Contract-based Predictive Control for Modularity in Hierarchical Systems

Baethge, T.; Kogel, M.; Di Cairano, S.; Findeisen, R.
TR2018-123 August 25, 2018

Abstract
Hierarchical control architectures pose challenges for control, as lower-level dynamics, such as from actuators, are often unknown or uncertain. If not considered correctly in the upper layers, requested and applied control signals will differ. Thus, the actual and the predicted plant behavior will not match, likely resulting in constraint violation and decreased control performance. We propose a model predictive control scheme in which the upper and lower levels-the controller and the actuator-agree on a "contract" that allows to bound the error due to neglected dynamics. The contract allows to guarantee a desired accuracy, enables modularity, and breaks complexity: Components can be exchanged, vendors do not need to provide in-depth insights into the components’ working principle, and complexity is reduced, as upper-level controllers do not need full model information of the lower level- the actuators. The approach allows to consider uncertain actuator dynamics with flexible, varying sampling times. We prove repeated feasibility and input-to-state stability and illustrate the scheme in an example for a hierarchical controller/plant cascade.

IFAC Conference on Nonlinear Model Predictive Control
Abstract: Hierarchical control architectures pose challenges for control, as lower-level dynamics, such as from actuators, are often unknown or uncertain. If not considered correctly in the upper layers, requested and applied control signals will differ. Thus, the actual and the predicted plant behavior will not match, likely resulting in constraint violation and decreased control performance. We propose a model predictive control scheme in which the upper and lower levels—the controller and the actuator—agree on a “contract” that allows to bound the error due to neglected dynamics. The contract allows to guarantee a desired accuracy, enables modularity, and breaks complexity: Components can be exchanged, vendors do not need to provide in-depth insights into the components’ working principle, and complexity is reduced, as upper-level controllers do not need full model information of the lower level—the actuators. The approach allows to consider uncertain actuator dynamics with flexible, varying sampling times. We prove repeated feasibility and input-to-state stability and illustrate the scheme in an example for a hierarchical controller/plant cascade.

Keywords: Hierarchical control; predictive control; modularization; contracts; robustness.

1. INTRODUCTION

Control problems in modern applications like Automatic Driving or the “Internet of Things” are often highly interconnected, as they involve combinations of supervisory controllers and lower-level actuators, see e.g. Campbell et al. (2010); Di Cairano and Borrelli (2016); Lucia et al. (2016). Such hierarchical structures, spanning multiple levels, often consist of controllers, sensors, and actuators from different manufacturers. Combining components from multiple vendors often has an impact on the available knowledge of and the communication between these components: Exact dynamics of subsystems or neighboring components might be unknown, as companies want to protect proprietary knowledge, or might change when replacing a component with a model from a different vendor. Designing model-based controllers without such knowledge is difficult, especially in the case of Model Predictive Control (MPC). MPC schemes rely on sufficiently correct and detailed system models for the prediction of the future system behavior, see e.g. Maciejowski (2002); Rawlings et al. (2017); Mayne (2014), to achieve good control performance.

We consider the case of unknown actuator dynamics, as shown in Fig. 1, to outline the appearing challenges and provide an MPC strategy that overcomes these, enables modularity, allows privacy between the components, and breaks complexity. The unknown or uncertain dynamics can “slow down”, delay, or modify the requested control input, leading to a real input that is different to the one that the controller commanded. As a consequence, the desired optimal behavior will not be reached and constraints might be violated, cf. Fig. 2. Neglecting this mismatch can result in increased conservatism, higher energy consumption, or even instability.

We propose an MPC scheme that takes the additional dynamics—without in-depth knowledge—into account as an additional uncertainty. The uncertainty is bounded in form of a suitable maximum error or an accuracy
The remainder of this paper is structured as follows: The general framework is introduced in Section 2. Section 3 presents an approach to bound the error that is introduced by the additional actuator dynamics in the control loop. The design of a robust MPC scheme that exploits the bound in form of a contract is presented in Section 4. A simulation example is shown in Section 5. Section 6 closes with a summary and suggestions for future work.

