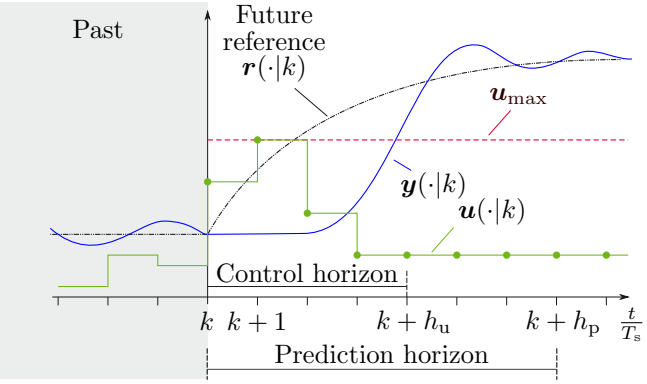


Fig. 1. Working principle of MPC

MPC is a popular class of control concepts within the group of model-based control. The working principle of MPC is depicted in Fig. 1. In each sampling step, the MPC algorithm solves an optimization problem consisting of a cost function to be minimized and possibly some constraints. During the optimization, a process model is used in order to predict the future system behavior $\boldsymbol{y}(\cdot|k)$ over the prediction horizon $h_{\mathrm{p}}$. The result is an optimal control input trajectory $\boldsymbol{u}(\cdot|k)$ from which only the first value is fed to the system. In the subsequent sampling step, the optimization problem is solved again from scratch over a shifted prediction horizon. Main components for successful development of MPC are a) the proper formulation of the optimization problem b) a suitable model of the system dynamics and c) the numerical solution of the resulting optimization problem.

The objective behind the development of MPC was to tackle multiple-input-multiple-output (MIMO) systems with constraints especially in the chemical and oil industry (Morari and Lee (1999)). Recent algorithms and more powerful processors made it possible to compute more complex optimization problems arising in the MPC context in less time and increased the field of possible application (Gros et al. (2016)). Nowadays, even nonlinear MPC algorithms can be solved within microseconds (Quirynen et al. (2015)) and made the MPC even more attractive for the industry. As MPC is the most popular advanced control technique in industry since two decades (Qin and Badgwell (2003)), universities around the world have modified the curricula and developed lectures to teach MPC to their students.

There already exist publications treating the subject of teaching MPC. In Honc et al. (2016) a concept for teaching linear MPC with a level control example as application is presented. The publication Shariati and Abel (2016) presents an outline for a two day MPC workshop focusing on linear MPC as well, with additional application lectures. The paper at hand presents an approach for teaching MPC, ranging from linear to nonlinear MPC and the application to fast systems with small available computation time. The teaching material covers theory to hands-on programming exercises. As theory and practice go hand in hand, it is necessary to teach not only the theory and fundamentals of MPC but also show the students how to implement the MPC algorithm for the use in a real-world application.

The presented material in this paper is based on the lecture "Model Predictive Control of Energy Conversion Systems". The lecture is held jointly by the Institute of Automatic Control at RWTH Aachen University, Germany as well as by the Institute for Dynamic Systems and Control at ETH Zurich, Switzerland. It was first held in the summer term 2017 at RWTH Aachen University. At ETH Zurich it was held first in the spring term 2018. At both institutions, the lecture is offered on an annual basis for Master Course students. The participating students are from the field of mechanical engineering, energy system engineering and also robotics. They are familiar with basic concepts of closed-loop control. Most of them have taken the fundamental and one more advanced control lecture. Additionally, they are familiar with programming and Matlab/Simulink at this stage. The lecture is based on the textbook Albin (2020) where the algorithmic details of the various algorithms are explained. In Albin (2020), also the example system is introduced in more detail and the code used for the control algorithm is also available there.

The intention of the paper is to give an overview on the lecture and thus present a possible approach to teach the topic to students. A special focus is set on the example system, which is used as a hand-on experience example. No attempt is made to explain MPC and the various algorithms comprehensively. For details and fundamentals of MPC and numerical optimization, the reader is referred to e.g. Albin (2020), Rawlings et al. (2017), Maciejowski (2002), Boyd and Vandenberghe (2004) or Nocedal and Wright (2006).

## 2. GOALS OF THE LECTURE

Before developing a new course, it is crucial to define the learning goals of the lectures and exercises. For the presented lecture and exercises, the following goals are defined:

- Benefits of model-based control in general and MPC in particular
- Fundamentals of optimization needed for MPC
- Formulation of the linear and nonlinear MPC
- Solving Quadratic Programs (QP) for linear MPC
- Fast solution of Nonlinear Programs (NLP) for nonlinear MPC
- Requirements and methods for implementation and application to real-world systems
- Applying the theory in hands-on programming exercises to real-world problems

The above mentioned learning goals lead to the contents of lectures and exercises depicted in Table 1. The lectures are divided roughly into four parts: a)Introduction and fundamentals b) Linear MPC c) Nonlinear MPC d) Application to Energy Conversion Systems.

