Efficient Embedded Model Predictive Vibration Control via Convex Lifting

Martin Gulan, Gergely Takács, *Member, IEEE*, Ngoc A. Nguyen, Sorin Olaru, *Senior Member, IEEE*, Pedro Rodríguez-Ayerbe, *Member, IEEE*, and Boris Rohaľ-Ilkiv, *Member, IEEE*

***Abstract*—This paper presents an efficient real-time implementation of embedded model predictive control, adopted in the context of active vibration control with the objective of minimizing the tip deflection of lightly damped cantilever beams. In particular, we focus on memory and time-efficient explicit solutions to the associated constrained optimal control problem that are easily implementable on low-end embedded hardware. To this end, we exploit the concept of convex lifting and show how it can be used to devise low-complexity, regionless piecewise affine controllers without any loss of optimality and performance. Efficiency of this constructive procedure is quantified via an extensive complexity analysis, evidenced by a successful practical deployment and optimal vibration control performance using a family of 32-bit ARM Cortex-M based microcontroller platforms.**

***Index Terms*—Model predictive control, explicit solutions, embedded systems, vibration control, convex lifting.**

I. Introduction

MODEL predictive control (MPC) has brought a tremendous improvement to the quality of many industrial applications [1]. Since its early conception, MPC was adopted in petrochemical plants for its inherent ability to handle process constraints and its increased control performance. The relatively slow processes in the chemical industry as well as the lack of need to miniaturize and aggressively cost-optimize computing hardware were initially concealing the main drawback of predictive control methodology—its demand for computing resources. The utilization of advanced optimal control schemes was therefore at first limited to systems with slow dynamics, powerful computing implementations, or both.

Since then, the evolution of control theory and applied automation has not stopped; besides the constantly dropping price of hardware and increased performance, new computationally efficient MPC methods have been devised. Thanks to the combination of these factors, new application areas are emerging. Applications involving fast dynamics, like mechanical systems with active vibration control (AVC)—the main interest of this article—are now within the realm of practical implementation possibilities.

There have been a plethora of academic works using various adaptations of the predictive control algorithm for vibration attenuation. However, with a few exceptions [2]–[4], these works are restricted to computing hardware that is feasible only in a laboratory setting due to its size and weight [5]. Whether one may use MPC for fast systems like active vibration control in consumer-level products, depends on an often overlooked but essential practical factor: hardware price.

Embedded single-chip systems, also known as microcontroller units (MCU), can offer only a small fraction of execution performance or memory compared to their laboratory or industrial counterparts, but they may be mass-produced and built-into miniaturized devices for very little money. Microcontroller families and architectures that were once part of bulky personal computers that were being slowly forgotten are now making a comeback in miniaturized versions, partly due to their low price [6].

This is where the efficiency-boosting achievements of control theory in the field of MPC come to the foreground: these developments allow one to implement better control methods with less resources. Because of the improvements in algorithm efficiency, model predictive control can now be implemented on embedded hardware such as MCUs [7], [8], programmable logic controllers (PLC) [9]–[11], or field programmable gate arrays (FPGA) [4], [12], [13], etc. Efficiency improvements in nominal or deterministic MPC can be divided into two main categories [14]. To the first belong online MPC algorithms that attempt to minimize the real-time computational requirements by a context-oriented reformulation of the optimization problem, or by the use of advanced, often hardware-targeted optimization solvers [15]–[17]; sometimes referred to as implicit MPC. The other group consists of a family of methods known as explicit MPC (EMPC), which essentially turn the real-time optimization problem into a simple table-lookup procedure by precomputing its optimal solution [18], [19].

Computationally efficient developments in implicit MPC attempt to minimize the online execution requirements, so that simpler control hardware is sufficient. These methods still require a considerable computing power, however, are usually not memory-intensive. The explicit approach is quite different; it trades execution speed for memory footprint. EMPC may be executed with very little computing effort, but it is known for its memory complexity that grows exponentially with the prediction horizon [20], which may quickly make the hardware deployment or the online evaluation intractable.

Manuscript received February 15, 2017; revised July 7, 2017; accepted October 8, 2017. Manuscript received in final form October 13, 2017. This work was supported by the Slovak Research and Development Agency under the contracts No. APVV-14-0399 and APVV SK-FR-2015-0015, and by the Scientific Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under the contract No. 1/0144/15. Recommended by . *(Corresponding author: Martin Gulan.)*

M. Gulan, G. Takács, and Boris Rohaľ-Ilkiv are with the Institute of Automation, Measurement and Applied Informatics, Faculty of Mechanical Engineering, Slovak University of Technology, 81231 Bratislava, Slovakia (e-mail: martin.gulan@stuba.sk; gergely.takacs@stuba.sk; boris.rohal-ilkiv@stuba.sk).

N. A. Nguyen, S. Olaru and P. Rodríguez-Ayerbe are with the Laboratoryof Signals and Systems (L2S, UMR CNRS 8506), CentraleSupélec-CNRS-Université Paris Sud, Université Paris-Saclay, 91190 Gif-sur-Yvette, France. (e-mail: ngocanh.nguyen.rs@gmail.com; sorin.olaru@centralesupelec.fr; pedro.rodriguez@centralesupelec.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2017.2764019

In this paper, we resort to the explicit approach to solving the MPC problem, since it inherently comes with several convenient features from the real-time applications' perspective. In particular, unlike its implicit counterpart, the structure and properties of the explicit solution allow for a straightforward implementation in a division-free fashion, and moreover provide a tool for exact worst-case certification of the implementation and closed-loop analysis [21]. We look at a particular possibility to dramatically decrease both the memory and runtime complexity of the EMPC code, so that it can be implemented in low-cost embedded microcontrollers. Our work is focused on a particular class of systems with fast dynamics—the aforementioned problem of active vibration control. Nowadays, experimental AVC systems start to leave laboratories and appear in consumer products, like active car suspensions [22] or even miniaturized medical devices to reduce hand tremor in the sufferers of Parkinson's disease [23]. Vibration control systems are this way getting smaller and thus require tiny and cheap microcontrollers that have limited power consumption. There is indeed no good reason why these devices and products should not benefit from advanced control algorithms, but are current embedded devices up to the task?

Of course, embedded model predictive control has been applied to AVC systems before. Possibly the first work of its kind used implicit approaches and machine level code optimization on digital signal processors (DSP) [3]. While therein achieved sampling speeds measured in tens of kilohertz were remarkable, high-end DSP hardware still remains prohibitively expensive for low-cost mass-produced applications. Many have suggested explicit MPC as the ideal formulation for cheap embedded hardware [9], [24]. Former trials with EMPC for vibration attenuation evaluated memory needs and algorithm timing, albeit on powerful hardware that is suitable only for laboratory tests [5], [25]. Explicit MPC without efficiency modifications was recently applied to AVC via embedded hardware, showing the limits of hardware and software on high-end microcontrollers [26]. Thanks to the underlying formulation of EMPC, execution speed does not create any prohibitive issues in embedded MCU, but the available volatile and non-volatile memory of a microcontroller does affect its class and therefore its cost. In an ideal case, microcontrollers closer to what is today considered as average should be capable of running model predictive vibration control. Nevertheless, contemporary mid-range MCUs are simply not sufficient for vibration attenuation based on explicit MPC [26].

To this end we introduce an efficient methodology to transform the nominal, memory-intensive EMPC into an equivalent formulation that enables to meet the strict requirements imposed by technical specifications of low-end embedded control hardware, with no implications in loss of optimality or closed-loop stability. To this end, we exploit the concept of convex lifting, recently adopted in [27]–[29] in the context of control design, and show that it can be used to devise regionless yet fully optimal EMPC controllers implementable in embedded AVC applications with fast sampling speeds. Compared with the former study [26], we aim at EMPC solutions with guaranteed stability—that render much smaller domains of attraction and hence need longer prediction horizons [30]. The resulting algorithm will be implemented in a range of 32-bit embedded microcontrollers, to provide an overview of its memory footprint and execution timing. This embedded AVC system will be used to minimize the tip deflections of an aluminum cantilever beam, by supplying the input decisions of the proposed control algorithm to the piezoceramic actuators in the form of a driving voltage, while gaining its feedback from the position measurements. The purpose of the active cantilever beam featured here is via release tests to emulate the dynamic behavior of a class of flexible mechanical structures under transient external disturbances [31].

Note that there are several well-known control algorithms routinely used in AVC, the standard one being the positive position feedback (PPF) controller [32], which is, however, not in the scope of this study. The use of PPF along with nominal EMPC for AVC can be found in our previous work [26].

The rest of the paper is structured as follows. After introducing the system model, Section II recalls the concept of explicit model predictive control and presents two convex lifting based methods for efficient embedded hardware implementation of MPC in active vibration control, followed by a detailed memory and runtime complexity analysis. Section III describes the laboratory setup and the experimental deployment of the auto-generated code on a class of 32-bit ARM Cortex-M microcontroller units. Finally, vibration damping performance, memory and timing properties of the proposed algorithms are discussed in Section IV.

Notation

We denote by $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times m}$, $\mathbb{N}$ and by $\mathbb{N}_+$ the sets of real numbers, n-dimensional real vectors, $n \times m$ dimensional real matrices, non-negative and positive integers, respectively. For a vector-valued function $f:\mathbb{R}^n\to\mathbb{R}^m$, $\mathrm{dom}(f)$ denotes its domain. Given an arbitrary set $\mathcal{S}$, $\mathrm{conv}(\mathcal{S})$ and $\dim(\mathcal{S})$ denote its convex hull and the dimension of its affine hull. Moreover, if $\mathcal{S}$ is full-dimensional, $\mathrm{int}(\mathcal{S})$ denotes its interior. Given a set $\mathcal{S}\subseteq\mathbb{R}^n$ and a subspace $\mathbb{S}$ of $\mathbb{R}^n$, $\mathrm{Proj}_{\mathbb{S}}\,\mathcal{S}$ denotes the orthogonal projection of $\mathcal{S}$ on the space $\mathbb{S}$. Given two sets $\mathcal{S}_1, \mathcal{S}_2$, we define the following set: $\mathcal{S}_1\backslash\mathcal{S}_2 := \{\mathbf{x} \mid \mathbf{x}\in\mathcal{S}_1, \mathbf{x}\notin\mathcal{S}_2\}$. In addition, a finite index set of $N\in\mathbb{N}_+$ elements will be denoted as $\mathcal{I}_N := \{1,\dots,N\}$, and its cardinality by $|\mathcal{I}_N|$.

II. Controller design

A. Modeling

The mechanical behavior of a given physical object mainly depends on its modal properties and energy dissipation, known as damping. There are infinitely many resonance frequencies and modes for every real-life object, however, it is sufficient to include the principal ones in a mathematical representation to achieve a good match with the measured behavior [22], [33]. The dynamic response of a flexible cantilever beam is clearly dominated by its first resonant frequency and its corresponding mode of vibration [5], [34]. Thus, in order to enable the real-time tractability of model predictive control on an embedded system, we chose to represent a beam driven by piezoceramics by a simplified nominal dynamic model—an equivalent single degree of freedom (SDOF) linearly driven mass-spring-damper unit.

By assuming a viscous damping model, the system equivalent in its response can be described by a second order linear differential equation as $\ddot{q}(t)+2\zeta\omega\dot{q}(t)+\omega^2 q(t) = cu(t)$, where $q(t)$ [m] is the position of the free end of the beam, ω [rad s^{-1}] is the first natural resonance frequency and ζ [-] is the unitless damping ratio [33]. The change of the voltage $u(t)$ to driving force is linear in the piezoceramic actuators, and the constant c [N V^{-1} kg^{-1}] represents this conversion ratio. Choosing position and velocity for the state vector $\mathbf{x}(t)$, we may express the continuous-time state-space representation of the beam as

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\zeta\omega \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ c \end{bmatrix} u(t), \tag{1}$$

with the state-transition and input matrix, $\mathbf{A} \in \mathbb{R}^{2\times 2}$ and $\mathbf{B} \in \mathbb{R}^{2\times 1}$, given as above, while assuming only the measurement of position gives the output matrix as $\mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}$. The driven beam was identified using a pseudo-random binary excitation signal, supplied to the actuating elements for a period of 100 s. Using a grey-box prediction error method system identification procedure, the unknown parameters for the laboratory system that is further described in Section III were determined as $\omega = 50.89$ rad s^{-1} (8.10 Hz), $\zeta = 0.005$ and $c = 5.91$ N kV^{-1} kg^{-1}.

We remark that the assumption that the beam dynamics may be represented by an SDOF model is, apart from providing satisfactory results, an essential premise to this work, as it is unlikely that complex prediction models are able to preserve real-time implementation feasibility on simple microcontrollers [5]. Although by using embedded computing devices with generous memory capacities one may be able to utilize EMPC for the vibration control of up to 2–3 resonant modes or multi-input multi-output systems, it is unreasonable to expect a complex electromechanical model derived e.g. by a finite element analysis to be viable on current hardware.

Before proceeding, let us emphasize that the following theoretical developments are valid for the class of linear systems, however, in the sequel we will specifically focus on the active vibration control problem which is central to this study.