We use standard notation: For two sets $A$, $B$ and a matrix $M$, $A \oplus B$, $A \ominus B$, $MA$, $X$ denote the Minkowski sum, the Minkowski difference, the set multiplication, and the Cartesian product, respectively, see e.g. Blanchini and Miani (2015). A set $S$ is called robust positive invariant (RPI) under $s_{k+1} = Ss_k + c_k$, $c_k \in \mathbb{E}$, where $\mathbb{E}$ is a convex compact set with $0 \in \mathbb{E}$, if $s_B$ is a set, $a_{i,j}$ denotes the value of $a$ at time $t_i$, calculated at time $t_j$.

## 2. HIERARCHICAL CONTROL ARCHITECTURE

We consider a hierarchical control architecture in which an upper-level controller interacts with a plant and a lower-level actuator. The plant dynamics, which the upper-level controller knows and can use, cf. Fig. 1, are given by

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t) + Du(t),$$

where $x(t) \in \mathbb{R}^{n_x}$ denotes the states, $v(t) \in \mathbb{R}^{n_u}$ the inputs applied by the actuator, and $y(t) \in \mathbb{R}^{n_y}$ the plant outputs. The plant states, inputs, and outputs need to fulfill state, input, and output constraints,

$$x(t) \in X, \quad v(t) \in U, \quad y(t) \in Y,$$

respectively. We assume that $X$, $U$, and $Y$ are compact convex polytopes containing the origin in their interior.

The upper-level controller requests a command signal $u(t) \in U$ from the actuator, which results in the applied input $v(t)$. Ideally,

$$v(t) = u(t),$$

i.e., the requested and the applied input signals match. However, in reality, the actuator introduces additional dynamics, which we consider to be of the form

$$\dot{z}(t) = A_s z(t) + B_s u(t), \quad v(t) = C_s z(t).$$

Here, $z(t) \in \mathbb{R}^{n_z}$ denotes the actuator states, $u(t) \in \mathbb{R}^{n_u}$ the inputs commanded to the actuator, and $v(t) \in \mathbb{R}^{n_v}$ the actuator outputs that are applied to the plant as inputs.

We assume that the actuator satisfies:

Assumption 1 (Stable actuator dynamics): The actuator dynamics (2) are asymptotically stable, i.e., $A_s$ is Hurwitz.

Assumption 2 (Unity gain of the actuator dynamics): The actuator has a steady state unity gain: A constant $u(t)$ implies $v(t) \rightarrow u(t)$ and $\forall u': 0 = A_s z_s + B_s u', u' = C_s z_s$.

Note that Assumption 2 can be satisfied easily by a suitable choice/scaling of the inputs $u(t)$.

Remark 3 (Actuator constraints): For simplicity and as we assume a well-behaving and stable actuator, we only consider input constraints $U$ for the actuator, but no actuator state constraints.
As outlined, the exact lower-level actuator dynamics might be unknown or unavailable, e.g. for proprietary or privacy reasons. Thus, the upper-level controller has no knowledge of the actuator model (2). However, it can negotiate a “contract” with the actuator during the design phase or at specific times during the operation phase, which defines allowable input changes and input limits to achieve a desired actuator-plant error.

Given this setup, we want to solve the following problem:

**Problem 4 (Modular contract-based controller design):** Design an upper-level controller that robustly stabilizes the lower-level plant (2) and achieves constraint satisfaction despite limited knowledge of the actuator dynamics.

To tackle this problem, we suggest that the controller treats the unknown actuator dynamics as an additional uncertainty that is directly considered in the prediction. The bounding set for the uncertainty, the accuracy, together with the necessary constraints on the control input form an actuator-controller contract. To calculate these sets, we derive bounds for the error that is introduced by the additional dynamics.

Note that, for clarity of presentation, we focus on a single actuator-controller configuration. Expansion to the multiple-actuator case is easily possible.

### 3. BOUNDING THE ACTUATOR ERROR

The contract between controller and actuator is based on an upper bound for the error that the actuator dynamics (2) cause in comparison to the ideal actuator dynamics (3).