The first part covers the lectures 1−3. The lecture 1 gives an introduction into the topic of MPC. Advantages are outlined along with the historical development of MPC. Additionally, energy conversion systems are introduced as a field where many complex nonlinear systems need to be controlled. The lecture 2 gives an overview of model-based control approaches. It is introduced how a model can be

Table 1. Content of lectures and exercises

| # | **Lecture** | **Exercise** |
|---|---|---|
| 1 | Introduction to MPC | Introduction to MATLAB/Simulink |
| 2 | Model-based control | System analysis of example system |
| 3 | Fundamentals of optimization | PID synthesis for example system |
| 4 | Linear MPC − optimization problem | Linear MPC − unconstrained |
| 5 | Linear MPC − formulation | Linear MPC − constrained |
| 6 | From linear to nonlinear MPC | Linear MPC − offset-free and deadtime handling |
| 7 | Nonlinear MPC − sequential quadratic programming (SQP) | Solving an NLP with SQP |
| 8 | Nonlinear MPC − discretization methods | Introduction to CasADi Toolbox |
| 9 | Stability of MPC | Implementation of direct shooting methods |
| 10 | Model-based engine control | Group work NMPC 1/4 |
| 11 | Turbocharger control | Group work NMPC 2/4 |
| 12 | Combustion control | Group work NMPC 3/4 |
| 13 | Organic Rankine Cycle control | Group work NMPC 4/4 |
| 14 | Wind energy control | Competing challenge at engine test bench |

used to synthesize a classical PID controller. Its drawbacks when treating MIMO systems with cross-couplings, strong nonlinearities and dead times are mentioned. The lecture 3 *Fundamentals of optimization* gives an overview of the different classes of optimization problems, and describes the properties and challenges of convex and non-convex optimization. Furthermore, the necessary and sufficient conditions of optimality for constrained nonlinear programs (NLP) are covered.

The second part (lectures 4−6) treats the concept of linear MPC. The lecture 4 introduces the unconstrained MPC controller and shows the similarity to linear state-feedback controllers. The incorporation of system constraints for linear systems is explained in detail and the resulting quadratic programs (QP) are derived. The differences between sparse and dense formulation of the optimization problem are addressed and benefits as well as downsides of each formulation are discussed. The lecture 5 details the formulation of a linear MPC problem for practical applications. To achieve offset-free control of the controlled variables, the observer-based disturbance estimation is described as a method to cope with model mismatch and influences of disturbances. Constraint softening techniques are introduced to ensure feasibility of the resulting optimization problem. Systems with dead times as well as reference tracking for non-square (in particular overactuated) systems are treated as well. In the lecture 6, the topic of linear time-variant (LTV) MPC is covered as transition from linear MPC to nonlinear MPC. The LTV MPC is introduced as one concept for handling slightly nonlinear systems.

The third part (lectures 7−9) covers the concept of nonlinear MPC. The main difference between linear and nonlinear MPC is the in general nonconvex NLP that arises due to nonlinear system dynamics or nonlinear constraints. The lecture 7 details how these NLPs can be solves in an MPC context. As one suitable solution technique, the Sequential Quadratic Programing (SQP) is introduced. The solution of the NLP via sequential QP approximations is covered by first explaining the procedure for the simpler equality constrained case. Afterwards, the more general inequality constrained case is explained.

The SQP method is introduced along with globalization strategies, such as step size control and regularization of negative-definite Hessian matrices. Additionally, the Gauss-Newton approximation of the Hessian is detailed. Beside the treatment of theory and algorithms, new state-of-the-art NLP solvers like FORCES (Zanelli et al. (2017)) are mentioned. The real-time iteration scheme, as discussed in (Diehl et al. (2005); Rawlings et al. (2017)), is introduced to reduce the computational burden. As the energy conversion systems are typically modeled by physical equations, the model is present in continuous-time dynamics. Therefore, the lecture 8 covers different approaches for the discretization of Optimal Control Problems (OCP). A special focus is set on the single-shooting and multiple-shooting discretization. The lecture 9 details the stability of MPC algorithms. The linear as well as the nonlinear case are treated.

In the forth and last part, some application examples from ongoing research projects in the field of energy conversion systems are presented. The focus of the last part is to highlight the important steps towards implementing a MPC on a real-world application. The lectures conclude with advantages and drawbacks of MPC for the respective application example.