B. Explicit model predictive control

Let us assume control of linear discrete-time systems in the state-space form, given as

$$\mathbf{x}(t+1) = \mathbf{A}_\mathrm{d}\mathbf{x}(t) + \mathbf{B}_\mathrm{d}\mathbf{u}(t), \tag{2}$$

where t denotes multiplies of the sampling period T_s, and the pair $(\mathbf{A}_\mathrm{d}, \mathbf{B}_\mathrm{d})$ is stabilizable. In the presence of input (and/or state) constraints, we may formulate the following constrained finite-time optimal control problem:

$$\min_{\mathbf{U}} \ \sum_{k=0}^{N-1} \left\{ \mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k \right\} + \mathbf{x}_N^T \mathbf{P} \mathbf{x}_N \tag{3a}$$

$$\text{s.t. } \mathbf{x}_{k+1} = \mathbf{A}_\mathrm{d}\mathbf{x}_k + \mathbf{B}_\mathrm{d}\mathbf{u}_k, \ \ k = 0, \ldots, N-1, \tag{3b}$$

$$\mathbf{u}_k \in \mathcal{U}, \, \mathbf{x}_k \in \mathcal{X}, \ \ k = 0, \ldots, N-1, \tag{3c}$$

$$\mathbf{x}_N \in \mathcal{C}_\infty, \tag{3d}$$

where $\mathbf{x}_k$ and $\mathbf{u}_k$ denote, respectively, state and control input predictions over a finite horizon $N \in \mathbb{N}$ at time instant $t+k$, initialized by the current state, i.e. $\mathbf{x}_0 = \mathbf{x}(t)$, and subject to polytopic constraints given by $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $\mathcal{U} \subseteq \mathbb{R}^{n_u}$. Within the quadratic objective (3a), the stage costs are weighted with $\mathbf{Q} \succeq 0$, $\mathbf{R} \succ 0$, while the terminal penalty uses $\mathbf{P} \succeq 0$ usually determined as a solution of the discrete-time algebraic Riccati equation (DARE) for the unconstrained problem (3a). Reformulating (3) into a quadratic program (QP) and solving it for a feasible initial state $\mathbf{x}_0$ yields a sequence of optimal control moves $\mathbf{U}^\star = \left[\mathbf{u}_0^{\star T}, \ldots, \mathbf{u}_{N-1}^{\star T}\right]^T \in \mathbb{R}^{Nn_u}$. The receding horizon MPC feedback thus becomes $\mathbf{u}_0^\star$, which is actually implemented to the controlled system, and the procedure is repeated at the next sampling instant for a new value of the state. In addition, persistent feasibility and stability may be guaranteed by employing a maximal control invariant set $\mathcal{C}_\infty$ [35] as terminal constraint set in (3d).

Note also that for ease of presentation and without loss of generality we will in further developments assume single-input systems, such as the vibration system (1).

In view of practical implementation aspects discussed in the introductory section, let us focus on the explicit representation of the optimizer to the MPC problem (3), $u_0^\star$, rather than its considerably more expensive computation in the implicit fashion outlined above. As shown e.g. in [18], this can be achieved by recasting and solving (3) as a parametric QP (pQP) using the technique of parametric programming, which allows us to precompute the MPC control law $\mathbf{U}^\star(\mathbf{x})$ for all feasible values of parameter $\mathbf{x}$, explicitly, as a continuous and piecewise affine (PWA) function. In the receding horizon implementation, the closed-loop explicit MPC feedback has the following form:

$$u_0^\star(\mathbf{x}) = \kappa(\mathbf{x}) := \begin{cases} \mathbf{f}_1^T \mathbf{x} + g_1 & \text{if } \mathbf{x} \in \mathcal{R}_1, \\ \quad \vdots & \\ \mathbf{f}_R^T \mathbf{x} + g_R & \text{if } \mathbf{x} \in \mathcal{R}_R. \end{cases} \tag{4}$$

A polyhedron denotes the intersection of a finite set of closed halfspaces. The collection of the polyhedral regions $\mathcal{R}_i$ in (4), $\{\mathcal{R}_i\}_{i=1}^R$, is referred to as a partition of the set of feasible parameters; a formal definition is given in next subsection. The online implementation effort thus reduces to a simple function evaluation, as per (4), where the most time is spent on the point location, i.e. determining which polyhedral region $\mathcal{R}_i = \{\mathbf{x} \in \mathbb{R}^{n_x} \mid \mathbf{H}_i\mathbf{x} \leq \mathbf{h}_i\}$, $\mathbf{H}_i \in \mathbb{R}^{n_\mathrm{h}^i \times n_x}$, $\mathbf{h}_i \in \mathbb{R}^{n_\mathrm{h}^i}$, the current state resides in, by checking its defining inequalities, i.e. halfspaces. A straightforward way for searching the state-space partition is the direct sequential region traversal (see Algorithm 1) with runtime complexity linear in the number of regions. The other crucial EMPC implementation factor is the amount of memory needed to store the regions $\mathcal{R}_i$ and the optimal PWA feedback $\kappa(\mathbf{x})$. Both of the aforementioned complexity indicators are of a great practical importance, namely in case of deployment on

Algorithm 1 Standard online EMPC implementation via direct sequential search

1: At each sampling instant t, measure or estimate the current state $\mathbf{x}_0 = \mathbf{x}(t)$. If $\mathbf{x}_0 \in \mathcal{R}$ (see Def. 1), proceed to step 2.
2: Among $\{\mathcal{R}_i\}_{i=1}^R$, search for the $i^\star$-th polyhedral region $\mathcal{R}_{i^\star}$ that contains $\mathbf{x}_0$, i.e. $\mathbf{H}_{i^\star}\mathbf{x}_0 \leq \mathbf{h}_{i^\star}$.
3: Evaluate the optimal control input as $u_0^\star = \mathbf{f}_{i^\star}^T\mathbf{x}_0 + g_{i^\star}$.
4: Implement $u_0^\star$ and return to step 1.

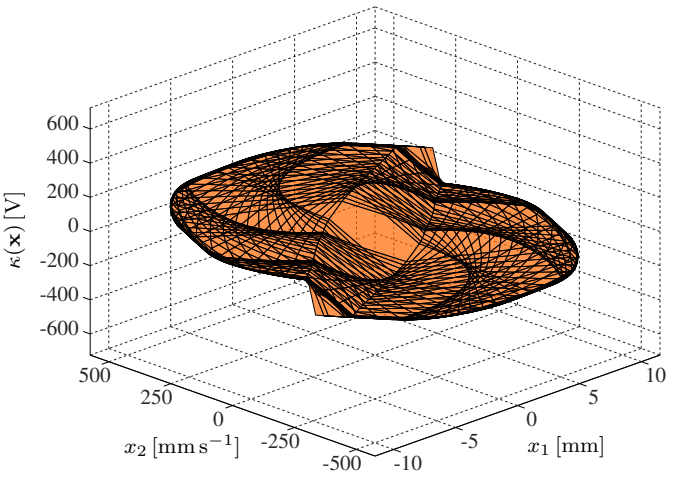


Fig. 1. The explicit MPC vibration controller we aim to implement ($N = 50$, $\overline{\kappa} = 100$, $\underline{\kappa} = -100$, $R = 5207$, $|\mathcal{I}_{\rm uns}| = 679$, $|\mathcal{I}_{\rm max}| = |\mathcal{I}_{\rm min}| = 2264$).

embedded control hardware, which has given rise to numerous techniques over the last decade, focused on reduction of both memory and runtime complexity of EMPC; see e.g. [36] and [37] for an overview of recent lower-complexity implementations and faster point location algorithms, respectively.

The explicit MPC vibration controller for the experimental system (to be introduced in Section III) described by Eq. (1) is illustrated in Fig. 1. The PWA feedback law was obtained assuming the sampling period of 20 ms and the prediction horizon of 50 steps. The underlying state-space partition $\{\mathcal{R}_i\}_{i=1}^{R}$ consists of 5207 regions and is omitted for clarity. More details on the computation are given in Section III.

C. Efficient embedded EMPC via convex lifting

This section aims to introduce the main algorithmic developments towards memory and time-efficient embedded vibration EMPC implementations using the geometric concept of convex lifting.

With respect to the scope of this paper we restrict ourselves to the essential theoretical concepts and focus on the controller synthesis and its practical deployment on embedded control hardware. A comprehensive overview of theoretical and structural properties of the convex lifting can be found in [38].

To put forward the algorithmic implementation, let us first recall some formal and necessary definitions.

Definition 1. A finite collection of $R \in \mathbb{N}_+$ full-dimensional polyhedra $\mathcal{R}_i \subset \mathbb{R}^{n_x}$, denoted by $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$, is called a polyhedral partition of a polyhedron $\mathcal{R} \subseteq \mathbb{R}^{n_x}$ if:

- $\mathcal{R} = \bigcup_{i\in\mathcal{I}_R} \mathcal{R}_i$,
- $\mathrm{int}(\mathcal{R}_i) \cap \mathrm{int}(\mathcal{R}_j) = \emptyset$ with $i \neq j,\ (i,j) \in \mathcal{I}_R^2$.

Regions $(\mathcal{R}_i, \mathcal{R}_j)$ are referred to as neighbors if $(i,j) \in \mathcal{I}_R^2$, $i \neq j$ and $\dim(\mathcal{R}_i \cap \mathcal{R}_j) = n_x - 1$. Also, if $\mathcal{R}$ is a polytope (bounded polyhedron), then $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$is called a polytopic partition. In addition, a cell complex of polyhedron $\mathcal{R}$ is defined as a polyhedral partition whose facet-to-facet property holds, meaning any pair of neighboring regions share a common facet [39]. In the context of MPC problem, $\mathcal{R}$ constitutes its feasible set [20]. The definition of a convex lifting is given next.

Definition 2. Given a polyhedral (polytopic) state-space partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$ of a polyhedron (polytope) $\mathcal{R} \subseteq \mathbb{R}^{n_x}$, the function $z : \mathcal{R} \to \mathbb{R}$ with

$$z(\mathbf{x}) := \mathbf{a}_i^T \mathbf{x} + b_i \text{ for } \mathbf{x} \in \mathcal{R}_i, \tag{5}$$

$\mathbf{a}_i \in \mathbb{R}^{n_x}$, $b_i \in \mathbb{R}$, $\forall i \in \mathcal{I}_R$, is called a convex piecewise affine lifting (for brevity henceforth referred to as convex lifting) if the following conditions hold:

- $z(\mathbf{x})$ is continuous over $\mathcal{R}$,
- for each $i \in \mathcal{I}_R$, $z(\mathbf{x}) > \mathbf{a}_j^T \mathbf{x} + b_j$ for all $x \in \mathcal{R}_i \backslash \mathcal{R}_j$ and all $j \neq i$, $j \in \mathcal{I}_R$.

We remark that the partition that admits a convex lifting is a cell complex [28]. The cell complex characterization is indeed a necessary, but not a sufficient condition for the existence of a convex lifting; and holds for non-degenerate MPC problems. In the following, we will suppose the polyhedral/polytopic partition to have the properties of a cell complex. Note that even a convexly non-liftable partiton may be easily modified into a liftable one by appropriate hyperplane arrangement [38], [40].

Let us append some useful definitions to be used next [41].

Definition 3. Let $\overline{\kappa}$ and $\underline{\kappa}$ denote, respectively, the maximum and minimum values which the PWA function $\kappa(\mathbf{x}) = \mathbf{f}_i^T \mathbf{x} + g_i$, $i \in \mathcal{I}_R$ in (4) attains over its domain $\mathcal{R}$. Denote by $\mathcal{I}_{\rm max}$ ($\mathcal{I}_{\rm min}$) the index set of regions where $\kappa(\mathbf{x})$ is saturated at the maximum (minimum), i.e. $\kappa(\mathbf{x}) = \overline{\kappa}$ ($\kappa(\mathbf{x}) = \underline{\kappa}$) for all $\mathbf{x} \in \mathcal{R}_i$, $i \in \mathcal{I}_{\rm max}$ ($i \in \mathcal{I}_{\rm min}$), and let $\mathcal{I}_{\rm sat} = \mathcal{I}_{\rm max} \cup \mathcal{I}_{\rm min}$. A region $\mathcal{R}_i$ with $i \in \mathcal{I}_{\rm sat}$ is therefore called the saturated region either at the minimum or at the maximum. Otherwise $\mathcal{R}_i$ is called unsaturated, with the index set denoted by $\mathcal{I}_{\rm uns} = \mathcal{I}_R \setminus \mathcal{I}_{\rm sat}$.

Definition 4. Given a continuous PWA function $\kappa(\mathbf{x})$, defined over a parameter-space partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$, we call the PWA function $\tilde{\kappa}(\mathbf{x}) := \tilde{\mathbf{f}}_j^T \mathbf{x} + \tilde{g}_j$ a suitable augmentation of $\kappa(\mathbf{x})$ if the following properties hold:

- $\tilde{\kappa}(\mathbf{x})$ is defined over $\{\tilde{\mathcal{R}}_j\}_{j\in\mathcal{I}_{\tilde{R}}}$ such that $\bigcup_{j\in\mathcal{I}_{\tilde{R}}} \tilde{\mathcal{R}}_j = \bigcup_{i\in\mathcal{I}_R} \mathcal{R}_i$, i.e. $\mathrm{dom}(\tilde{\kappa}(\mathbf{x})) = \mathrm{dom}(\kappa(\mathbf{x}))$,
- $\tilde{\kappa}(\mathbf{x}) = \kappa(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{R}_{\mathcal{I}_{\rm uns}}$,
- $\tilde{\kappa}(\mathbf{x}) \geq \overline{\kappa},\ \forall \mathbf{x} \in \mathcal{R}_{\mathcal{I}_{\rm max}}$,
- $\tilde{\kappa}(\mathbf{x}) \leq \underline{\kappa},\ \forall \mathbf{x} \in \mathcal{R}_{\mathcal{I}_{\rm min}}$.