As a consequence of the made assumptions, we furthermore have:

**Proposition 5 (Basic properties of the discretized systems):** The matrices \( \hat{F} \) and \( F \) satisfy \( \hat{F} = F = e^{AT} \). If Assumption 1 holds, then \( F_a = e^{A_T} \) is Schur stable, there exists an RPI set \( \tilde{Z} \) for the actuator dynamics, such that

\[
\tilde{Z} \subseteq F_a Z \oplus G_z U,
\]

and the steady state \( z_\infty \) of the actuator for a given steady state input \( u_\infty \) satisfies

\[
z_\infty = (I - F_a)^{-1} G_z u_\infty.
\]

If Assumptions 1 and 2 hold, then

\[
\Delta G = G_x - \tilde{G} = -F_c(I - F_a)^{-1} G_z,
\]

as, from Assumption 2, any constant \( u(t) = \hat{u} \) gives \( v(t) = u(t) = \hat{u} \) and \( Bv(t) = Bu(t) = B\hat{u} \) (compare linear dynamics of normal and augmented system). Then, \( \hat{G}_u = G_x \hat{u} + F_c(I - F_a)^{-1} G_z \hat{u} \).

**Remark 6 (Flexible sampling time):** In principle, there is no need to use a constant input signal for the full sampling time \( T \) in the actuator. One could use a faster sampling time, e.g. \( h = \frac{T}{2} \), \( H \in \mathbb{N} \), allowing a faster actuator controller of the form

\[
u^c(t) = \begin{cases}
K_1 \left( z(t_k) \right), & t \in [kT, kT + h) \\
K_2 \left( z(t_k + h) \right), & t \in [kT + h, kT + 2h) \\
& \vdots
\end{cases}
\]

Here, \( u^c \) is the command sent to the plant in this case, which can be reformulated in a similar form as (7).

The controller does not know the actuator dynamics (2). Thus, the predicted state at the next time instant, \( \hat{x}(t_{k+1}) = \hat{F}x(t_k) + \hat{G}u(t_k) \), \( \hat{y}(t_k) = C\hat{x}(t_k) + D u(t_k) \).

Here, \( \hat{y} \) denotes the variables of the combination of plant and ideal actuator. This model, expanded by an error term, forms the basis for the upper-level controller.

**Model of the plant with ideal actuator:** The ideal system, consisting of the plant dynamics (2) and the ideal actuator dynamics (3), is given by

\[
\begin{align*}
\hat{x}(t_{k+1}) &= \hat{F}x(t_k) + \hat{G}u(t_k) \\
\hat{y}(t_k) &= C\hat{x}(t_k) + D u(t_k).
\end{align*}
\]

**Model of the plant and the actuator:** The real system, combining the plant dynamics (2) and the actuator dynamics (2), is given by

\[
\tilde{x}(t_{k+1}) = F \tilde{x}(t_k) + G \tilde{u}(t_k), \quad \tilde{y}(t_k) = C \tilde{x}(t_k) + D \tilde{u}(t_k).
\]

The corresponding discrete-time system, using the sampling time \( T \), is given by

\[
\begin{align}
\xi(t) &= \begin{pmatrix} A & B C_a \\ 0 & A_a \end{pmatrix} \xi(t) + \begin{pmatrix} 0 \\ B_a \end{pmatrix} u(t), \quad \xi = \begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}.
\end{align}
\]

In other words, the set-based dynamics

\[
\tilde{x}(t_{k+1}) = \hat{F} \tilde{x}(t_k) + \hat{G} \tilde{u}(t_k), \quad \tilde{w}(t_k) \in \tilde{W},
\]

outer-bound the behavior of the real system.

First, we establish a new way to find such a set \( \tilde{W} \) and that the error approaches zero as the input goes to zero:

**Proposition 7 (Bounding the actuator error):** Under Assumption 1 and if \( z(t_0) \in Z \), where \( Z \) satisfies (8), a \( \tilde{W} \) such that (12) holds is given by

\[
\tilde{W} = F_c Z \oplus \Delta G U.
\]

Furthermore, if \( u(t_k) \rightarrow 0 \), then \( \tilde{w}(t_k) \rightarrow 0 \).