Within the exercises, the students implement several control algorithms for one exemplary system. That way, a consistent example exists throughout the entire exercises and the students do not have to get familiar with a new system at the beginning of each exercise. The example system treated in all the exercises is the air path of a turbocharged diesel engine with high pressure exhaust gas recirculation (EGR). All the exercises are based on a reduced-order model of the aforementioned diesel engine air path. The reasons for choosing this type of system for the exercises are explained in Section 3.

The exercises are divided into four parts, such as the lecture. In the first part (exercises 1−3), the students get familiar with MATLAB/Simulink and analyze the system dynamics of the example system. Amongst others, stability, non-minimum phase behavior, controllability and cross-couplings via Relative-Gain Array are investigated. The exercise 3 covers the conventional PID controller synthesis where the students design decentralized PID controllers to control charging pressure and EGR rate. The students get to know the difficulties of designing a conventional controller for challenging systems with significant cross-couplings.

The second part (exercises 4−6) covers the implementation of a linear MPC in MATLAB/Simulink. The students learn the essentials and methods for programming their

own MPC algorithm. First, in exercise 4 an unconstrained linear MPC controller is implemented. The focus lies in the generic, algorithmic implementation of the prediction matrices in dependence of the chosen prediction and control horizons considering the number of system inputs, states and outputs. Based on this, the exercise 5 covers the inclusion of constraints on the actuated values and on the system states. The necessary matrices for the QP solver are derived and subsequently it is solved using a QP solver. The excercise 6 treats the advanced formulation of the optimization problem. The observer for state and disturbance estimation are implemented to achieve offset-free control.
In the third part (exercises 7−9), the transition from linear MPC to nonlinear MPC is made. After formulating the nonlinear OCP direct solution methods are employed. Therefore, the students write their own discretization method using single as well as multiple shooting to derive the final NLP. In the exercises, the students implement their own SQP method for solving the NLP. Furthermore, the open-source toolbox CasADi (Andersson et al. (2018)) is presented that is used for calculating the required derivatives and sensitivities (Hessian and Jacobian matrices, etc.). The toolbox can be included in the MATLAB framework. Hence, only a few new commands need to be explained to the students.
The last part of the exercises (exercises 10−14) are carried out as group work exercises. Typically groups of 3-4 students are built. In these exercises, step-by-step a MPC controller is implemented by the students which can be deployed on a real-time prototyping hardware. Finally, the various MPC controllers from the students are applied on the real engine test bench. At the final event, there is a competition for the best performing controller on the engine test bench. Details about the group work are given in Section 6 and 7.

All the lectures and exercises start with a recap of the previous lecture and exercise, followed by the educational objectives of the current lecture and exercise, respectively. At the end of each chapter/topic, a short summary is given by the lecturer.

## 3. REQUIREMENTS FOR THE EXAMPLE SYSTEM

As an example system the air path control of a turbocharged diesel engine with high pressure EGR is chosen. It has many properties that makes it well suited for teaching MPC, as explained in the following.

A suitable example system should have the following characteristics and properties:

- MIMO system with significant cross-couplings
- Nonlinearity
- Dead times
- Non-minimum phase behavior
- Input/output/state constraints
- Physics-based modeling

In order to become acquainted with the benefits of MPC the example system should be a MIMO system with significant cross-couplings and nonlinearity. For the air path system a decoupled control synthesis with design of decoupling terms does not provide sufficient control

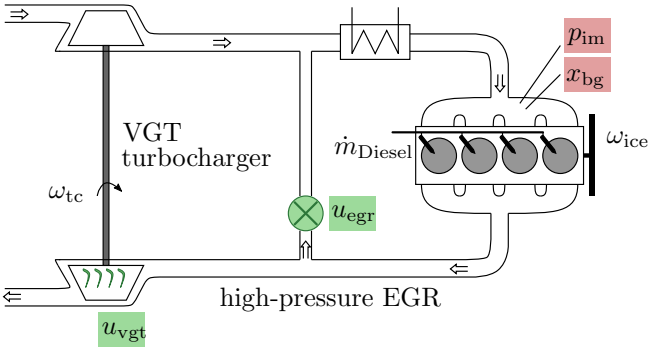


Fig. 2. System setup of the investigated air path of a turbocharged diesel engine with high pressure EGR: Manipulated variables (green) are VGT actuation $u_{\mathrm{vgt}}$ and EGR valve position $u_{\mathrm{egr}}$; controlled variables (red) are intake manifold pressure $p_{\mathrm{im}}$ and burnt gas ratio $x_{\mathrm{bg}}$

performance. Furthermore, the system should have some time delays. For conventional control, this results in a more conservative controller design which does not allow for highly transient operation. There exist model-based methods, besides MPC, for handling systems with time delays, e.g. Smith predictor (Normey-Rico and Camacho (2007)). The same holds for non-minimum phase systems for which the controller tuning has to be conservative as well. As all real-world examples have limited actuator power and some constraints on the outputs, this characteristic is not restrictive for the choice of an example system. In addition, the underlying system should be modeled in a physics-based manner, such that the students can follow the modeling approach. Last but not least, as we intend to show the control performance on a real test bench, the model should not be an artificial, theoretical, mathematical example system but should reflect a real-world application.