Herein, $\mathcal{R}_{\mathcal{I}}$ denotes the subset of regions $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$ for some index set $\mathcal{I} \subseteq \{1, \ldots, R\}$.

Note that the premise of existence of saturated regions does not hold in general, however, $|\mathcal{I}_{\rm uns}| \ll R$ is a common case in most practical MPC setups [36] (see the explicit vibration controller in Fig. 1 with $|\mathcal{I}_{\rm uns}| = 679$), and we show it can be efficiently exploited in the proposed controller implementation. Denoting the vertex set of a polytope $\mathcal{P}$ by $\mathcal{V}(\mathcal{P})$, let us recall the algorithmic procedure for construction of a convex lifting for a given state-space partition [28], herein aiming specifically at its unsaturated regions. It is summarized in Algorithm 2, and its outcome are the gains of a convex lifting defined over the unsaturated regions.

Note that the feasibility of the optimization problem (8) can serve as a necessary and sufficient condition for the existence of a convex lifting of a given partition. Since there exist in fact infinitely many candidates for $z(\mathbf{x})$ belonging to the epigraph

Algorithm 2 Construction of a convex lifting for unsaturated regions of a given polytopic partition.

Input: Polytopic partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$, a constant $c>0$.
Output: Convex lifting gains $(\mathbf{a}_i, b_i)$, $\forall i \in \mathcal{I}_{\rm uns}$.

1: Find the index set of unsaturated regions $\mathcal{I}_{\rm uns} \subseteq \mathcal{I}_R$.
2: Register all pairs of neighboring regions in $\mathcal{R}_{\mathcal{I}_{\rm uns}}$.
3: For each pair of neighboring regions $(\mathcal{R}_i, \mathcal{R}_j), (i,j) \in \mathcal{I}^2_{\rm uns}$, add:
- continuity conditions $\forall \mathbf{v} \in \mathcal{V}(\mathcal{R}_i \cap \mathcal{R}_j)$:

$$\mathbf{a}_i^T\mathbf{v} + b_i = \mathbf{a}_j^T\mathbf{v} + b_j; \tag{6}$$

- convexity conditions $\forall \mathbf{u} \in \mathcal{V}(\mathcal{R}_i), \mathbf{u} \notin \mathcal{V}(\mathcal{R}_i \cap \mathcal{R}_j)$:

$$\mathbf{a}_i^T\mathbf{u} + b_i \geq \mathbf{a}_j^T\mathbf{u} + b_j + c. \tag{7}$$

4: Solve the following convex optimization problem by minimizing a chosen cost function, e.g.,

$$\min_{\mathbf{a}_i, b_i} \sum_{i\in\mathcal{I}_{\rm uns}} (\mathbf{a}_i^T\mathbf{a}_i + b_i^2) \quad \text{subject to} \quad (6)\text{–}(7). \tag{8}$$

to obtain the gains of a convex lifting

$$\underline{z}^{\rm uns}(\mathbf{x}) := \mathbf{a}_i^{\star T}\mathbf{x} + b_i^\star = \ell^{\rm uns}(\mathbf{x}), \text{ for any } \mathbf{x} \in \mathcal{R}_{\mathcal{I}_{\rm uns}}.$$

of $\underline{z}(\mathbf{x})$, let us exclusively denote the convex lifting $\underline{z}(\mathbf{x})$ obtained as per Algorithm 2 by $\ell(\mathbf{x})$, in this case $\ell^{\rm uns}(\mathbf{x})$, which is to be exploited by the two efficient EMPC implementation techniques presented next. Before proceeding, let us therefore denote the quantities resulting from Section II-C1 and II-C2 by superscripts I and II, respectively.

1) Inverse parametric optimization based implementation: In the following we recall a recent technique that exploits the concept of inverse parametric convex programming via convex liftings [28], [42], [43]. It will be briefly described here since it had drawn attention to the interesting properties of convex lifting that ultimately led to the main developments presented in Section II-C2; and was also shown to be relevant in robust and explicit MPC design [27], [29].

Inverse parametric convex programming is defined as an inverse optimality problem of parametric convex programming (pCP), which aims to build an alternative optimization problem characterized by an appropriate constraint set and a cost function such that its optimal solution coincides with the one of an original problem. In particular, the goal of inverse parametric linear/quadratic programming (IpL/QP) is to construct a linear constraint set and a linear/quadratic cost function such that the optimal solution of this newly formulated problem is equivalent to a given PWA function defined over a given polyhedral partition. Construction of such an optimization problem based on convex lifting was proposed in [28] and provided a novel perspective on the structural link between linear MPC design and pCP that can be stated as follows: every continuous PWA control law can be recovered via a linear optimal control problem with control horizon at most equal to 2 prediction steps.

Considering e.g. the IpLP problem, a given continuous PWA function $\kappa(\mathbf{x})$ defined over a polyhedral partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$ is the image via the orthogonal projection onto $\mathbb{R}^{n_u}$ ($=\mathbb{R}$ in this case) of the optimal solution to the parametric LP below [28]:

$$\min_{[z\ u]^T} z \quad \text{subject to} \quad \begin{bmatrix}\mathbf{x}^T & z & u\end{bmatrix}^T \in \Pi_{[\mathbf{x}^T\ z\ u]^T}, \tag{9}$$

where z represents a 1–dimensional auxiliary 'lifting' variable, and the constraint set $\Pi_{[\mathbf{x}^T\ z\ u]^T}$ is obtained as follows:

$$V_{[\mathbf{x}^T\ z\ u]^T} := \left\{ \begin{bmatrix}\mathbf{v}^T & \ell(\mathbf{v}) & \kappa(\mathbf{v})\end{bmatrix}^T : \mathbf{v} \in \bigcup_{i\in\mathcal{I}_R} \mathcal{V}(\mathcal{R}_i) \right\}, \tag{10}$$

$$\Pi_{[\mathbf{x}^T\ z\ u]^T} := \operatorname{conv}\left(V_{[\mathbf{x}^T\ z\ u]^T}\right), \tag{11}$$

and can be equivalently expressed as $\mathbf{H}_x\mathbf{x} + \mathbf{H}_z z + \mathbf{H}_u u \leq \mathbf{K}$. The vertex set $V_{[\mathbf{x}^T\ z\ u]^T} = \mathcal{V}(\Pi_{[\mathbf{x}^T\ z\ u]^T})$ is simply composed of the partition vertices $\mathbf{v} \in \bigcup_{i\in\mathcal{I}_R}\mathcal{V}(\mathcal{R}_i)$ appended by corresponding values of convex lifting $\ell(\mathbf{v})$ and control action $\kappa(\mathbf{v})$ in the augmented space $\mathbb{R}^{n_x+1+n_u}$, in case of system (1), $\mathbb{R}^4$.

Since the solution of the 'horizon 2' IpLP (9) exactly recovers the original 'horizon N' pLP/pQP-based EMPC solution, its structural complexity and hence also online implementation effort remain the same. Let us therefore recall the extension proposed recently in [29], aiming at practical EMPC problems featuring active input constraints, i.e. a large number of saturated regions which typically inhibit the direct convex liftability. The so-called extended IpLP procedure is summarized in Algorithm 3. It starts by finding the lifting $\ell^{\rm uns}(\mathbf{x})$ to be used for construction of a polyhedron $\Pi^{\rm uns}_{[\mathbf{x}^T\ z\ u]^T}$ in the augmented space. This is then exploited by keeping only the constraints (facets of $\Pi^{\rm uns}_{[\mathbf{x}^T\ z\ u]^T}$) which practically contribute to the optimal solution [29], yielding a constraint set $\tilde{\Pi}$ of the extended IpLP (12). Its solution consists of $\tilde{\ell}^{\rm I}(\mathbf{x})$, $\tilde{\kappa}^{\rm I}(\mathbf{x})$ defined over

Algorithm 3 Extended IpLP with clipping [29]

Input: Saturated continuous PWA function $\kappa(\mathbf{x})$ defined over a polytopic partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$ of a polytope $\mathcal{R}\subset\mathbb{R}^{n_x}$.
Output: $\tilde{\kappa}^{\rm I}(\mathbf{x})$, $\phi(\tilde{\kappa}^{\rm I}(\mathbf{x}))$, $\tilde{\ell}^{\rm I}(\mathbf{x})$, defined over $\{\tilde{\mathcal{R}}^{\rm I}_j\}_{j\in\mathcal{I}_{\tilde{R}^{\rm I}}}$.

1: Compute the gains of convex lifting $\ell^{\rm uns}(\mathbf{x})$ corresponding to a possibly non-convex collection of unsaturated regions $\{\mathcal{R}_i\}_{i\in\mathcal{I}_{\rm uns}}$ via Algorithm 2.
2: Construct the set $\Pi^{\rm uns}_{[\mathbf{x}^T\ z\ u]^T}$, defined over $\mathcal{R}_{\mathcal{I}_{\rm uns}}$, analogously to (10)–(11).
3: Form a new constraint set $\tilde{\Pi}$ associated with $\{\tilde{\mathcal{R}}^{\rm I}_j\}_{j\in\mathcal{I}_{\tilde{R}^{\rm I}}}$, as described in Algorithm 3.1 in [29].
4: Formulate and solve the extended IpLP problem with the constraints on $\mathbf{x}$, z, u given by the polyhedron $\tilde{\Pi}$ to obtain:

$$\begin{bmatrix}\tilde{z}^\star(\mathbf{x}) \\ \tilde{u}^\star(\mathbf{x})\end{bmatrix} = \arg\min_{[z\ u]^T} z \quad \text{s.t.} \quad \begin{bmatrix}\mathbf{x}^T & z & u\end{bmatrix}^T \in \tilde{\Pi}. \tag{12}$$

5: Extract the appropriate component of the above optimizer:

$$\tilde{u}^\star(\mathbf{x}) = \operatorname{Proj}_u \begin{bmatrix}\tilde{z}^\star(\mathbf{x}) \\ \tilde{u}^\star(\mathbf{x})\end{bmatrix} = \tilde{\kappa}^{\rm I}(\mathbf{x}) \tag{13}$$

to obtain a PWA function $\tilde{\kappa}^{\rm I}(\mathbf{x})$ defined over a rearranged partition $\{\tilde{\mathcal{R}}^{\rm I}_j\}_{j\in\mathcal{I}_{\tilde{R}^{\rm I}}}$. The other component, $\tilde{z}^\star(\mathbf{x}) = \tilde{\ell}^{\rm I}(\mathbf{x})$ denotes the corresponding convex lifting.
6: Employ a clipping filter $\phi(\cdot)$ as per (14) to maintain equivalence between $\tilde{\kappa}^{\rm I}(\mathbf{x})$ and $\kappa(\mathbf{x})$.

$\{\tilde{\mathcal{R}}_j^{\rm I}\}_{j\in\mathcal{I}_{\tilde{R}^{\rm I}}}$—a rearranged partition of $\mathcal{R}$.

The PWA function $\tilde{\kappa}^{\rm I}(\mathbf{x})$ (13) obtained from the extended IpLP (12) is by Definition 4 a suitable augmentation of $\kappa(\mathbf{x})$ (4), i.e. only the portion defined over $\mathcal{R}_{\mathcal{I}_{\rm uns}}$ is retrieved exactly. To establish the equivalence between the two over the entire feasible domain $\mathcal{R}=\mathrm{dom}(\kappa(\mathbf{x}))$, and thus the optimality, we employ a clipping filter $\phi(\cdot)$ adopted from [41] as follows:

$$\kappa(\mathbf{x})=\phi(\tilde{\kappa}(\mathbf{x})):=\begin{cases}\overline{\kappa} & \text{if } \tilde{\kappa}(\mathbf{x})\geq\overline{\kappa},\\ \underline{\kappa} & \text{if } \tilde{\kappa}(\mathbf{x})\leq\underline{\kappa},\\ \tilde{\kappa}(\mathbf{x}) & \text{otherwise.}\end{cases}\tag{14}$$

This way $\tilde{\kappa}(\mathbf{x})$ inherits all the performance, closed-loop stability and feasibility guarantees of the original EMPC feedback.

We remark that if the EMPC problem yields a partition with $|\mathcal{I}_{\rm uns}|\ll R$, then $|\mathcal{I}_{\rm uns}|\leq\tilde{R}^{\rm I}\ll R$ tends to hold as well [29]. While in [29] we proposed the traditional EMPC implementation via point location, yet using the lower-complexity problem data given by $\tilde{\kappa}^{\rm I}(\mathbf{x})$, $\{\tilde{\mathcal{R}}_j^{\rm I}\}_{j\in\mathcal{I}_{\tilde{R}^{\rm I}}}$, in the following we show that it may be performed in a substantially more efficient—regionless fashion, employing the lifting $\ell^{\rm uns}(\mathbf{x})$ instead.

