2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

# Explicit MPC based on Approximate Dynamic Programming

Peter Bakaráč, Juraj Holaza, Martin Kalúz, Martin Klaučo, Johan Löfberg and Michal Kvasnica

*Abstract*— **In this paper we show how to synthesize simple explicit MPC controllers based on approximate dynamic programming. Here, a given MPC optimization problem over a finite horizon is solved iteratively as a series of problems of size one. The optimal cost function of each subproblem is approximated by a quadratic function that serves as a cost-to-go function for the subsequent iteration. The approximation is designed in such a way that closed-loop stability and recursive feasibility is maintained. Specifically, we show how to employ sum-of-squares relaxations to enforce that the approximate cost-to-go function is bounded from below and from above for all points of its domain. By resorting to quadratic approximations, the complexity of the resulting explicit MPC controller is considerably reduced both in terms of memory as well as the on-line computations. The procedure is applied to control an inverted pendulum and experimental data are presented to demonstrate viability of such an approach.**

## I. Introduction

In recent years, safety, economic and environmental aspects play a crucial role in the controller design. For this reason, Model Predictive Control (MPC) has become a widespread control policy mainly due to its natural capability of synthesizing feedback controllers for large systems, while embedding all of the system's constraints directly to the optimization problem [1], [2]. The main drawback of this control strategy, however, lies in the computational burden associated with solving a given optimization problem at each sampling instant.

One way of tackling this issue is to resort to *explicit MPC*, introduced in [3]. Here, the underlying idea is to pre-calculate the optimal solution for all feasible initial conditions via parametric programming [4], [5]. For the type of problems tackled in this paper (i.e., for linear systems subject to linear constraints, and with quadratic performance objective to be minimized), the explicit MPC feedback law takes a form of a piecewise affine (PWA) function defined over polyhedral regions [6], [7]. The advantage of such explicit solutions is that the on-line calculation of the optimal control inputs reduces to a mere function evaluation that can be performed with low implementation effort even on simple control hardware, such as on FPGAs, PLCs, or microcontrollers. However, the main drawback is that the explicit MPC feedback law is often too complex to be stored in the memory of the control hardware because the number of polyhedral regions grows, in the worst case, exponentially with the prediction horizon.

P. Bakráč, J. Holaza, M. Kalúz, M. Klaučo and M. Kvasnica are with the Slovak University of Technology in Bratislava, Slovakia, {peter.bakarac, juraj.holaza, martin.kaluz, martin.klauco,michal.kvasnica}@stuba.sk. J Löfberg is with the Linköping University, johanl@isy.liu.se.

Various approaches to reduction of the memory footprint of explicit MPC solutions have been proposed in the literature. Generally, two categories of methods are considered. In the first one, the explicit controller is replaced by its simpler counterpart while preserving optimality. This can be achieved, e.g., by discarding the polytopic partition and exploiting the convex nature of the value function in the point location problem [7], by encoding regions by primal and dual optimizers [8], [9], by merging regions that share the same feedback law [10], or by compressing the data using universal numbers [11]. The second category of methods replaces the optimal (but complex) feedback law by a function of lower complexity, e.g. by polynomials [12], [13], barycentric functions [14], or by subdivision of regions into hypercubes [15].

In this paper we propose a novel procedure of obtaining explicit MPC feedback laws of small complexity, measured by the number of polyhedral regions over which the feedback law is computed. Specifically, we propose to solve the MPC problem backwards in time using dynamic programming (DP), as suggested in [16]. There, it is shown that the explicit MPC feedback takes a form of piecewise affine function if the MPC objective function consists of weighted 1 and infinity norms. For quadratic performance indices (which are more common in practice), however, the application of DP is not straightforward as it involves solving, parametrically, problems with piecewise quadratic (PWQ) cost-to-go functions. Such cases can be solved either by introducing binary variables, or by solving multiple parametric quadratic programs (one for each region of the PWQ cost-to-go function). Both approaches, however, are ill-suited for practical cases due to their computational burden. An alternative was presented in [17] where the PWQ nature of the cost-to-go function is tackled by solving parametric optimization problems with a larger number of parameters, increasing both the on-line computational load as well as the complexity of the resulting solution. In this paper, such complications are avoided by applying *approximate* dynamic programming. Here, at each iteration of dynamic programming the PWQ cost-to-go function is approximated by a convex quadratic function, allowing each DP iteration to be solved as a single parametric quadratic program with just the states of the system as parameters, thus yielding a PWA feedback law. The approximation is done in such a way that the resulting feedback law provides guarantees of closed-loop stability and satisfaction of hard constraints on states and inputs.