All in all, the example system should be difficult to control with conventional PID controllers, such that the benefits of model-based approaches and in particular MPC become obvious.

## 4. EXAMPLE SYSTEM – MODELING

The setup of the investigated example system is depicted in Fig. 2. The air path of the diesel engine is equipped with a high-pressure exhaust gas recirculation path and a turbocharger with variable geometry turbocharger (VGT). The only two inputs considered in the context of the exercises are the position of the VGT guide vanes $u_{\mathrm{vgt}}$ and the position of the EGR valve $u_{\mathrm{egr}}$. Both actuator positions are normalized such that their values are always between 0 and 1. For the VGT, the position $u_{\mathrm{vgt}} = 0$ corresponds to the position with minimal boost pressure. The EGR valve is fully closed at position $u_{\mathrm{egr}} = 0$, resulting in no exhaust gas being recirculated. The challenge of this control task is to adjust the pressure $p_{\mathrm{im}}$ and the burnt gas ratio $x_{\mathrm{bg}}$ in the intake manifold of the engine precisely and fast. A common approach to model the engine's air path is based on physical, first-order principles. More details about control-oriented modeling and control for the air path of engines can be found in Albin (2020). Typically, state-space models are used for the controller-internal model of the MPC

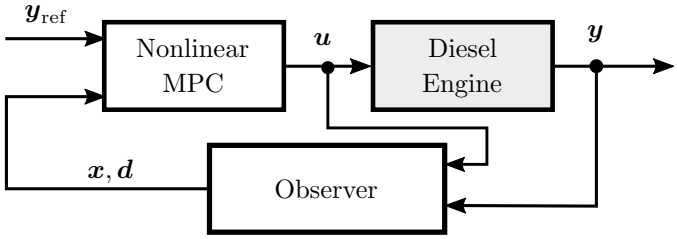


Fig. 3. Control structure to be implemented by the students during the exercises

algorithm (Qin and Badgwell (2003)). Thus, the controller-internal model for the air-path system was formulated as a state-space system. Furthermore, the model should be smooth, i.e. continuously differentiable, such that it can be embedded in gradient-based optimization methods.

The overall control-oriented process model has the following general state-space representation:

$$\begin{aligned} \dot{\boldsymbol{x}} &= f(\boldsymbol{x}, \boldsymbol{u}), \\ \boldsymbol{y} &= g(\boldsymbol{x}). \end{aligned} \tag{1}$$

The two manipulated variables $\boldsymbol{u}$ are the respective positions of the VGT and the EGR valve

$$\boldsymbol{u} = \left[\, u_{\mathrm{vgt}} \;\; u_{\mathrm{egr}} \,\right]^{\mathrm{T}}. \tag{2}$$

The two controlled outputs $\boldsymbol{y}$ are intake manifold pressure and burnt gas ratio

$$\boldsymbol{y} = \left[\, p_{\mathrm{im}} \;\; x_{\mathrm{bg}} \,\right]^{\mathrm{T}}. \tag{3}$$

There are five different system states that are considered. They are given by the turbocharger speed $\omega_{\mathrm{tc}}$, the intake manifold pressure $p_{\mathrm{im}}$, the exhaust manifold pressure $p_{\mathrm{em}}$, the oxygen mass fraction in the intake manifold $F_{\mathrm{im}}$ and in the exhaust manifold $F_{\mathrm{em}}$. The burnt gas ratio can be determined by

$$x_{\mathrm{bg}} = 1 - \frac{F_{\mathrm{im}}}{F_{\mathrm{air}}}. \tag{4}$$

In addition, the process is subject to considerable dead time which are acting on the outputs due to sensor delays. For the intake manifold pressure, the dead time is identified to $\tau_{\mathrm{p}} = 0.1\,\mathrm{s}$ and $\tau_{\mathrm{egr}} = 0.4\,\mathrm{s}$ for the burnt gas ratio. All in all, the model reproduces the system's characteristics in a sufficiently detailed manner while being smooth and real-time feasible.