2) 'Pure' convex lifting based implementation:
Recall that the output of Algorithm 2 are the gains $(\mathbf{a}_i,b_i),\forall i\in\mathcal{I}_{\rm uns}$ of a convex lifting $\ell^{\rm uns}(\mathbf{x})$, defined over $\mathcal{R}_{\mathcal{I}_{\rm uns}}$. Now, let us define the following convex lifting:

$$\tilde{\ell}^{\rm II}(\mathbf{x}):=\max_{i\in\mathcal{I}_{\rm uns}}\ (\mathbf{a}_i^T\mathbf{x}+b_i)\ \text{ for }\ \mathbf{x}\in\mathcal{R}.\tag{15}$$

Since $\tilde{\ell}^{\rm II}(\mathbf{x})=\ell^{\rm uns}(\mathbf{x})$ for all $\mathbf{x}\in\bigcup_{i\in\mathcal{I}_{\rm uns}}\mathcal{R}_i$, let us for ease of presentation denote the polytopic partition of $\mathcal{R}$ associated with $\tilde{\ell}^{\rm II}(\mathbf{x})$ (15) by $\{\tilde{\mathcal{R}}_j^{\rm II}\}_{j\in\mathcal{I}_{\tilde{R}^{\rm II}}}$, $\tilde{R}^{\rm II}=|\mathcal{I}_{\rm uns}|$. The corresponding PWA control law has the following property [38]:

$$\tilde{\kappa}^{\rm II}(\mathbf{x})=\tilde{\mathbf{f}}_j^T\mathbf{x}+\tilde{g}_j=\mathbf{f}_i^T\mathbf{x}+g_i\ \text{ for }\ \mathbf{x}\in\tilde{\mathcal{R}}_j^{\rm II},\\ \text{such that }\ i\in\mathcal{I}_{\rm uns},\ \mathcal{R}_i\subseteq\tilde{\mathcal{R}}_j^{\rm II}.\tag{16}$$

Denoting $\tilde{\ell}^{\rm II}(\mathbf{x})$ as $\tilde{\ell}^{\rm II}(\mathbf{x})=\tilde{\mathbf{a}}_j^T\mathbf{x}+\tilde{b}_j$ for $\mathbf{x}\in\tilde{\mathcal{R}}_j^{\rm II}$, it may be used within online implementation in the following fashion:

$$\mathbf{x}_0\in\tilde{\mathcal{R}}_{j^\star}^{\rm II}\ \Leftrightarrow\ \tilde{\mathbf{a}}_{j^\star}^T\mathbf{x}_0+\tilde{b}_{j^\star}=\max_{j\in\mathcal{I}_{\tilde{R}^{\rm II}}}\ (\tilde{\mathbf{a}}_j^T\mathbf{x}_0+\tilde{b}_j).\tag{17}$$

This property enables to easily identify index $j^\star$ of the $j^\star$–th affine control law to be evaluated at a given $\mathbf{x}_0\in\mathcal{R}$, without the need to search for $\tilde{\mathcal{R}}_{j^\star}^{\rm II}$ by traversing $\{\tilde{\mathcal{R}}_j^{\rm II}\}_{j\in\mathcal{I}_{\tilde{R}^{\rm II}}}$, yielding a regionless EMPC controller. As in the case of $\tilde{\kappa}^{\rm I}(\mathbf{x})$, optimality of the PWA feedback $\tilde{\kappa}^{\rm II}(\mathbf{x})$ (possibly discontinuous over $\mathcal{R}_{\mathcal{I}_{\rm sat}}$) for any $\mathbf{x}\in\mathcal{R}$ can be achieved by taking $\phi(\tilde{\kappa}^{\rm II}(\mathbf{x}))$.

We remark that an explicit MPC implementation exploiting property (17) was proposed in [44], yet employing the optimal cost function, which for the explicit solutions based on pLP (only if the optimizer is unique) represents nothing else than a convex lifting from the geometrical viewpoint. However, in the pQP case such approach is no longer applicable.

The online (convex) Lifting-based EMPC (LEMPC) implementation is summarized in Algorithm 4. Note that executing Step 2 practically amounts to a mere searching for the maximum among the list $\{\tilde{\mathbf{a}}_j^T\mathbf{x}_0+\tilde{b}_j\}_{j=1}^{\tilde{R}^{\rm II}}$ by sequential comparing of its components evaluated at $\mathbf{x}_0$. A more illustrative, pseudocode implementation can be found in Appendix.

Algorithm 4 Efficient regionless on-line EMPC implementation using convex lifting and clipping

1: At each sampling instant t, measure or estimate the current state $\mathbf{x}_0=\mathbf{x}(t)$. If $\mathbf{x}_0\in\mathcal{R}$, proceed to step 2.
2: Find index $j^\star\in\mathcal{I}_{\tilde{R}^{\rm II}}$ such that
$$j^\star=\arg\max_{j\in\mathcal{I}_{\tilde{R}^{\rm II}}}\ (\tilde{\mathbf{a}}_j^T\mathbf{x}_0+\tilde{b}_j).$$
3: Evaluate the optimal control input as per (14), encoded as
$$u_0^\star=\max\left\{\underline{\kappa},\min\left\{\tilde{\mathbf{f}}_{j^\star}^T\mathbf{x}_0+\tilde{g}_{j^\star},\overline{\kappa}\right\}\right\}.$$
4: Implement $u_0^\star$ and return to step 1.

The interested reader is referred to [28], [38] for a comprehensive theoretical background on the convex lifting concept. It is also shown therein that both the constructive and the implementation procedure may be easily generalized for systems with multiple inputs. An alternative construction of $\ell(x)$ based on halfspace representation of $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$ is shown in [38] and shall be preferred in case of higher-dimensional systems since it allows to avoid the vertex enumeration. Scalability of both vertex and halfspace based construction are also shown there.

Note that a component of explicit solution of the extended IpLP (12) in Section II-C1, the convex lifting $\tilde{\ell}^{\rm I}(\mathbf{x})$, can be also employed to accelerate the online EMPC implementation using Algorithm 4. However, in contrast to the latter, simplified procedure relying only on construction of the convex lifting $\ell^{\rm uns}(\mathbf{x})$ itself (Algorithm 2), obtaining $\tilde{\ell}^{\rm I}(\mathbf{x})$ per Algorithm 3 requires an extra offline effort spent on constructing the constraint set $\tilde{\Pi}$ for the LP (12) and its subsequent explicit solution, i.e. steps 2–5, which renders this variant from computational perspective clearly redundant.

3) Complexity analysis:
In terms of complexity, let us first assess the memory footprint of particular EMPC implementations described above. Taking the original EMPC controller as a reference, the total memory consumed by the PWA feedback $\kappa(\mathbf{x})$ and its underlying partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}_R}$, given in general ($\mathbf{u}\in\mathbb{R}^{n_u}$) by $\{\mathbf{F}_i,\mathbf{g}_i,\mathbf{H}_i,\mathbf{h}_i\}$, is $(n_x+1)\sum_{i=1}^R(n_{\rm h}^i+n_u)$ real numbers, where $n_{\rm h}^i$ is number of halfspaces defining the i-th region. On the other hand, memory footprint of the regionless LEMPC controllers (either I or II), given by $\{\tilde{\mathbf{F}}_j,\tilde{\mathbf{g}}_j,\tilde{\mathbf{a}}_j,\tilde{b}_j\}$, amounts to $\tilde{R}(1+n_u)(n_x+1)+2n_u$ real numbers, where the second negligible term represents memory needed to encode $\{\underline{\kappa},\overline{\kappa}\}$ used by the clipping function ϕ. Recall that in case of variant II (Sect. II-C2) $\tilde{R}^{\rm II}=|\mathcal{I}_{\rm uns}|$; while $\tilde{R}^{\rm I}\geq|\mathcal{I}_{\rm uns}|$ in general.

Secondly, let us quantify the necessary online computational effort. The critical time-consuming task in the standard EMPC implementation is the point location, i.e. finding the index i of the region $\mathcal{R}_i$ that contains $\mathbf{x}_0$, followed by simple evaluation of the associated i-th affine control law. Performing the direct sequential search per Algorithm 1 amounts in the worst case to $\sum_{i=1}^R n_{\rm h}^i(2n_x+1)+2n_un_x$ floating point operations (FLOPs). The regionless LEMPC implementation can solve this problem in an efficient way exploiting clipping (Algorithm 4), avoiding the expensive region-based point location. In total, it requires a

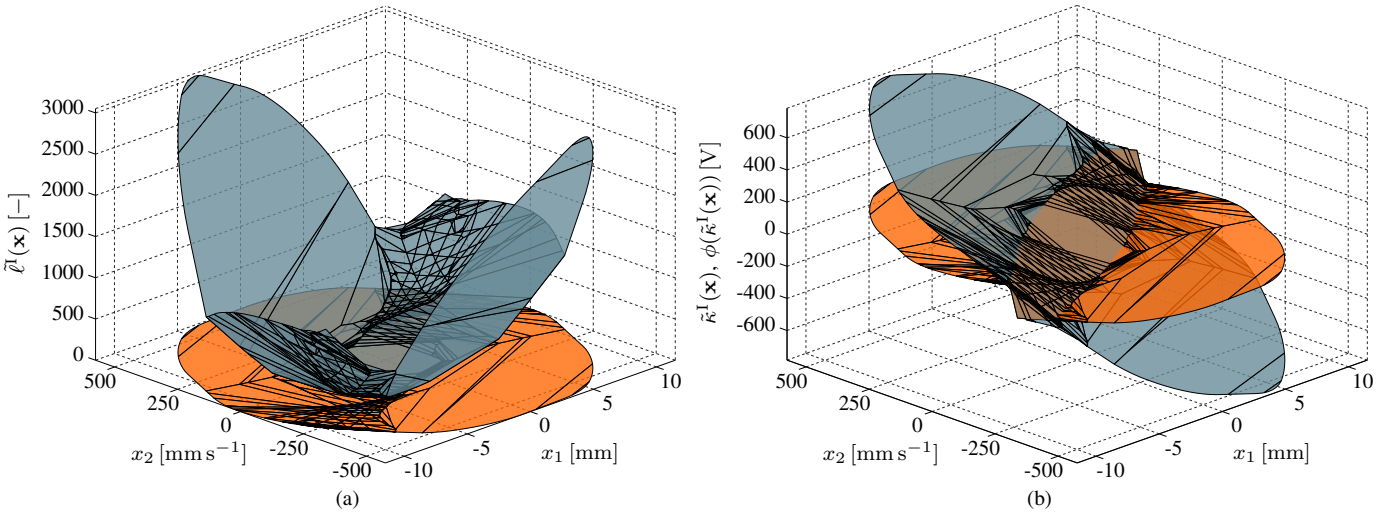


Fig. 2. Illustration of the inverse parametric optimization ($N = 2$) based EMPC implementation (acc. Section II-C1) for the experimental vibration system. (a) A convex lifting $\tilde{\ell}^{\mathrm{I}}(\mathbf{x})$ and the associated state-space partition $\tilde{\mathcal{R}}^{\mathrm{I}}$ (for illustration only). (b) The recovered PWA control law $\tilde{\kappa}^{\mathrm{I}}(\mathbf{x})$ in blue, and the result of clipping $\phi(\tilde{\kappa}^{\mathrm{I}}(\mathbf{x}))$ at $\overline{\kappa} = 100$ and $\underline{\kappa} = -100$ in orange color (illustration only, cf. Fig. 1); $\tilde{R}^{\mathrm{I}} = 1160$.