The contributions of the paper are as follows: 1) we formally prove that approximate dynamic programming yields a stabilizing feedback law *by construction*; 2) we show that

978-3-9524-2699-9 

the approximate feedback law is less complex than its optimal counterpart; 3) we provide a computationally tractable approach of finding, in each DP iteration, the approximate cost-to-go function via convex semi-definite programming; 4) we show that the proposed scheme can also be applied to reduce computational complexity of conventional on-line MPC; and 5) we illustrate the complexity reduction and control performance by means of experimental results obtained by controlling an inverted pendulum on a cart.

## II. Preliminaries and Problem Statement

In this paper we aim to control a discrete-time linear time-invariant system that is represented in the state-space domain by

$$x(t+T_{\mathrm{s}}) = Ax(t) + Bu(t), \tag{1}$$

with the state vector $x(t) \in R^{n_{\mathrm{x}}}$, input vector $u(t) \in \mathbb{R}^{n_{\mathrm{u}}}$, sampling time $T_{\mathrm{s}}$, and $(A, B)$ controllable. Moreover, we consider that the system (1) is restricted by

$$x(t) \in \mathcal{X},\ u(t) \in \mathcal{U}, \tag{2}$$

where $\mathcal{X}$ and $\mathcal{U}$ are non-empty sets that contain the origin in their respective interiors.

To design an MPC feedback law $u^\star = \mu(x)$, $\mu : \mathbb{R}^{n_{\mathrm{x}}} \to \mathbb{R}^{n_{\mathrm{u}}}$ which drives states of the system (1) to the origin, while providing asymptotic closed-loop stability and recursive satisfaction of input and state constraints in (2), one can consider constrained finite-time optimal control problem (CFTOC) of the form

$$\begin{aligned} J^\star(x_0) = \min_{u_0,\dots,u_{N-1}}\ & \ell_N(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) && \text{(3a)}\\ \text{s.t. } & x_{k+1} = Ax_k + Bu_k, && \text{(3b)}\\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U}, && \text{(3c)}\\ & x_N \in \mathcal{X}_{\mathrm{N}}, && \text{(3d)} \end{aligned}$$

with constraints (3b)−(3c) enforced for all $k = 0, 1, \dots, N-1$. Here, $x_k$ and $u_k$ are, respectively, the predictions of states and inputs at the $k$-th step of the prediction horizon $N$, initialized from $x_0 = x(t)$. Moreover,

$$\ell(x_k, u_k) = x_k^{\mathsf{T}} Q_{\mathrm{x}} x_k + u_k^{\mathsf{T}} Q_{\mathrm{u}} u_k \tag{4}$$

is the stage cost,

$$\ell_N(x_N) = x_k^{\mathsf{T}} Q_{\mathrm{N}} x_k \tag{5}$$

is the terminal penalty, and $\mathcal{X}_{\mathrm{N}}$ denotes the terminal set. We make the following standard assumptions:

*Assumption 2.1:* The constraint sets $\mathcal{X}$, $\mathcal{U}$, $\mathcal{X}_N$ are polyhedra, and $Q_N = Q_N^{\mathsf{T}} \succeq 0$, $Q_{\mathrm{x}} = Q_{\mathrm{x}}^{\mathsf{T}} \succeq 0$, $Q_{\mathrm{u}} = Q_{\mathrm{u}}^{\mathsf{T}} \succ 0$ are symmetric positive (semi) definite weighting matrices. □

*Assumption 2.2:* The terminal penalty function $\ell_N$ is a Lyapunov function for system (1) subject to (2), i,e., $\ell_N(x_N) \geq 0$ for all $x_N$, $\ell_N(x_N) = 0$ for $x_N = 0$, and $\ell_N(x_{N+1}) - \ell_N(x_N) \leq -\alpha(x_N)$ for some convex function $\alpha$. Moreover, the terminal set $\mathcal{X}_N \subseteq \mathcal{X}$ is positively invariant with respect to some terminal feedback law $\kappa_N$, i.e., $\forall x \in \mathcal{X}_N$ we have $(Ax + B\kappa_N(x)) \in \mathcal{X}_N$ and $\kappa_N(x) \in \mathcal{U}$. □