## 5. EXAMPLE SYSTEM - CONTROLLER DESIGN

For controlling the charging pressure and the burnt gas ratio, several controllers are implemented during the exercises. First, a PI-based controller is implemented. Several pitfalls are illustrated, such as the need for anti-windup schemes and decoupling terms. Subsequently, a liner MPC algorithm is implemented which works well at one operating point. It impairs concerning control quality when the entire operation region is examined. Finally, a nonlinear MPC controller is implemented by the students during the exercises. The control concept consists of the NMPC algorithm and an observer which is used to achieve offset-free control, see Fig. 3. For the nonlinear MPC, the students establish their own optimal control problem (OCP) based on the following general description and adapt it to meet their individual control goals. The following OCP represents one possibility:

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} = \int_0^{t_{\mathrm{ch}}} \left( ||\boldsymbol{y}_{\mathrm{ref}}(t) - \boldsymbol{y}(t)||^2_{\boldsymbol{Q}} + ||\dot{\boldsymbol{u}}(t)||^2_{\boldsymbol{R}} \right) \mathrm{d}t \tag{5}$$

$$\begin{aligned} \text{s.t.} \quad & 0 = \boldsymbol{x}(0) - \boldsymbol{x}_0, \\ & 0 = f(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \dot{\boldsymbol{u}}(t)) \quad \forall t \in [0, t_{\mathrm{ch}}], \end{aligned} \tag{6}$$

$$\begin{aligned} & \boldsymbol{y}_{\mathrm{lb}} \le \boldsymbol{y}(t) \le \boldsymbol{y}_{\mathrm{ub}} \quad \forall t \in [0, t_{\mathrm{ch}}], \\ & \boldsymbol{x}_{\mathrm{lb}} \le \boldsymbol{x}(t) \le \boldsymbol{x}_{\mathrm{ub}} \quad \forall t \in [0, t_{\mathrm{ch}}], \\ & \boldsymbol{u}_{\mathrm{lb}} \le \boldsymbol{u}(t) \le \boldsymbol{u}_{\mathrm{ub}} \quad \forall t \in [0, t_{\mathrm{ch}}], \\ & \dot{\boldsymbol{u}}_{\mathrm{lb}} \le \dot{\boldsymbol{u}}(t) \le \dot{\boldsymbol{u}}_{\mathrm{ub}} \quad \forall t \in [0, t_{\mathrm{ch}}]. \end{aligned} \tag{7}$$

Here $\boldsymbol{x}(t)$ denotes the differential states, $\dot{\boldsymbol{x}}(t)$ its time derivatives, $\boldsymbol{u}(t)$ determines the control inputs and $\dot{\boldsymbol{u}}(t)$ its time derivative. As initial state vector, the current state estimate $\boldsymbol{x}_0$ is used. The cost function for the control horizon $t_{\mathrm{ch}}$ consists of the tracking costs for the charging pressure and burnt gas ratio as well as costs for changing the control input. The nonlinear dynamics are considered in (6). The path constraints in (7) consist of simple bounds for the inputs, states and outputs. For discretization of the OCP and to derive a nonlinear program (NLP), the students implement a direct method (single shooting or multiple shooting) (see e.g. Bock and Plitt (1984)). In order to solve the NLP, the student first use the interior-point NLP solver IPOPT (Wächter and Biegler (2006)). For solving the NLP in real time on the RCP hardware, the students implement their own SQP algorithm.

## 6. REAL-TIME IMPLEMENTATION OF THE MPC ALGORITHM IN THE EXERCISES

According to the strong focus on real-world application of this lecture, the students design and implement their own MPC algorithm on the engine hardware. The major challenges to deal with are real-time feasibility, model-plant mismatch and the demanding system behavior including time delays. For validating the NMPC algorithm in simulation, a high-fidelity mean-value model (MVM) is provided which captures the major characteristics of the engine. Besides support during exercises and consultation hours, the current state of the implemented algorithms can be sent to the teaching assistants for review and additional hints.

In the following, the real-time NMPC implementation is detailed. For educational purposes, the choice of methods within the class of NMPC is deliberately left to the students. As a consequence, a thorough consideration of advantages and disadvantages of each method in the context of real-time and real-world implementation is encouraged. Still, a guideline is given to the students, such that with only little adjustments and in a reasonable amount of time, a first working NMPC can be implemented.