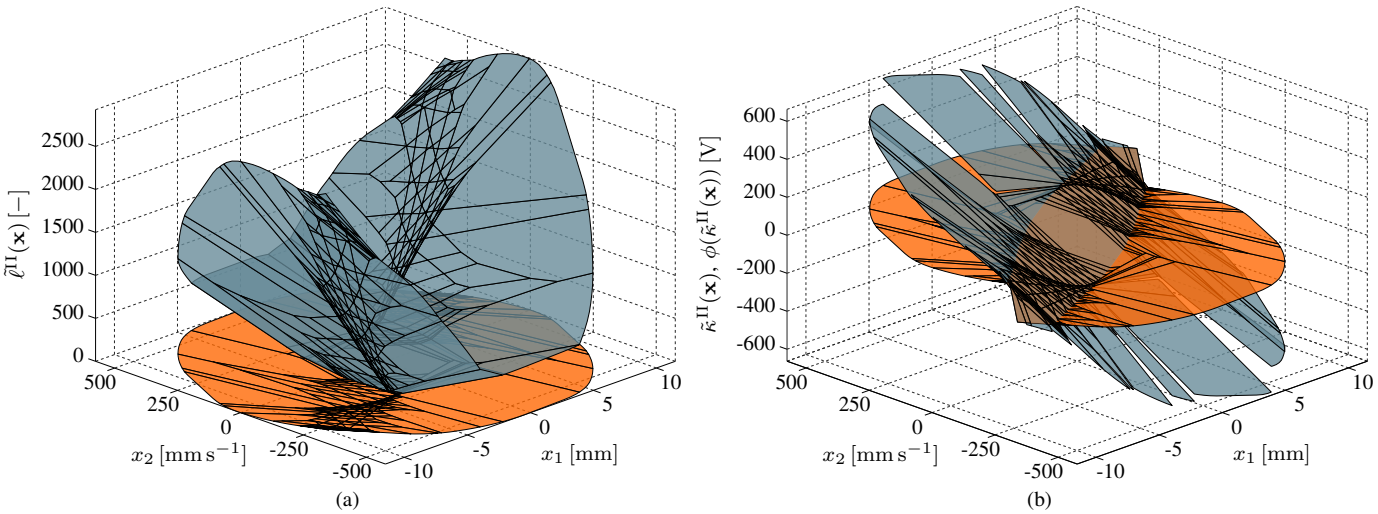


Fig. 3. Illustration of the convex lifting based EMPC implementation (acc. Section II-C2) for the experimental vibration system. (a) A convex lifting $\tilde{\ell}^{\mathrm{II}}(\mathbf{x})$ and the associated state-space partition $\tilde{\mathcal{R}}^{\mathrm{II}}$ (for illustration only). (b) The associated PWA control law $\tilde{\kappa}^{\mathrm{I}}(\mathbf{x})$ in blue, and the result of clipping $\phi(\tilde{\kappa}^{\mathrm{I}}(\mathbf{x}))$ at $\overline{\kappa} = 100$ and $\underline{\kappa} = -100$ in orange color (illustration only, cf. Fig. 1); $\tilde{R}^{\mathrm{II}} = |\mathcal{I}_{\mathrm{uns}}| = 679$.

constant number of $2\tilde{R}n_x+2n_un_x+2n_u$ FLOPs ($\tilde{R}^{\mathrm{II}} = |\mathcal{I}_{\mathrm{uns}}|$), which implies a significant reduction in runtime complexity; even if $|\mathcal{I}_{\mathrm{uns}}| = R$ was the case, while typically $|\mathcal{I}_{\mathrm{uns}}| \ll R \ll \sum_{i=1}^{R} n_{\mathrm{h}}^i$.

Note that none of the terms above consider storing or evaluating the full optimizer $\mathbf{U}^\star$ with Nn_u elements since only its first element $\mathbf{u}_0^\star$ is required for implementation in a receding horizon fashion. The offline computation times needed to construct $\tilde{\ell}(\mathbf{x})$, $\tilde{\kappa}(\mathbf{x})$ are reported in Sect. III-A for completeness.

One may also compare the above LEMPC techniques with other complexity reduction approaches in EMPC, e.g. with the clipping-based one in [41] as they both exploit the concept of clipping. The latter, however, relies on replacing the saturated regions with extensions of the unsaturated ones, for which the achievable reduction may range from none ($\tilde{R}^{[41]} = R$) to the case when the new partition of $\mathcal{R}$ has $\tilde{R}^{[41]} = |\mathcal{I}_{\mathrm{uns}}|$ regions. Another approach of [36] requires to store only the unsaturated regions by devising a separating function. Clearly, the family of these methods still requires storing a modified state-space partition, and hence performing the traditional point location. On the contrary, techniques aiming at faster online evaluation in EMPC usually come at a price of a larger memory footprint or preprocessing time [37], [45]. Alternatively, a memory-efficient regionless EMPC implementation was proposed recently in [46], combining the approaches of [47] and [48]. Its nature renders it applicable for problems with even moderately large parametric space, however, only very short prediction horizon; and does not imply reduction in evaluation effort. LEMPC, as indicated above, enables a significant reduction in both storage and online computation requirements with a relatively low preprocessing load. For completeness we remark that study [49] recently presented a linear machine concept, which is similar to convex lifting presented in this paper. However, necessary and sufficient conditions for the existence of a convex lifting are not available therein, while these are stated in our previous studies such as [28], [38].

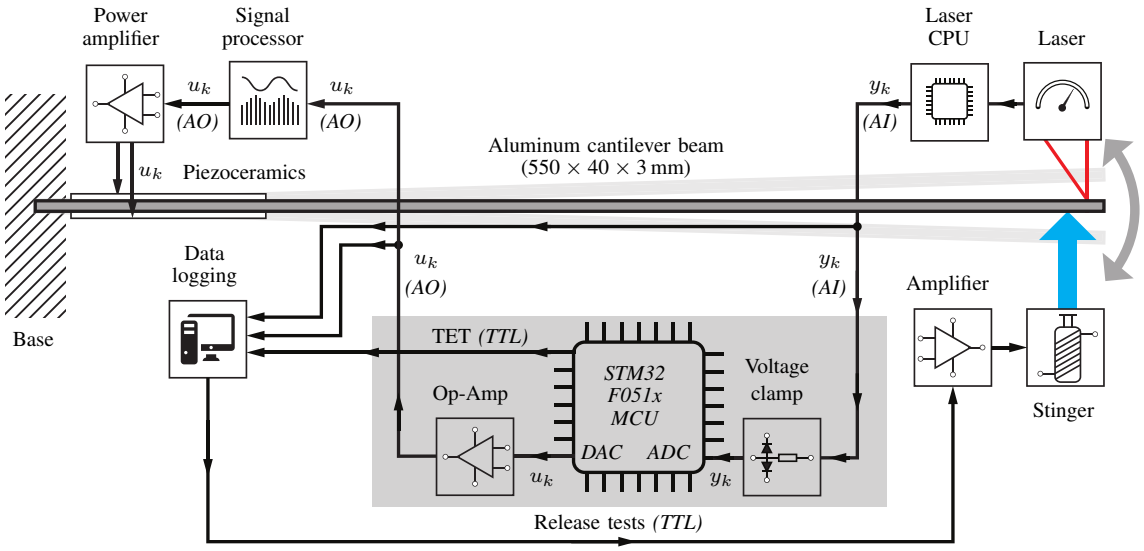


Fig. 4. Schematic representation of the experimental setup.

The concepts presented in Sections II-C1 and II-C2, adopted for the experimental vibrating system described by Eq. (1), are illustrated in Figs. 2 and 3, respectively. In particular, Fig. 3a visualizes the piecewise affine convex lifting $\tilde{\ell}^{\mathrm{II}}(\mathbf{x})$ (15) for all $\mathbf{x} \in \mathcal{R}$. We recall that the associated state-space partition $\{\tilde{\mathcal{R}}_j^{\mathrm{II}}\}_{j\in\mathcal{I}_{\tilde{R}^{\mathrm{II}}}}$ needs not to be constructed, and is depicted for illustration only. Fig. 3b in blue shows the associated control law $\tilde{\kappa}^{\mathrm{II}}(\mathbf{x})$ (16). From the geometrical point of view, these can be interpreted as continuous extensions of $\ell^{\mathrm{uns}}(\mathbf{x})$ and $\kappa(\mathbf{x})$, $\forall \mathbf{x} \in \bigcup_{i\in\mathcal{I}_{\mathrm{uns}}} \mathcal{R}_i$, respectively, over the entire feasible domain $\mathcal{R}$. The number of local control laws and convex lifting terms of such a regionless EMPC implementation is $\tilde{R}^{\mathrm{II}} = |\mathcal{I}_{\mathrm{uns}}| = 679$. On the other hand, Figs. 2a and 2b illustrate the convex lifting $\tilde{\ell}^{\mathrm{I}}(\mathbf{x})$ and the PWA feedback $\tilde{\kappa}^{\mathrm{I}}(\mathbf{x})$, both obtained as optimal solution of the 'horizon 2' extended IpLP (12). These can be geometrically interpreted as orthogonal projections of $\tilde{\Pi}$ onto the respective subspaces. Number of the affine lifting as well as feedback terms in this case is $\tilde{R}^{\mathrm{I}} = 1160$. Recall that in case of the original MPC problem (3) with $N = 50$ solved as pQP the number of regions was $R = 5207$. Finally, the result of clipping per (14) allowing to maintain the equivalence between $\tilde{\kappa}^{\mathrm{II}}(\mathbf{x})$ ($\tilde{\kappa}^{\mathrm{I}}(\mathbf{x})$) and $\kappa(\mathbf{x})$ is visualized in Fig. 3a (2a) in orange color; cf. Fig. 1.

III. Experimental Implementation

A. Active Beam Test

The active cantilever beam first introduced in Section II-A is illustrated in Fig. 5, where the fixed-free aluminum beam is shown mounted to a sturdy base. The deflection of the free end of the beam is measured by a Keyence LK-G80 spot-type laser triangulation head, providing its signal to a Keyence LK-G3000 series central processing unit. The beam is actuated by a pair of cross-coupled MIDÉ QP16N piezoceramic actuators on its clamped end, receiving the same amplified input signal with inverted polarity.

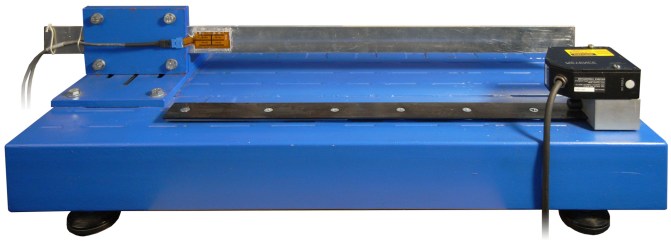


Fig. 5. Laboratory device with the actively controlled mechanical structure.

The configuration of the experimental setup is summarized in the simplified scheme shown in Fig. 4. The LEMPC algorithms introduced in Sect. II-C1 and Sect. II-C2 were deployed in the stand-alone mode to a microcontroller unit. The inputs generated by the controller are fed to an operational amplifier (Texas Instruments TLC2272CP), then to a signal processor that shifts signal levels to a bipolar configuration (Advantech ADAM 3014) needed for the final capacitive amplifier (MIDÉ EL-1224) feeding the piezoceramic actuators. The input is read directly by the microcontroller in the form of a single analog signal from the laser triangulation system.

The cantilever beam had been tested in a scenario that is frequently referred to as a release test. Release tests are often used to emulate transient mechanical behavior often encountered in aerospace applications in the laboratory environment [31]. In a release test, the free end of the beam is deflected by a force to an initial position, and then released to reach its equilibrium with or without feedback control. The repeat release tests were generated by a stinger mechanism (not shown in Fig. 5), that pushes the end of the beam away from its zero position upon receiving a digital signal. The experiment itself is launched, controlled and logged using an external computer, sending the engage signals to the release tests, and capturing input, output and timing data. This external computer contains a laboratory measurement card and runs a simple experiment control and data logging program under the Simulink Real-Time algorithm

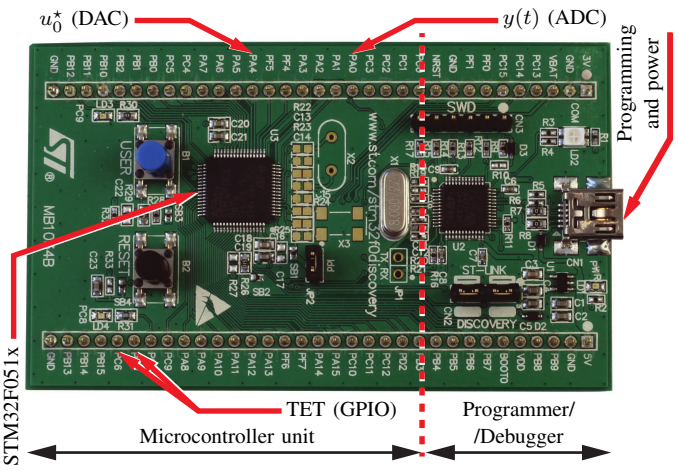


Fig. 6. STM32 F0 Discovery prototyping board with the F051x microcontroller.

prototyping suite.

The microcontroller used in the experimental testing was an STMicroelectronics (STM) 32-bit (STM32) F051x-series microcontroller unit, specifically STM32 F051R8T6. This MCU features 64 kB non-volatile read only memory (ROM) and 8 kB volatile random access memory (RAM). The MCU considered for the laboratory test and the rest of the investigated devices belong to a family of microcontrollers that use the very popular ARM Cortex-M core design. The ARM Cortex-M family considered in this paper represents modern 32-bit embedded devices that are used in a wide variety of applications, including control systems technology. The F051x-series MCU in the laboratory test incorporates the basic M0 core architecture and is marketed as a low-cost entry-level device with a unit cost of approximately $1.50 for large quantities. The maximum clock speed of 48 MHz is more than that of the typical 20 MHz of budget 8-bit devices, however, it is still a true embedded device without the power and possibilities of its more expensive counterparts.

This microcontroller was tested using a STM32 F0 Discovery prototyping and evaluation board that integrates the MCU with a programmer and debugger and provides an easy access to the physical pins of the MCU. The prototyping board employed in the beam tests is shown in Fig. 6. Input is generated and output is acquired via the integrated digital-to-analog and analog-to-digital converters of the MCU. The task execution timing (TET) is sent as an output to the data logging computer in the form of a simple digital signal that is logical true when the algorithm runs, and false when it idles. The board outputs two TET signals, one for the LEMPC algorithm only, and the other for the entire feedback control algorithm including input and output data processing and state estimation.

The peripheral initialization code for the processor was developed using the STM32CubeMX initialization code generator. Subsequently, the rest of the program was finished and compiled to target using the IAR Embedded Workbench for ARM (EWARM) in C-language. The feedback LEMPC algorithms proposed in Sect. II-C received state estimates from a time-varying linear Kalman filter employing the model from Sect. II-A. Input and output measurements were re-scaled linearly according to the input and output measurement chains.