**Proof.** Exploiting equation (12) and Proposition 5, it is clear that

\[
w(t_k) = x(t_{k+1}) - \hat{x}(t_{k+1}) = F_c z(t_k) + \Delta G u(t_k).
\]

This yields equation (14) using the bounds on \( z(t_k) \) and \( u(t_k) \). Furthermore, note that \( F_a \) is Schur stable. Thus, \( u(t_k) \rightarrow 0 \) in (7) implies \( z(t_k) \rightarrow 0 \), thus, \( F_c z(t_k) + \Delta G u(t_k) \rightarrow 0 \).
Thus, we obtain from (12) that
\[ \Delta z(t_i) = z(t_i) - (I - F_a)^{-1}G_z u(t_{i-1}). \] (16)

This enables the establishment of the following:

**Proposition 8 (Error in formulation):** Let Assumptions 1 and 2 and \( \Delta z(t_0) \in \Delta Z \) hold, where \( \Delta Z \) satisfies
\[ \Delta Z \subseteq F_a \Delta Z \oplus -F_a(I - F_a)^{-1}G_z \Delta U. \] (17)

If \( \Delta u(t_i) \in \Delta U, i = 0, \ldots, k, \) then a \( W \) leading to satisfaction of (12) is given by
\[ W = F_c \Delta Z \oplus \Delta G \Delta U. \] (18)

**Proof.** Proposition 5 and (16) yield
\[ \Delta G u(t_{i-1}) = -F_c(I - F_a)^{-1}G_z u(t_{i-1}) = F_c(\Delta z(t_i) - z(t_i)). \] (19)

Thus, we obtain from (12) that
\[ w(t_i) = F_c z(t_i) + \Delta G(u(t_{i-1}) + \Delta u(t_i)) = F_c \Delta z(t_i) + \Delta G \Delta u(t_i), \] (20)

which establishes (18) using the bounds on \( \Delta z(t_i) \) and \( \Delta u(t_i). \)

Using (7), (22), and (16), we obtain
\[ \Delta z(t_{i+1}) = F_a z(t_i) + G_z w(t_i) - (I - F_a)^{-1}G_z u(t_i) = F_a z(t_i) - F_a(I - F_a)^{-1}G_z u(t_i) \]
\[ = F_a \Delta z(t_i) - F_a(I - F_a)^{-1}G_z \Delta u(t_i), \] (21)

leading to (17). □

Thus, to achieve the desired actuator accuracy error, we need to enforce input and input rate constraints in the upper-level controller. We do so using an MPC scheme in input-change formulation, as often done in MPC, see e.g., Maciejowski (2002): The input \( u(t_k) \) can be written as a sum of input changes and the input of the actuator at the time shortly before the initial time \( t = 0, \) i.e.,
\[ u(t_k) = u(t_{i-1}) + \sum_{i=0}^{k} \Delta u(t_i). \] (22)

Here, \( u(t_{i-1}) = \lim_{t \to 0^-}z(t) \) is the actuator state just before \( t = 0. \)

**Remark 9 (Uncertain actuator dynamics):** We assume that the dynamics to establish the contract in the actuator are exactly known, which in practice, however, is often not the case. A straightforward extension to cover “multiplicative uncertainties” is to assume that the matrices \( F_a, F_c, G_z, \) and \( G_z \) are given by
\[ F_a = \sum_{i=1}^{V} \lambda_i F_a^i, \quad F_c = \sum_{i=1}^{V} \lambda_i F_c^i, \] (23a)
\[ G_z = \sum_{i=1}^{V} \lambda_i G_z^i, \quad G_z = \sum_{i=1}^{V} \lambda_i G_z^i. \] (23b)

where \( \lambda_i \geq 0 \) and \( \sum_{i=1}^{V} \lambda_i = 1. \) \( F_a^i, F_c^i, G_z^i, \) and \( G_z^i \) form the corners defining the uncertainty.

It is easy to show that \( W \subseteq W^i, i = 1, \ldots, V, \) where \( W^i \) are the sets obtained from Proposition 7 (or Proposition 8) for the matrices \( F_a^i, F_c^i, G_z^i, \) and \( G_z^i. \) This allows to determine a bound on \( w(t_k) \) such that equation (12) holds for such multiplicative uncertainties (23).