To start with, a single NMPC step is implemented. The focus is on arriving at a basic NMPC formulation employing the direct approach to solve the OCP. Therefore, starting from the OCP formulation in (5) − (7), the students implement a direct solution method using CasADi, single or multiple shooting, combined with a numerical integration scheme of their choice for discretizing the aforementioned OCP. In order to solve the resulting NLP, a SQP algorithm employing a damped Newton's method and exact Hessian is implemented in MATLAB. The necessary derivatives

for the gradient-based optimization procedure, namely the Hessian of the Lagrangian, the gradient of the cost function as well as the Jacobians of the constraints, are implemented as CasADi functions. As a result, a single NMPC step can be calculated. With this result, a parameter study can be conducted by the students. First, the convergence behavior of the SQP algorithm in dependence of the number of SQP steps is investigated. Secondly, the resulting simulation time is considered, depending on the number of SQP steps, the length of the prediction horizon, the sampling time and the number of integration steps within a discretization step. As the next step, the MATLAB implementation has to be transferred to Simulink for automatic code generation for the RCP hardware. A provided Simulink SQP template, which contains automatically generated s-functions for the CasADi functions used for the generation of sensitivity information, needs to be completed. A comparison between the MATLAB and Simulink results verifies a correct implementation. With this working single NMPC step, the control loop can be closed to achieve a first closed-loop NMPC algorithm. Analogously to previous exercises, an observer is required to estimate the states and the disturbances for offset-free tracking. Additionally, a time delay compensation must be implemented. Again, with this modifications, the algorithm must be tuned accordingly.

In the following group work session, students are encouraged to implement more advanced formulations presented in the lecture or in literature. Because of a strictly modular implementation, individual methods can be interchanged easily. For the SQP algorithm, this includes for example Hessian approximations, such as Gauss-Newton or BFGS, or the introduction of a step length control using e.g. Armijo or Wolfe conditions. Additionally, students are encouraged to compare explicit and implicit numerical integration schemes with respect to the number of integration steps per discretization interval and the resulting computation time. As a result, various flavors of NMPC real-time implementations establish an exciting premise for the subsequent testing at the engine test bench.

## 7. EXPERIMENTAL VALIDATION OF THE STUDENTS' CONTROL ALGORITHMS

A working controller in simulation still has to prove its fitness on a real system, where computation power is limited and robustness is crucial. The last step for the students is to tune their controllers on the diesel engine, which was used to derive the provided engine models. The engine characteristics are depicted in Table 2.

Table 2. Diesel engine characteristics

| | |
|---|---|
| Engine Type | VW TDI 2.0 − 475 NE Industrial Engine |
| Number of Cylinders | 4 |
| Displacement Volume | 1.968 liter |
| Compression Ratio | 16.5 |

For the controller testing, the engine is operated at a constant engine speed and injected diesel mass. All air-path actuators except for the ones controlled by the students are held constant. In Fig. 4, the engine test bench control structure is illustrated. It consists of two RCP systems, both programmable via MATLAB/Simulink. The first RCP system consists of a dSPACE MicroAutoBox II together with a dSPACE RapidPro signal conditioning and power stage. It provides a platform for the basic engine control tasks, engine operation and surveillance. A second RCP system consisting of a dSPACE MicroLabBox is used exclusively for the implementation of the students' nonlinear model predictive air-path control approach. It provides the necessary computation power and is connected to the MicroAutoBox II via an Ethernet interface.

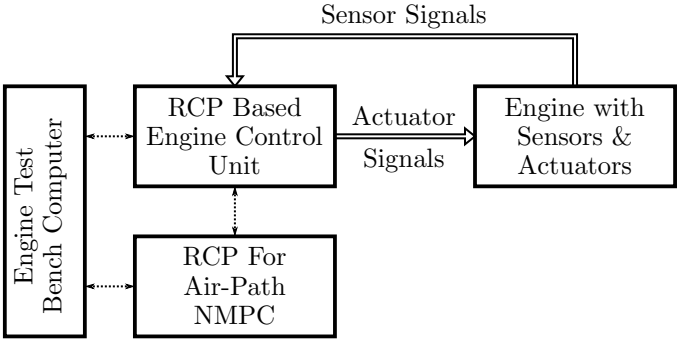


Fig. 4. Engine test bench operation and control scheme.

Using a different hardware for the standard engine control tasks and the NMPC algorithms developed by the students offers several advantages. Amongst others, it guarantees the safe engine test bench operation in case of a task overrun caused by the students' controller. Furthermore, the hardware processor turnaround time is a direct measurement of the students' control algorithm computational demand and can be used for the algorithm performance evaluation. In addition, the real-time capability of the students' control algorithms can already be assessed during the exercises in the class room. Both dSPACE RCP systems are connected to the test bench operation computer via an Ethernet interface. dSPACE software toolchain provides auto-code-generation directly from MATLAB/Simulink, which allows to automatically deploy the students' controllers to the real-time hardware.