Finally, let us shed more light on the essential computation of the controllers discussed so far and used in the experiments. As described in Sect. II-A, the beam dynamics (1) was experimentally identified and then discretized with $T_\mathrm{s} = 0.02\,\mathrm{s}$. The sampling time was chosen so as to enable implementation on low-cost hardware and it is also related to the dominant resonant frequency of the beam, such that the closed-loop system is sampled over six times an oscillation period. The MPC problem (3) was formulated with control horizon of $N = 50$ steps which practically amounts to the prediction over 1 second. The choice for this horizon is motivated by the stability guarantees assumed in the problem formulation and allows the expected range of positions and velocities to be covered by the controller's domain of attraction, see [25]. In addition, state and input penalties were chosen as $\mathbf{Q} = \mathbf{I}$ and $\mathbf{R} = 1$, respectively, $\mathbf{P}_N$ set to the solution of DARE and $\mathcal{C}_\infty$ designed as the maximal control invariant set—rendering the controller response such that it attenuates the beam tip vibration efficiently and yet does not behave like an overly aggressive 'bang-bang'-like controller. The controllers used symmetric $\pm 100\,\mathrm{V}$ bounds to prevent depolarization of the piezoceramics. We remark that although state constraints are also allowed by the formulation, it is not practical to impose them on this process as it would unnecessarily restrict the performance of vibration damping and their feasibility cannot be guaranteed in presence of transient excitation. The resulting MPC problem was solved parametrically using the Multi-Parametric Toolbox (MPT) [50]. The explicit MPC feedback in Fig. 1 was obtained in 10 min on a 2.2 GHz i7 core CPU with 8 GB of RAM running MATLAB 8.5, MPT3 and YALMIP [51]. The respective data were used to construct the LEMPC controllers per Sect. II-C1 and II-C2, visualized in Figs. 2 and 3. In particular, the 'legacy' convex lifting $\tilde{\ell}^{\mathrm{I}}(\mathbf{x})$ was obtained in 250 s (via Algorithm 3), whereas finding the convex lifting $\tilde{\ell}^{\mathrm{II}}(\mathbf{x})$ took mere 3 s (Algorithm 2 and Eq. (15)) which is a negligible post-processing effort compared with the time spent on computing the nominal EMPC controller. The obtained LEMPC controllers were then independently passed through a custom code generation routine to provide an efficient C-code—tailored for the embedded target and integrated into the main code described above.

B. Cross-Platform Comparison

The cross-platform microcontroller comparison tests were performed similarly to the experimental investigation with the active cantilever beam, but in processor only, without the physical beam assembly. The goal of these tests is not to evaluate the vibration damping performance of the controllers as this should be the same in all the cases, but to provide a common foundation for comparing the memory needs and the execution timing of the algorithms. The critical question to ask then is, whether a given class of microcontrollers has enough memory or computation power for similar applications employing the LEMPC methodology proposed here.

Therefore, in these trials, only the C-code for the LEMPC algorithms without state estimation or input–output pre- and



Fig. 7. Microcontrollers used in the cross-platform comparison, size-compared to a €1 coin.

post-processing was implemented on various microcontrollers. Each controller implementation was tested in double and single numeric precision version. The former represents the baseline controller version in finite-precision digital computing, but in certain cases, one may limit memory footprint and computation needs by using only single precision data without significantly affecting the quality of control [52]. This in turn allows usage of better performing controllers on the same hardware or further reduction of hardware cost by using less expensive MCUs. The numeric precision was varied only in the feedback control algorithms, not on the MCU initialization code; however, the chip, clock and peripheral initialization routines are unlikely to contain floating point arithmetic anyway.

In order to make the timing and memory data meaningful for comparison, all controllers were created using the same model and identical parameters. In every single test case, a common and fixed initial state of $\mathbf{x}(0) = [-5\,\text{mm}\ \ 400\,\text{mm}\,\text{s}^{-1}]^T$ was considered. This state simulated an initial condition where the beam is deflected to the position of -5 mm and having the initial speed of $400\,\text{mm}\,\text{s}^{-1}$. This initial state is far outside the terminal set of the controllers and emulates a state for which constraints will be guaranteed to remain active.

Just as in case of the experiments with the active beam, here the MCUs were initialized using the automatically generated code from STM32Cube MX, then it was compiled and deployed by IAR EWARM. In order to minimize the code overhead in addition to the LEMPC algorithms, all the peripherals were disabled in software besides a single digital output pin and the single wire debugging capability of the chip. This means that, though the memory footprint listed here includes a minimal overhead for initializing the MCU, this is kept at an absolute possible minimum. The digital signal from the remaining pin was used to measure TET using an oscilloscope.

The range of MCUs tested in this paper represents a good cross-section of the devices typically available on the market at current time. The microcontrollers considered in this article are photographed and compared in geometric scale in Fig. 7. All MCUs were tested using electronic prototyping boards by STMicroelectronics (Discovery-series boards) for easy physical pin access, integrated programming and debugging features. The MCUs evaluated here range from devices that are considered to be entry level (F051x), extremely cost efficient (F030x), or low power (L100x); up to chips created for high-performance embedded applications (e.g. F407x and F746x). Obviously, the price point of the microcontrollers depends on many factors not directly related to computing performance or memory, such as peripheral selection or purchase quantity. The unit cost of the devices considered here ranges from approximately $1.50 to $10 for large quantities.

The exact types and basic specifications of the MCU considered here will be introduced along with the experimental results in the next section. The non-volatile memory used to store the program on the microcontroller ranged from 64 kB up to 2048 kB, while the volatile data memory ranged from 8 kB up to 340 kB. All microcontrollers were clocked to their respective maximum clock speeds, ranging from 24 MHz up to the considerable 216 MHz, except for STM F303x, which was tested at 64 MHz instead of its 72 MHz maximum, due to a hardware limitation of the prototyping board.

Besides the raw information on clock speed, the real performance of a microcontroller depends on many other factors; including that of the core design. The MCUs employed here incorporate the so-called ARM Cortex M0, M3, M4 and M7 architectures. Generally speaking, the architecture may affect both memory footprint and execution speed of the compiled program, even when using the same exact high-level code and accounting for clock speed differences. Some of the controllers tested here featured a floating point unit (FPU) as well, which may affect the computational performance of predictive controllers in theory—in fact, any task with floating point arithmetic. Although a previous work did not suggest a considerable performance increase with MCUs using their FPU [52], and while any EMPC implementation is an unlikely candidate for FPU-boosted efficiency, we have turned these components on for all tests featured in the paper—if available in the core.

IV. Results and Discussion

A. Active Beam Test

First, let us review the results from laboratory experiments with the active beam. A typical release test is shown in Fig. 8; with output (position), input (voltage) and execution timing depicted from top to bottom. The open-loop response is shown in the background, this is compared to feedback control using

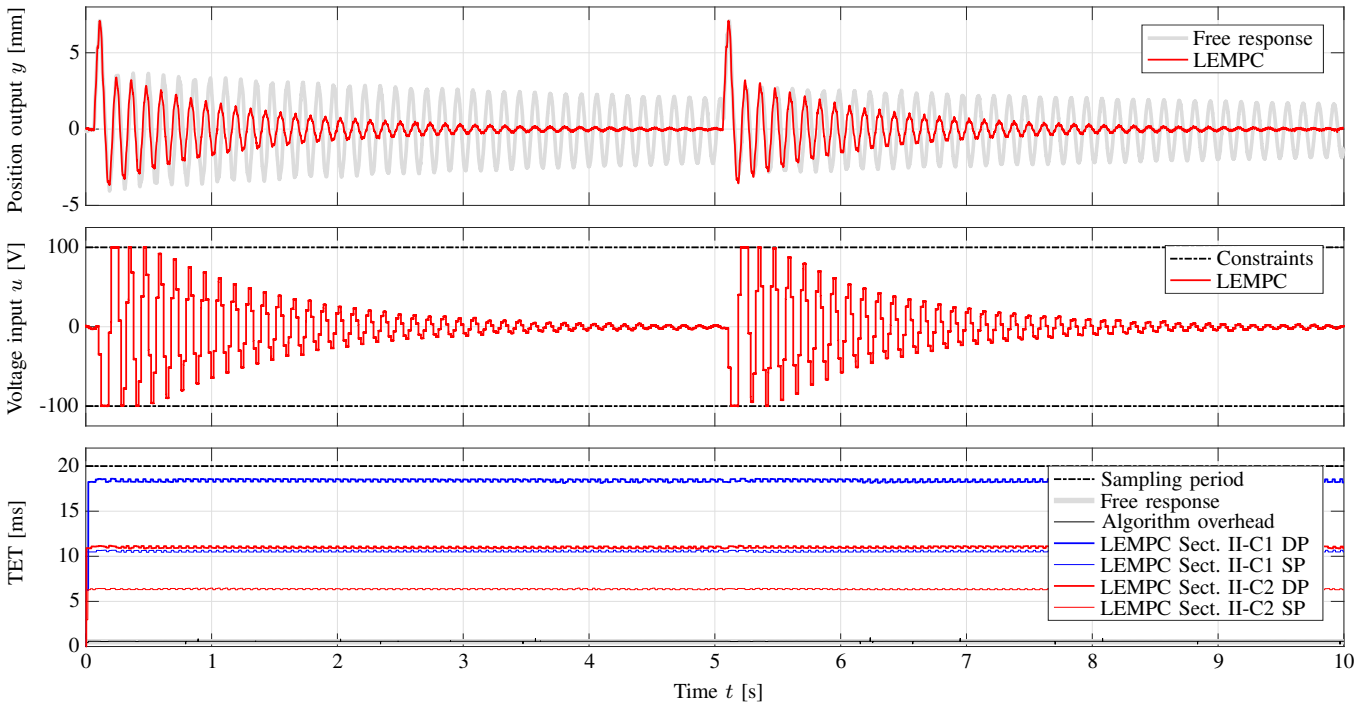


Fig. 8. Typical output, input and timing profiles for the laboratory release tests (two cycles) using the active cantilever beam and the F051x MCU; DP (SP) denotes Double(Single)-Precision floating-point format.

the LEMPC algorithm. As it is expected, the settling time of the transients is reduced considerably (∼20 times) by the active control. Also, although the input constraints are active for a while, they are inherently respected by MPC. None of these facts shall be surprising, as the LEMPC formulation proposed here will create exactly the same outputs as any other optimal control algorithm using dual-mode infinite-horizon predictive control with guaranteed stability. In other words, neither theoretically nor practically is there any difference between the inputs generated by the proposed LEMPC controllers and other known MPC algorithms with the same configuration. This, of course, also means that the vibration damping performance, or more generally the control performance, remains unchanged.

Therefore, the important feature of the LEMPC framework is not an increased control performance but the memory and timing efficiency; which may enable the deployment of fast sampling applications even on simple hardware. The bottom of Fig. 8 shows the timing process for both versions of LEMPC introduced in Sect. II-C in both double and single precision implementations. The chosen baseline hardware executes the controller from Sect. II-C1 with a total TET approaching the 20 ms sampling period, while the controller from Sect. II-C2, central to this work, is executed in less than 11 ms. By employing single-precision floating-point format these figures can be yet reduced nearly twice, without any noticeable deterioration in the damping performance. Note that the overhead (state estimation and data processing) amounts to roughly 1 ms. The timing chart shown here indicates that online execution timing does not depend on state estimates for LEMPC; just as it was suggested in Sect. II-C3. This is unlike the execution time of nominal EMPC employing the sequential search that depends on the location of state within the polyhedral partition as well as the complexity of its regions [26]. The memory footprint of the controller from Sect. II-C1 occupies most of the available non-volatile memory of the F051x MCU; taking up 58 kB of the available 64 kB. Approximately 22 kB of this memory is used by the program itself, while 36 kB by the data—meaning the static variables. The even more efficient controller of Sect. II-C2 requires mere 36 kB of non-volatile memory for the entire feedback control algorithm implementation, including state estimation and data processing. Volatile memory needs for dynamic variables are minimal in all cases, well under 1 kB.

We remark that the nominal EMPC controller has been attempted to be run on the F051x MCU as well. However, the formulation with only a 14-steps long prediction horizon was the maximum that could fit into the available memory of the MCU. Nonetheless, such controller did not yield the feasible set large enough to allow for experimental testing. More details on the embedded implementation of explicit MPC for vibration control, yet without stability guarantees, can be found in our previous study [26].

To appreciate the mere fact that both LEMPC versions were deployable in the fairly limited memory of the F051x MCU and running in real time under 20 ms, one has to recollect the typical hardware requirements for dual-mode infinite-horizon MPC with stability guarantees. It is extremely unlikely that an online QP strategy would be able to solve the same problem within 20 ms using the available 24 MHz clock speed without the possibility of suboptimal solution, although it could possibly fit into the allocated 64 kB ROM. On the other hand, the clock speed and architecture of this low-end MCU could be sufficient to evaluate a nominal EMPC formulation online even with sequential search, but it would be out of the question to fit its memory footprint under 64 kB. To put this in perspective:

TABLE I
CROSS-PLATFORM COMPARISON FEATURING 32-BIT STM MICROCONTROLLERS FOR ACTIVE VIBRATION CONTROL USING THE EFFICIENT REGIONLESS LEMPC IMPLEMENTATION ($T_\mathrm{s} = 20\,\mathrm{ms}$, $N = 50$, $\mathbf{x}(0) = [-5\,\mathrm{mm}\ 400\,\mathrm{mm\,s^{-1}}]^T$) PROPOSED IN SECTION II-C.