As an immediate consequence, a bounding set \( W \) can be obtained from (14). \( W \) then needs to be the desired accuracy bound \( W_{\text{request}} \) that the upper-level controller wants to be guaranteed. During the contract negotiation, the actuator obtains a request \( W_{\text{request}} \) from the controller. This can be rejected or confirmed by suitable sets \( U, \Delta U, \) and \( W. \) The controller can then certify whether it accepts the sets \( U \) and \( \Delta U \) or asks for a different \( W_{\text{request}}. \)

### 4. CONTROLLER DESIGN

We use a tube-based controller in the upper level to achieve constraint satisfaction, exploiting the controller-actuator accuracy-input contract. In tube-based MPC, the set of perturbed trajectories is bounded by a “tube” around the nominal trajectory, cf. Mayne et al. (2005, 2006).

To consider the input rate constraints, we embed the input rate in virtual dynamics,
\[ \tilde{x} = \begin{pmatrix} \tilde{x}_k \\ u_{k-1} \end{pmatrix}, \quad \tilde{y} = \begin{pmatrix} y_k \\ u_k - u_{k-1} \end{pmatrix} = \begin{pmatrix} C & 0 \\ 0 & -I \end{pmatrix} \tilde{x} + \begin{pmatrix} D \\ I \end{pmatrix} u, \]

for which we enforce the constraints \( \tilde{Y} = \tilde{y} \times \Delta U. \) Here, \( \tilde{Y} \) are general “output” constraints.

The resulting tube-based MPC scheme solves
\[ \min_{x_k, u_k} J(x_k, u_k) \] (24)

in a receding-horizon fashion, where
\[ x_k = \{ x_{kj}, \ldots, x_{kj+N[k]} \}, \] (25a)
\[ u_k = \{ u_{kj}, \ldots, u_{kj+N[k]-1} \}, \] (25b)
and \( N \in \mathbb{N}, N > 1, \) is the length of the prediction horizon, and where the cost function is given by
\[ J(x_k, u_k) = x_{k+N[k]}^T P x_{k+N[k]} + \sum_{i=k}^{k+N-1} x_i^T Q x_i + u_i^T R u_i, \] (25c)

with positive definite weighting matrices \( Q, P, R. \) The dynamics and constraints are given by
\[ x_{i+1} = F x_i + G u_i, \quad i = k, \ldots, k + N - 1, \] (25d)
\[ x_{k+1} = \{ x(t_k) \} \oplus \mathbb{D}, \] (25e)
\[ x_{ij} \in \tilde{X}, \quad i = k, \ldots, k + N - 1, \] (25f)
\[ u_{ij} \in \tilde{U}, \quad i = k, \ldots, k + N - 1, \] (25g)
\[ C x_{ij} + D u_{ij} \in \tilde{Y}, \quad i = k, \ldots, k + N - 1, \] (25h)
\[ x_{k+N[k]} \in \tilde{T}. \] (25i)
The sets $\mathcal{X}$ and $\mathcal{Y}$ contain a neighborhood of the origin and satisfy
\begin{align*}
\mathcal{X} &= \mathcal{D} \cap \mathcal{F} \cap \mathcal{T}, \\
\mathcal{Y} &= \mathcal{D} \cap \mathcal{F} \cap \mathcal{T}.
\end{align*}

The constraint back-off set $\mathcal{D}$ satisfies
\begin{equation}
\mathcal{D} \supseteq (\mathcal{F} + \mathcal{G}K)\mathcal{D} + \mathcal{W}. 
\end{equation}

The sets $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{Y}$ are tightened constraint sets for states, inputs, and outputs, and $\mathcal{T}$ is the terminal set. Suitable choices for these sets that guarantee repeated feasibility and stability will be given below.