On the engine test bench, students get feedback about their controllers computational demand and they get the chance to tune their algorithms to fit the real hardware. For the competition, a reference trajectory is defined, which has to be tracked by the students' controller as accurately as possible. This reference trajectory is illustrated in Fig. 5. In order to evaluate the control algorithm's tracking performance, a cost function is defined, which is implemented on the RCP system to track the performance in real time:

$$J_e(k) = a \cdot \sum_{i=1}^{k} \left(p_{\mathrm{im}}(i) - p_{\mathrm{im,ref}}(i)\right)^2 \tag{8}$$

$$+ b \cdot \sum_{i=1}^{k} \left(x_{\mathrm{bg}}(i) - x_{\mathrm{bg,ref}}(i)\right)^2 \tag{9}$$

$$+ c \cdot \sum_{i=1}^{k} \Delta u_{\mathrm{vgt}}(i)^2 + d \cdot \sum_{i=1}^{k} \Delta u_{\mathrm{egr}}(i)^2 \tag{10}$$

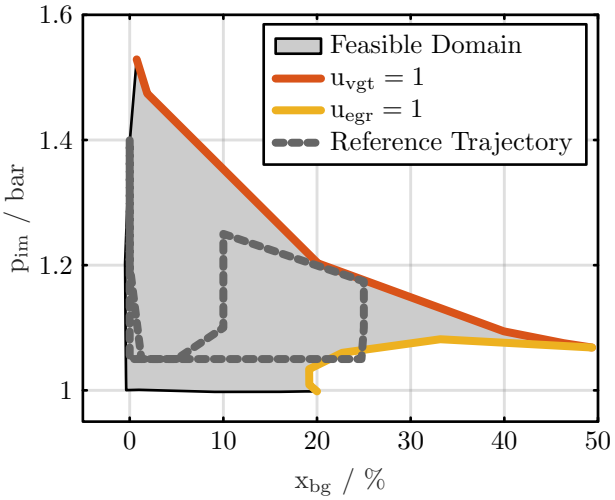


Fig. 5. Feasible domain of the reference signals, with highlighted limits of the EGR and the VGT actuators and chosen reference trajectory for the student controller validation.

$$+ e \cdot \sum_{i=1}^{k} T_{\text{task}}(i) \tag{11}$$

$$+ f \cdot \sum_{i=1}^{k} \left(T_{\text{task}}(i) \geq T_{\text{s}}\right). \tag{12}$$

The controller evaluation cost $J_e(k)$ penalizes the tracking error in (8) and (9), fast actuator position changes like oscillations (10), the computational burden of the algorithm (11) and the number of task overruns (12).
For the competition, a layout for the calibration software (ControlDesk) is designed, which gives an overview of the students' control algorithm parameters and allows to follow the controller performance in real time. Figure 6 shows this layout after a student controller was applied to track the reference trajectory. Two different controllers are investigated, one where the reference trajectory is known in advance and one which just knows the recent reference value. As expected, the non-causal controllers have a better control performance, as depicted in Fig. 7.

During the competition, the performance cost function in (8) – (12) is calculated for each group. The group with the best control result wins the competition and a gift coupon for pizza.

## 8. ORGANIZATION AND EVALUATION

Each lecture and exercise unit has a duration of 90 min, respectively. The weekly work load with preparation and wrap-up after the course leads to an overall average work load of about eight hours per week. After passing the oral exam at the end of the semester, the students obtain five European Credit Transfer System (ECTS) points within their curriculum.

The application lectures are taught by researchers from the specific fields. Thereby, the students obtain insider information from first hand. The encountered difficulties and their overcoming are explained and taught to the students. The exercises are held by two research assistants

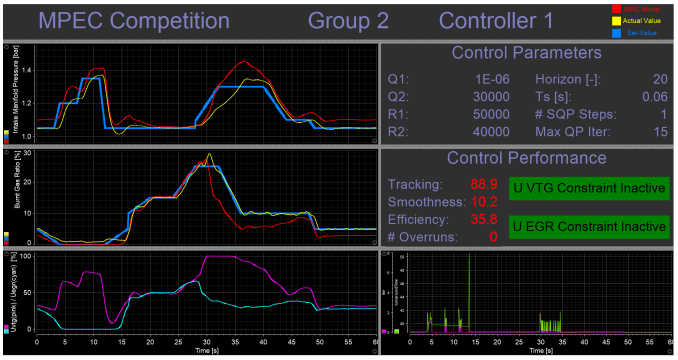


Fig. 6. Real-time layout connected to the RCP, showing the competition results of a causal student controller for the tracking task.

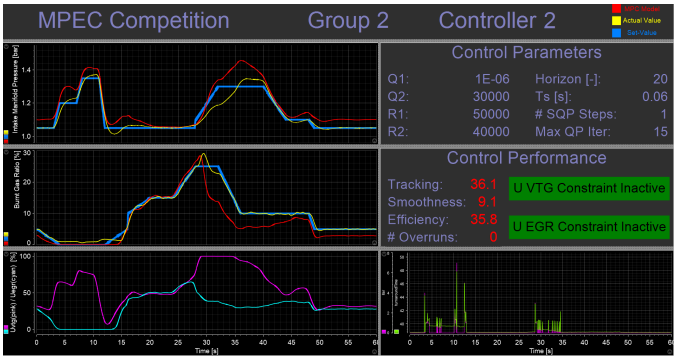


Fig. 7. Real-time layout connected to the RCP, showing the competition results of a non-causal student controller for the tracking task.

in the institute's computer room. The student can ask their questions during the exercises and also at consultation hours during the week. That way, the students can resolve their open questions already during the semester.