Online implementation via Algorithm 4					using $\tilde{\ell}^{\,\mathrm{I}}(\mathbf{x})$, $\tilde{\kappa}^{\,\mathrm{I}}(\mathbf{x})$ obtained per Section II-C1												using $\tilde{\ell}^{\,\mathrm{II}}(\mathbf{x})$, $\tilde{\kappa}^{\,\mathrm{II}}(\mathbf{x})$ obtained per Section II-C2											
Precision					double						single						double						single					
Metrics†					ROM			RAM	TET		ROM			RAM	TET		ROM			RAM	TET		ROM			RAM	TET	
MCU type	ARM Cortex architecture	Clock frequency [MHz]	Available ROM [kB]	Available RAM [kB]	CODE [B]	DATA [B]	Occupied ROM [kB]	Occupied RAM [B]	TET [ms]	Norm. TET [µs/DMIPS]	CODE [B]	DATA [B]	Occupied ROM [kB]	Occupied RAM [B]	TET [ms]	Norm. TET [µs/DMIPS]	CODE [B]	DATA [B]	Occupied ROM [kB]	Occupied RAM [B]	TET [ms]	Norm. TET [µs/DMIPS]	CODE [B]	DATA [B]	Occupied ROM [kB]	Occupied RAM [B]	TET [ms]	Norm. TET [µs/DMIPS]
STM32F051R8T6‡	M0	48	64	8	22 836	37 180	58.6	552	18.43	485	13 124	18 604	31.0	536	9.99	263	15 140	21 788	36.1	552	10.75	283	9280	10 908	19.7	536	5.84	154
STM32F030R8T6	M0	48	64	8	22 820	37 180	58.6	552	18.48	458	13 108	18 604	31.0	536	10.02	249	15 124	21 788	36.0	552	10.79	268	9264	10 908	19.7	536	5.86	145
STM32F100RB	M3	24	128	8	23 014	37 182	58.8	552	19.31	644	13 198	18 606	31.1	536	10.81	360	15 318	21 790	36.2	552	11.28	376	9354	10 910	19.8	536	6.35	212
STM32L100RCT6	M3	32	256	16	23 594	37 182	59.4	552	17.64	441	13 778	18 606	31.6	536	10.68	267	15 898	21 790	36.8	552	10.25	256	9934	10 910	20.4	536	6.54	163
STM32L152RCT6	M3	32	256	32	23 590	37 182	59.3	552	16.07	402	13 774	18 606	31.6	536	9.87	247	15 894	21 790	36.8	552	9.38	235	9930	10 910	20.4	536	5.83	146
STM32F303VCT6	M4	64	256	40	14 100	46 464	59.1	552	9.95	124	8264	23 248	30.8	536	1.63	20	10 252	27 224	36.6	552	5.80	73	11 770	8197	19.5	536	0.95	12
STM32F401VCT6	M4	84	256	64	13 832	46 464	58.9	552	5.16	49	7996	23 248	30.5	536	0.67	6	9984	27 224	36.3	552	3.03	29	11 502	8198	19.2	536	0.40	4
STM32F407VGT	M4	168	1024	192	14 084	46 464	59.1	552	2.63	13	8248	23 248	30.8	536	0.35	2	10 236	27 224	36.6	552	1.54	7	11 754	8198	19.5	536	0.21	1
STM32F429ZI	M4	180	2048	256	14 204	46 464	59.2	552	2.41	11	8368	23 248	30.9	536	0.32	1	10 356	27 224	36.7	552	1.42	6	11 874	8198	19.6	536	0.19	1
STM32F746NGH6	M7	216	1024	340	14 504	46 464	59.5	552	5.52	42	8668	23 248	31.2	536	1.03	2	10 656	27 224	37.0	552	2.99	6	12 174	8198	19.9	536	0.60	1

† ROM (non-volatile, read-only memory) = CODE (program) + DATA (static variables); RAM – volatile, random-access memory; TET – task execution time (MPC only); Normalized TET = TET / (clock × DMIPS / MHz), DMIPS – Dhrystone MIPS (millions of instructions per second).
‡ The embedded target used in the experiments in Fig. 8.

an equivalent MPC formulation with guaranteed stability and recursive feasibility would be near-impossible to deploy to the same hardware by other currently known means.

B. Cross-Platform Comparison

Continuing in introducing and evaluating the results in this spirit, in the upcoming paragraphs we will only focus on the timing and memory footprint of the two LEMPC algorithms in single and double numeric precision formats. Results of the STM cross-platform comparison are summarized in Table I. The left side of the table lists the type designation of the microcontrollers, their architecture family, clock speed, and the maximum available non-volatile ROM for the program and data, and volatile RAM for variable manipulation. Note also that the maximum non-volatile memory refers to that available on the chip itself; except F429 which also lists the external memory of the prototyping board.

Table I is further divided into two variants of the LEMPC algorithm, each of which was tested in double and single numeric precision—creating a total of four test scenarios. Each of these lists the non-volatile ROM footprint for the complete algorithm including the overhead, and its program (CODE) and static data (DATA) variables as reported by the compiler. The RAM contains dynamic variables, though this footprint is minimal for the proposed controllers and keeps the same size between hardware versions.

Timing is characterized by the task execution time as measured for the single initial state $\mathbf{x}(0)$ without overhead, however, this timing metric is indicative for any other state as well. The absolute execution timing is, of course, the function of the computational performance of the microcontroller. The final piece of information presented in Table I is the normalized TET, which attempts to level the differences in the small variation in architecture between various microcontrollers shown here. The Dhrystone million instructions per second (DMIPS) metric indicates the relative processor efficiency, but does not include floating point operations. The DMIPS metric is readily available for the processors tested here, and accounts for some of the differences in architecture. A typical 8-bit microcontroller has 1.0 DMIPS for each MHz of its clock speed; essentially performing a single elementary operation for each clock cycle. The relationship is more complicated in modern 32-bit MCUs, as the devices tested here range between 0.8–2.1 DMIPS/MHz. Rescaling this to the actual clock speed, and then dividing the TET by it yields the time necessary to evaluate the LEMPC algorithm for one DMIPS, which we chose to call normalized TET in Table I.

Let us first look at the differences and the similarities for a given experimental configuration, that is, for a given LEMPC variant and numeric precision. Upon inspecting the table, one may see that the volatile memory remains virtually unchanged across platforms for the same type of controller. This is also true for different variants, as the RAM footprint remains well under 1 kB for all cases. This size is not prohibitive even for the most fundamental types of modern MCUs. The non-volatile memory footprint varies only slightly for a given test case; this can be caused by minor differences in architecture or even the size of overhead that must include the initialization code for the given MCU. Timing varies greatly amongst the tested platforms for a given test case, as the timing depends on clock speed an architecture.

On average, switching to the single precision yields an $\sim$1.9 times smaller footprint in ROM for both LEMPC implementations. Changing the formulation from the one in Sect. II-C1 to the one in Sect. II-C2 will further reduce the non-volatile memory footprint by a factor of $\sim$1.6 for both numerical precision cases. To approximate the effect of numerical precision on computational effort, note that TET will be reduced anywhere from a factor of $\sim$1.5–7.7 (averaging around 4) which depends on architecture and clock speed strongly. This suggests that limiting the numeric precision—or even converting to fixed point—can broaden the horizons for applying predictive control on cheap hardware, if the reduced precision does not affect algorithm convergence or control quality. We remark

that the aspect of possible performance loss due to the lower precision may be further investigated from a theoretical point of view via a fragility analysis [53], based on the disturbances of the explicit PWA control law.

Yet again, the relative differences between the two proposed LEMPC techniques and the numerical precision is unimportant compared to the fact that the traditional implementation of EMPC or most implicit quadratic programming solvers could not have solved the same problem on the majority of investigated MCUs; either due to their large memory needs, or computational requirements. We may consider the memory footprint and online evaluation effort of the proposed controllers from two viewpoints. By using efficient MPC algorithms, one can either opt for much cheaper control hardware for the same application; or use more complex models or higher sampling rates than before. Table I illustrates both of these critical aspects. By implementing the proposed algorithms, cheap entry-level MCU like the F100x can compute the same optimal constrained MPC moves with guaranteed stability and even long horizons that would otherwise need powerful hardware [25]. Conversely, high-end microcontrollers like the F407x can execute the benchmark vibration control problem in the microsecond range, suggesting that even faster applications or better mathematical models are within the realm of possibilities.

V. Conclusion

This paper presented a methodology to synthesize memory and time-efficient MPC solutions, herein utilized to suppress lateral vibrations of an active cantilever beam. The key feature of the proposed convex lifting based implementation is that the online algorithm is extremely simple with only a tiny footprint, which makes it deployable even on low-end embedded control hardware. In addition, the optimal explicit solution to a given MPC problem can be evaluated at runtime with the pre-defined online computation time guarantees, allowing for an easy verification of correctness of implementation in rapid prototyping design and real-world applications. The proposed framework was experimentally tested to provide optimal vibration control performance, accompanied by a comprehensive cross-platform comparison study using the family of 32-bit microcontrollers; suggesting a step towards feasible embedded implementations in memory- and time-critical optimal control applications.

The proposed framework will likely allow the use of more complex structural vibration models that cover further resonant frequencies and modes, possibly including rotational degrees of freedom. Further research shall explore this possibility and also discover theoretical and implementation challenges posed by the need of more complex observer algorithms.

Acknowledgment

The authors would like to thank the associate editor and the anonymous reviewers for their helpful comments.

Appendix

Algorithm 5 Pseudocode implementation of Algorithm 4

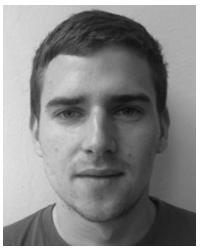
Stored data: Convex lifting and control input gains $\tilde{\mathbf{a}}_j$, $\tilde{b}_j$, $\tilde{\mathbf{f}}_j$, $\tilde{g}_j$, $\forall j \in \mathcal{I}_{\tilde{R}^{\mathrm{II}}}$, input bounds $\overline{\kappa}$, $\underline{\kappa}$ if any.
Input: State $\mathbf{x}_0 \in \mathcal{R}$.
Output: Optimal control input $u_0^\star$.

1: **function** LEMPCCONTROLLER($\mathbf{x}_0$)
2: $objmax \leftarrow 0$
3: **for** $j \leftarrow 1$ to $\tilde{R}^{\mathrm{II}}$ **do** ▷ $\tilde{R}^{\mathrm{II}} = |\mathcal{I}_{\mathrm{uns}}|$
4: $obj \leftarrow \tilde{\mathbf{a}}_j^T \mathbf{x}_0 + \tilde{b}_j$
5: **if** $obj > objmax$ **then**
6: $obj \leftarrow objmax$
7: $j^\star \leftarrow j$
8: **end if**
9: **end for**
10: $ctrl \leftarrow \tilde{\mathbf{f}}_{j^\star}^T \mathbf{x}_0 + \tilde{g}_{j^\star}$
▷ lines 11–15, and 17 are not needed if $\mathcal{I}_{\mathrm{sat}} = \emptyset$
11: **if** $ctrl > \overline{\kappa}$ **then**
12: $u_0^\star \leftarrow \overline{\kappa}$
13: **else if** $ctrl < \underline{\kappa}$ **then**
14: $u_0^\star \leftarrow \underline{\kappa}$
15: **else**
16: $u_0^\star \leftarrow ctrl$
17: **end if**
18: **return** $u_0^\star$
19: **end function**

References

[1] S. Qin and T. Badgwell, "A survey of industrial model predictive control technology," *Control Eng. Pract.*, vol. 11, no. 7, pp. 733–764, 2003.
[2] D. Niederberger, *Hybrid systems: computation and control*. Springer Berlin Heidelberg, 2005, vol. 3414, ch. Design of optimal autonomous switching circuits to suppress mechanical vibration, pp. 511–525.
[3] A. G. Wills, D. Bates, A. J. Fleming, B. Ninness, and S. O. R. Moheimani, "Model predictive control applied to constraint handling in active noise and vibration control," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 1, pp. 3–12, 2008.
[4] A. Wills, A. Mills, and B. Ninness, "FPGA implementation of an interior-point solution for linear model predictive control," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 14 527–14 532, 2011, 18th IFAC World Congress.
[5] G. Takács and B. Rohal'-Ilkiv, *Model Predictive Vibration Control: Efficient Constrained MPC Vibration Control for Lightly Damped Mechanical Structures*. Springer London, 2012.
[6] C. Edwards, "The 8bit strikes back," *Electron. Syst. Software*, vol. 5, no. 2, pp. 36–39, 2007.
[7] P. Busono, A. Iswahyudi, M. A. A. Rahman, and A. Fitrianto, "Design of embedded microcontroller for controlling and monitoring blood pump," *Procedia Comput. Sci.*, vol. 72, pp. 217–224, 2015.
[8] D. K. M. Kufoalor, V. Aaker, T. A. Johansen, L. Imsland, and G. O. Eikrem, "Automatically generated embedded model predictive control: moving an industrial PC-based MPC to an embedded platform," *Optim. Contr. Appl. Methods*, vol. 36, no. 5, pp. 705–727, 2015.
[9] P. Dua, K. Kouramas, V. Dua, and E. Pistikopoulos, "MPC on a chip—recent advances on the application of multi-parametric model-based control," *Comput. Chem. Eng.*, vol. 32, no. 4-5, pp. 754–765, 2008.
[10] B. Huyck, J. D. Brabanter, B. D. Moor, J. F. V. Impe, and F. Logist, "Online model predictive control of industrial processes using low level control hardware: A pilot-scale distillation column case study," *Control Eng. Pract.*, vol. 28, pp. 34–48, 2014.
[11] D. K. M. Kufoalor, S. Richter, L. Imsland, T. A. Johansen, M. Morari, and G. O. Eikrem, "Embedded model predictive control on a PLC using a primal-dual first-order method for a subsea separation process," in *Proc. 22nd Mediterr. Conf. Control and Autom.*, June 2014, pp. 368–373.