To achieve robust stability and constraint satisfaction, we require:

**Assumption 10 (Conditions on sets $\mathcal{D}$, $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$, and $\mathcal{T}$):**
The constraint back-off set $\mathcal{D}$ satisfies
\begin{equation}
\mathcal{D} \supseteq (\mathcal{F} + \mathcal{G}K)\mathcal{D} + \mathcal{W}. 
\end{equation}

The sets $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{Y}$ contain a neighborhood of the origin and satisfy
\begin{align*}
\mathcal{X} &= \mathcal{D} \cap \mathcal{F} \cap \mathcal{T}, \\
\mathcal{U} &= \mathcal{D} \cap \mathcal{F} \cap \mathcal{T}, \\
\mathcal{Y} &= \mathcal{D} \cap \mathcal{F} \cap \mathcal{T}.
\end{align*}

The terminal penalty $\mathcal{P}$ in the cost function (25c) satisfies
\begin{equation}
P = (\mathcal{F} + \mathcal{G}K)^T P (\mathcal{F} + \mathcal{G}K) + \mathcal{Q} + \mathcal{K}^T \mathcal{K}. 
\end{equation}

Using this and Theorem 12, we can establish stability:

**Theorem 14 (Asymptotic stability):** Let Assumptions 1, 10, 11, and 13 and $z(t_0) \in \mathbb{Z}$ hold, where $\mathbb{Z}$ satisfies (8). If (25) is feasible at $t_0$, then the closed-loop system (7), (26) is asymptotically stable.

**Proof. (Sketch).** Theorem 12 guarantees recursive feasibility and constraint satisfaction. Moreover, with the assumptions made for $x_{00} = x(t_0)$, the LQR controller is locally admissible. Together with the fact that all constraint sets in the optimization problem (25) are compact, $\exists c_1 > 0$ s.t. $\|x_{00}\| \leq c_1 \|x(t_0)\|$ and $\|u_{00}\| \leq c_1 \|x(t_0)\|$.

This allows, similar to standard tube-based MPC, to show that the nominal state $x^*_{ik}$ is asymptotically stable.

Considering the dynamics of the plant and actuator, we can furthermore derive that
\begin{equation}
\Delta z(t_k) = z(t_k) - z^*_{ik} 
\end{equation}
and
\begin{equation}
\Delta z(t_{k+1}) = F \Delta z(t_k) + G u^*_{ik} + \left( \Delta x^* \right) \n
\end{equation}
where $\Delta x^* = F x^*_{ik} + G u^*_{ik} - x^*_{ik-1}$.

5. SIMULATION EXAMPLE

To illustrate the approach, we consider a double integrator as the plant, given by
\begin{equation}
\dot{x}(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} v(t), \quad y(t) = (1 0) x(t).
\end{equation}

The actuator is described by the first-order system
\begin{equation}
\dot{z}(t) = -20 z(t) + 20 u(t), \quad v(t) = z(t).
\end{equation}

Corresponding discrete-time formulations (5) and (7) are obtained using a sampling time $T = 0.3$, and states, control inputs, and control input changes are box-constrained:

\begin{equation}
X = \begin{pmatrix} -10 & 10 \\ -10 & 10 \end{pmatrix}, \quad U = [-2, 2], \quad \Delta U = [-0.4, 0.4].
\end{equation}

The weighting matrices in the cost function are chosen as $Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $R = 1000$. They are used to calculate the tube controller gain $K$ as the corresponding LQR gain and terminal penalty $P$ to satisfy Assumption 13.

We consider that the MPC controller should bring the plant from the initial state $x(t_0) = (-7.7, 5.7)^T$ to the origin. The initial state for the actuator is $z(t_0) = 0$. 

\begin{equation}
\hat{x}(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} v(t), \quad y(t) = (1 0) x(t).
\end{equation}
The simulation results are obtained from solving optimization problem (25) in a receding-horizon manner in MATLAB. The problem is formulated with YALMIP (Löfberg, 2004). MPT3 (Herceg et al., 2013) and PnPMPC (Riverso et al., 2013) are used to determine the required sets.

Fig. 3 shows the trajectories of plant and actuator states, as well as control inputs. All constraints are satisfied and the plant is successfully stabilized at the origin. The feasible region for the plant states is shown in Fig. 4.

With this, we offer a method to improve controller performance in settings where not all information is available, e.g. due to privacy, legal, or modularity reasons. This is often the case in industrial applications, where different manufacturers might consider internal actuator dynamics as proprietary knowledge. Furthermore, the approach allows constructing modular controller-actuator-plant cascades, in which actuators can be exchanged—e.g. for other models or vendors with different system dynamics—as long as they satisfy the same accuracy contract.