During the lecture PowerPoint slides are used for teaching. The textbook (Albin (2020)) serves as lecture notes. For the exercises, work sheets and explaining PowerPoint slides are handed out. Furthermore, a proposal for solution is given to the students as MATLAB file.

All in all, the evaluation of the lectures and exercises by the students showed that the concept is well received by the participating students. In the summer semester 2019, the lecture and the exercise were evaluated by the students with an average grade of 1.3 (lecture) and 1.5 (exercise) on the scale from 1.0 (very good) to 5.0 (poor). According to the evaluations, the students appreciate the concept and the structure of the lectures. Furthermore, the availability for questions and consultation of the lecturer is acknowledged. The student encourage publishing a textbook for a better wrap-up and preparation. Also, different states of knowledge of the students should be addressed more appropriately during lectures and exercises.

## 9. CONCLUSION AND OUTLOOK

In this contribution, the content, structure and realization of a lecture and the corresponding exercise for teaching nonlinear model predictive control (NMPC) to master students are introduced. The lecture covers the topic from

theoretical fundamentals of numerical optimization and NMPC to hands-on exercises in MATLAB/Simulink. The students establish the necessary knowledge for implementing their own NMPC and even test their algorithms on a real engine test bench at the end of the semester.

Besides the theoretical framework with advantages and disadvantages of the numerous possibilities of formulation, they learn important methods for the real-time implementation of NMPC algorithms. A competition between the students at the end of the semester on a real diesel engine test bench arouse their motivation such that they keep focused during the semester.

## ACKNOWLEDGEMENTS

The presented work was conducted within the research project *Optimization Based Multiscale Control for Low Temperature Combustion Engines* in the Research Unit (Forschungsgruppe) FOR 2401 from the German Research Foundation (Deutsche Forschungsgemeinschaft).

## REFERENCES

Albin, T. (2020). *Nonlinear Model Predictive Control of Combustion Engines*. Springer.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2018). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*.

Bock, H. and Plitt, K. (1984). A multiple shooting algorithm for direct solution of optimal control problems. In *IFAC World Congress*.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

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

Gros, S., Zanon, M., Quirynen, R., Bemporad, A., and Diehl, M. (2016). From linear to nonlinear mpc: bridging the gap via the real-time iteration. *International Journal of Control*, 0(0), 1–19.

Guzzella, L. and Onder, C.H. (2010). *Introduction to Modeling and Control of Internal Combustion Engine Systems*. Springer Berlin Heidelberg.

Honc, D., K., R.S., Abraham, A., Duek, F., and Pappa, N. (2016). Teaching and practicing model predictive control. *IFAC-PapersOnLine*, 49(6), 34 – 39. 11th IFAC Symposium on Advances in Control Education ACE 2016.

Isermann, R. (2014). *Engine Modeling and Control*. Springer Berlin Heidelberg.

Maciejowski, J. (2002). *Predictive Control with Constraints*. Prentice Hall, England.

Morari, M. and Lee, J.H. (1999). Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(45), 667 – 682.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, second edition.

Normey-Rico, J.E. and Camacho, E.F. (2007). *Control of Dead-time Processes*. Springer.

Qin, S. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7), 733 – 764.

Quirynen, R., Vukov, M., and Diehl, M. (2015). Multiple shooting in a microsecond. In T. Carraro, M. Geiger, S. Körkel, and R. Rannacher (eds.), *Multiple Shooting and Time Domain Decomposition Methods*, 183–201. Springer International Publishing, Cham.

Rawlings, J.B., Mayne, D.Q., and Diehl, M.M. (2017). *Model Predictive Control: Theory, Computation, and Design, 2nd Edition*. Nob Hill Publishing, LLC.

Shariati, S. and Abel, D. (2016). Model predictive control in two days: Educating a new way of thinking. *IFAC-PapersOnLine*, 49(6), 40 – 45. 11th IFAC Symposium on Advances in Control Education ACE 2016.

Tschanz, F., Zentner, S., Onder, C.H., and Guzzella, L. (2014). Cascaded control of combustion and pollutant emissions in diesel engines. *Control Engineering Practice*, 29, 176 – 186.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57.

Zanelli, A., Domahidi, A., Jerez, J., and Morari, M. (2017). Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 1–17.