[12] K. Ling, B. Wu, and J. Maciejowski, "Embedded model predictive control MPC using a FPGA," *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 15 250–15 255, 2008, 17th IFAC World Congress.
[13] P. D. Vouzis, L. G. Bleris, M. G. Arnold, and M. V. Kothare, "A system-on-a-chip implementation for embedded real-time model predictive control," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1006–1017, 2009.
[14] D. Q. Mayne, "Model predictive control: recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.
[15] S. Richter, C. Jones, and M. Morari, "Real-time input-constrained MPC using fast gradient methods," in *Proc. 48th IEEE Conf. Decision and Control and the 28th Chin. Control Conf.*, 2009, pp. 7387–7393.
[16] P. Zometa, M. Kögel, T. Faulwasser, and R. Findeisen, "Implementation aspects of model predictive control for embedded systems," in *Proc. Amer. Control Conf.*, 2012, pp. 1205–1210.
[17] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3238–3251, 2014.
[18] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.
[19] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear Model Predictive Control: Towards New Challenging Appl.*, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Springer Berlin Heidelberg, 2009, pp. 345–369.
[20] F. Borrelli, *Constrained Optimal Control of Linear and Hybrid Systems*. Springer-Verlag Berlin Heidelberg, 2003, vol. 290.
[21] M. Kvasnica, *Real-Time Model Predictive Control via Multi-Parametric Programming: Theory and Tools*. VDM Verlag Saarbrücken, 2009.
[22] A. Pneumont, *Vibration Control of Active Structures: An Introduction*, 3rd ed. Springer Netherlands, 2011.
[23] "Liftware," https://www.liftware.com/, accessed: 2016-10-29.
[24] R. Oberdieck, N. A. Diangelakis, I. Nascu, M. M. Papathanasiou, M. Sun, S. Avraamidou, and E. N. Pistikopoulos, "On multi-parametric programming and its applications in process systems engineering," *Chem. Eng. Res. Des*, 2016.
[25] G. Takács and B. Rohal'-Ilkiv, "Model predictive control algorithms for active vibration control: a study on timing, performance and implementation properties," *J. Vib. Control*, vol. 20, no. 13, pp. 2061–2080, 2014.
[26] G. Takács, G. Batista, M. Gulan, and B. Rohal'-Ilkiv, "Embedded explicit model predictive vibration control," *Mechatronics*, vol. 36, pp. 54–62, 2016.
[27] N. A. Nguyen, S. Olaru, P. Rodríguez-Ayerbe, and M. Kvasnica, "Convex liftings-based robust control design," *Automatica*, vol. 77, pp. 206–213, 2017.
[28] N. A. Nguyen, S. Olaru, P. Rodríguez-Ayerbe, M. Hovd, and I. Necoara, "Constructive solution of inverse parametric linear/quadratic programming problems," *J. Optim. Theory. Appl.*, pp. 1–26, 2016.
[29] M. Gulan, N. A. Nguyen, S. Olaru, P. Rodríguez-Ayerbe, and B. Rohal'-Ilkiv, "Implications of inverse parametric optimization in model predictive control," in *Develop. in Model-Based Optim. and Control: Distrib. Control and Ind. Appl.*, S. Olaru, A. Grancharova, and F. L. Pereira, Eds. Springer Int. Publishing, 2015, pp. 49–70.
[30] E. C. Kerrigan, "Robust constraint satisfaction: invariant sets and predictive control," Ph.D. dissertation, University of Cambridge, UK, 2000.
[31] J. Richelot, J. Bordeneuve-Guibe, and V. Pommier-Budinger, "Active control of a clamped beam equipped with piezoelectric actuator and sensor using generalized predictive control," in *Proc. 23rd IEEE Int. Symp. Ind. Electron.*, vol. 1, 2004, pp. 583–588.
[32] G. Ferrari and M. Amabili, "Active vibration control of a sandwich plate by non-collocated positive position feedback," *J. Sound Vib.*, vol. 342, pp. 44–56, 2015.
[33] D. J. Inman, *Engineering Vibrations*, 4th ed. Pearson, 2014.
[34] R. Y. Chiang and M. G. Safonov, "Design of $\mathcal{H}_\infty$ controller for a lightly damped system using a bilinear pole shifting transform," in *Proc. Amer. Control Conf.*, 1991, pp. 1927–1928.
[35] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*. Birkhäuser Boston, 2008.
[36] M. Kvasnica, J. Hledík, I. Rauová, and M. Fikar, "Complexity reduction of explicit model predictive control via separation," *Automatica*, vol. 49, no. 6, pp. 1776–1781, 2013.
[37] F. Bayat, T. A. Johansen, and A. A. Jalali, "Flexible piecewise function evaluation methods based on truncated binary search trees and lattice representation in explicit MPC," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 3, pp. 632–640, 2012.
[38] N. A. Nguyen, M. Gulan, S. Olaru, and P. Rodríguez-Ayerbe, "Convex liftings: theory and control applications," *IEEE Trans. Autom. Control*, 2017.
[39] J. Spjøtvold, E. C. Kerrigan, C. N. Jones, P. Tøndel, and T. A. Johansen, "On the facet-to-facet property of solutions to convex parametric quadratic programs," *Automatica*, vol. 42, no. 12, pp. 2209–2214, 2006.
[40] N. A. Nguyen, "Explicit robust constrained control for linear systems: analysis, implementation and design based on optimization," Ph.D. dissertation, CentraleSupélec, Université Paris-Saclay, France, 2015.
[41] M. Kvasnica and M. Fikar, "Clipping-based complexity reduction in explicit MPC," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1878–1883, 2012.
[42] N. A. Nguyen, S. Olaru, P. Rodríguez-Ayerbe, M. Hovd, and I. Necoara, "Inverse parametric convex programming problems via convex liftings," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 2489–2494, 2014, 19th IFAC World Congress.
[43] ——, "Fully inverse parametric linear/quadratic programming problems via convex liftings," in *Develop. in Model-Based Optim. and Control: Distrib. Control and Ind. Appl.*, S. Olaru, A. Grancharova, and F. L. Pereira, Eds. Springer Int. Publishing, 2015, pp. 27–47.
[44] M. Baotić, F. Borrelli, A. Bemporad, and M. Morari, "Efficient on-line computation of constrained optimal control," *SIAM J. Control Optim.*, vol. 47, no. 5, pp. 2470–2489, 2008.
[45] F. Bayat, T. A. Johansen, and A. A. Jalali, "Using hash tables to manage the time-storage complexity in a point location problem: application to explicit model predictive control," *Automatica*, vol. 47, no. 3, pp. 571–577, 2011.
[46] M. Kvasnica, B. Takács, J. Holaza, and S. D. Cairano, "On region-free explicit model predictive control," in *Proc. 54th Conf. Decision and Control*, Osaka, Japan, 2015, pp. 3669–3674.
[47] A. Gupta, S. Bhartiya, and P. Nataraj, "A novel approach to multiparametric quadratic programming," *Automatica*, vol. 47, no. 9, pp. 2112–2117, 2011.
[48] F. Borrelli, M. Baotić, J. Pekar, and G. Stewart, "On the computation of linear model predictive control laws," *Automatica*, vol. 46, no. 6, pp. 1035–1041, 2010.
[49] A. Airan, M. Bhushan, and S. Bhartiya, "Linear machine solution to point location problem," *IEEE Trans. Autom. Control*, vol. 62, no. 3, pp. 1403–1410, 2017.
[50] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *Proc. 12th Eur. Control Conf.*, Zürich, Switzerland, 2013, pp. 502–510.
[51] J. Löfberg, "YALMIP: a toolbox for modeling and optimization in MATLAB," in *IEEE Int. Symp. Comput. Aided Control Syst. Design*, 2004, pp. 284–289.
[52] G. Takács, P. Zometa, R. Findeisen, and B. Rohal'-Ilkiv, "Efficiency and performance of embedded model predictive control for active vibration attenuation," in *Proc. 15th Eur. Control Conf.*, Aalborg, Denmark, 2016, pp. 1334–1340.
[53] N. A. Nguyen, S. Olaru, P. Rodríguez-Ayerbe, G. Bitsoris, and M. Hovd, "Explicit robustness and fragility margins for linear discrete systems with piecewise affine control law," *Automatica*, vol. 68, pp. 334–343, 2016.



Martin Gulan received the M.Sc. degree in applied mechanics and the Ph.D. degree in mechatronics from the Slovak University of Technology in Bratislava, Bratislava, Slovakia, in 2012 and 2015, respectively.

He was a Research Scholar with the Advanced Micro and Nanosystems Laboratory, Department of Aerospace and Mechanical Engineering, University of Arizona, Tucson, AZ, USA, in 2011, and a Visiting Researcher with the Laboratory of Signals and Systems, CentraleSupélec, Gif-sur-Yvette, France, in 2014. Since 2015, he has been a Post-Doctoral Researcher with the Slovak University of Technology in Bratislava, and serves as a Research Assistant at the Institute of Automation, Measurement and Applied Informatics, Faculty of Mechanical Engineering at the same university. His research interests are mainly in computation and fast implementation of model predictive control and real-time embedded optimization.



Gergely Takács (M'11) received both the M.Sc. and Ph.D. degrees in mechatronics from the Slovak University of Technology in Bratislava, Bratislava, Slovakia, in 2006 and 2009, respectively.

From 2009 to 2014 he was a Research Assistant with the Institute of Automation, Measurement and Applied Informatics, Faculty of Mechanical Engineering, Slovak University of Technology in Bratislava. Since 2015, he has been an Associate Professor with the same university. His research interests include optimal control and estimation, active vibration control and embedded systems.



Ngoc Anh Nguyen received a double M.Sc. degree in electrical engineering from the Hanoi University of Science and Technology, Vietnam, and from the Grenoble Institute of Technology, France, in 2012. In 2015 he received his Ph.D. degree at the Laboratory of Signals and Systems, CentraleSupélec, University Paris-Saclay, France.

Since 2016, he has been a Post-Doctoral Scholar with the Johannes Kepler University, Linz, Austria. His main research interests lie in optimization-based control and set-theoretic methods.



Sorin Olaru (M'1–SM'12) received the M.Sc. degree in electrical engineering from the University Politehnica of Bucharest, Bucharest, Romania, in 2002, and the Ph.D. degree in automatic control from the University Paris-Sud, Orsay, France, in 2005, and from the University Politehnica of Bucharest, in 2010. In 2011 he received the HDR degree from the University Paris-Sud.

Since 2012, he has been a Professor with CentraleSupélec, and a member of the CNRS Laboratory of Signals and Systems and of the INRIA team DISCO —all these institutions being part of the Paris-Saclay University, France. His research interests encompass optimization-based control design, set-theoretic characterization of constrained dynamical systems as well as numerical methods in control. He is currently involved in research projects related to embedded predictive control, fault tolerant control and time-delay systems.



Pedro Rodríguez-Ayerbe (M'10) received the M.Sc. degree in electrical engineering from École Supérieure d'Électricité (Supélec), Gif-sur-Yvette, France, in 1996. He then received the Ph.D. degree in automatic control and the HDR degree from Supélec and the University Paris-Sud, Orsay, France, in 2002 and 2014, respectively.

Since 2016, he has been a Professor with CentraleSupélec, and a member of the CNRS Laboratory of Signals and Systems, both being part of the Paris-Saclay University, France. From 1996 until 2000 he worked in Fagor Automation, Spain, and in Lore, France. His research interests include constrained predictive control and robust control theory.



Boris Rohaľ-Ilkiv (M'11) received both the M.Sc. degree and the Ph.D. degree in technical cybernetics from the Slovak University of Technology in Bratislava, in 1972 and 1978, respectively.

Since 2003 he has been a Professor in mechatronics with the Institute of Automation, Measurement and Applied Informatics, Faculty of Mechanical Engineering, Slovak University of Technology in Bratislava. His main research interests lie in optimization-based estimation and control of industrial processes and mechatronic systems, including combustion engines, underactuated mechanical systems and active vibration systems.