Future work and potential extensions will consider nonlinear dynamics for plant and actuator as well as various descriptions of uncertainty.

6. SUMMARY AND OUTLOOK

We considered the problem of hierarchical control, where the upper-level controller does not have a detailed model of the lower level. As a special case, we presented a Model Predictive Control scheme that stabilizes a system consisting of known plant dynamics and an actuator, whose dynamics are unknown to the MPC controller. To achieve stability and constraint satisfaction, the controller and the actuator agree on “contract” during the design phase. This contract consists of a bound on the error resulting from the unknown intermediate actuator. To achieve the desired accuracies, the actuator provides bounds on the allowable inputs and input changes. First, we derived a robust positive invariant set for the error bound, in a discrete-time setting. With this bound, we setup a tube-based MPC scheme to guarantee constraint satisfaction and stability. The simulation example provided further insights into the control scheme.

REFERENCES

Blanchini, F. and Miani, S. (2015). Set-Theoretic Methods in Control. Systems & Control: Foundations & Applications. Birkhäuser, Basel, 2nd edition.

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

Bäthge, T., Lucia, S., and Findelen, R. (2016). Exploiting Models of Different Granularity in Robust Predictive Control. In Proc. IEEE Conf. Dec. Contr., 2763–2768.

Campbell, M., Egerstedt, M., How, J.P., and Murray, R.M. (2010). Autonomous driving in urban environments: approaches, lessons and challenges. Philosophical Trans. Royal Society A. 368(1928), 4649–4672.

Di Cairano, S. and Borrelli, F. (2016). Reference tracking with guaranteed error bound for constrained linear systems. IEEE Trans. Autom. Contr., 61(8), 2245–2250.

Di Cairano, S., Bäthge, T., and Findelen, R. (2018). Modular MPC Design for Constrained Control of Plant-Actuator Cascades. Subm. to IEEE Conf. Dec. Contr.

Herceg, M., Kvasnica, M., Jones, C.N., and Morari, M. (2013). Multi-Parametric Toolbox 3.0. In Proc. Euro. Contr. Conf., 502–510.

Kögel, M. and Findelen, R. (2015). Robust output feedback model predictive control using reduced order models. In Proc. Int. Symp. Adv. Contr. Chem. Process., 1009–1015.

Kögel, M. and Findelen, R. (2018). Improved Robust Decentralized MPC. In Proc. Euro. Contr. Conf., 312–318.

Lucia, S., Kögel, M., and Findelen, R. (2015). Contract-based Predictive Control of Distributed Systems with Plug and Play Capabilities. In Proc. IFAC Conf. Nonlinear Model Predictive Contr., 205–111.

Lucia, S., Kögel, M., Zometa, P., Quevedo, D.E., and Findelen, R. (2016). Predictive control, embedded cyberphysical systems and systems of systems – A perspective. Annu. Rev. Contr., 41, 193–207.

Löfberg, J. (2004). YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In Proc. IEEE Int. Symp. Comp. Aided Control Syst. Design (CACSD), 284–289.

Maciejowski, J.M. (2002). Predictive Control with Constraints. Prentice Hall, Harlow.
Mayne, D.Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, 50(12), 2967–2986.
Mayne, D.Q., Raković, S.V., Findeisen, R., and Allgöwer, F. (2006). Robust output feedback model predictive control of constrained linear systems. *Automatica*, 42(7), 1217–1222.
Mayne, D.Q., Seron, M.M., and Raković, S.V. (2005). Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41, 219–224.
Ogata, K. (1995). *Discrete-time Control Systems*. Prentice Hall, Englewood Cliffs, NJ, 2nd edition.
Rawlings, J.B., Mayne, D.Q., and Diehl, M.M. (2017). *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, Madison, WI, 2nd edition.
Rivero, S., Battocchio, A., and Ferrari-Trecate, G. (2013). PnPMPC toolbox. URL http://sisdin.unipv.it/pnmpc/pnmpc.php.