Under Assumption 2.1, the problem in (3) is a *parametric quadratic program* (pQP) with a strictly convex quadratic objective function and linear constraints. Under Assumption 2.2, the receding horizon control action $u_0^\star$ provides assymptotic closed-loop stability of system (1) subject to persistent satisfaction of the constraints in (2). We remark that the conditions of Assumption 2.2 can be achieved for instance by taking $Q_{\mathrm{N}}$ in (5) as the solution to the discrete-time algebraic Riccati equation, $\kappa_N(x) = Kx$ as the associated LQR state-feedback controller, and computing $\mathcal{X}_N$ as the invariant subset of $\mathcal{X}$ subject to $u = \kappa_N(x)$.

*Lemma 2.3 ([18], [19]):* The minimizer $u_0^\star$ to (3) as a function of the initial condition $x_0$, i.e., $u_0^\star = \mu_0^\star(x_0)$, is a continuous piecewise affine (PWA) function of $x_0$, i.e.,

$$\mu_0^\star(x) := F_i x + f_i \text{ if } x \in \mathcal{R}_i, \tag{6}$$

with $i = 1, \dots, M$, $M$ denoting the number of polyhedral regions

$$\mathcal{R}_i = \{x \mid \Phi_i x \leq \phi_i\}, \tag{7}$$

where $\Phi_i$, $\phi_i$, $F_i$, and $f_i$ are computed by solving (3) using parametric programming. Moreover, the feasible set of (3), i.e.,

$$\mathcal{X}_0 = \{x_0 \mid \exists u_0, \dots, u_{N-1} \text{ s.t. (3b)} - \text{(3d) holds}\}, \tag{8}$$

is a polyhedron and $\mathcal{X}_0 = \cup_i \mathcal{R}_i$. Finally, the minimum of (3a) subject to (3b)−(3d), i.e., the optimal value function $J^\star(x_0)$, is a convex continuous piecewise quadratic (PWQ) function

$$J^\star(x) := x^{\mathsf{T}} H_i x + E_i^{\mathsf{T}} x + h_i \text{ if } x \in \mathcal{R}_i, \tag{9}$$

defined over the same set of polyhedral regions. □

*Remark 2.4:* The on-line implementation of the PWA feedback law (6), i.e., the process of obtaining $u_0^\star = \mu_0^\star(x(t))$ for some $x(t)$, can be done is several ways. The simplest approach is to use sequential search. Here, we start with $i = 1$ and check if $\Phi_i x(t) \leq \phi_i$. If all inequalities are satisfied, $i^\star = i$ and $u_0^\star = F_{i^\star} x(t) + f_{i^\star}$ is the optimal control action for $x(t)$. Otherwise we set $i = i + 1$ and repeat. The procedure terminates, in the worst case, after checking all regions, i.e., for $i = 1, \dots, M$. □

In this paper we aim at synthesizing a PWA feedback law $\widetilde{\mu}_0$ that is simpler than $\mu_0^\star$, trades lower complexity for suboptimality, but still guarantees closed-loop stability. Formally:

*Problem 2.5:* We aim at computing a suboptimal feedback law $\widetilde{\mu}_0(x_0)$ such that:

1) $\widetilde{\mu}_0$, when applied in the receding horizon fashion, enforces assymptotic closed-loop stability of the system in (1) subject to constraints in (2) for all $t \geq 0$;
2) $\widetilde{\mu}_0$ is less complex than the optimal feedback law $\mu_0^\star$, i.e., it consists of fewer regions.

□

In what follows we will synthesize the suboptimal, yet stabilizing, feedback law $\widetilde{\mu}_0$ by approximate dynamic programming. We stress that the computation of $\widetilde{\mu}_0$ *does not* require the construction of $\mu_0^\star$.

## III. Approximate Dynamic programming

### A. Exact Dynamic Programming

Dynamic programming (DP) is based on the Bellman's principle of optimality. In short, it says that instead of solving one problem of size $N$ as in (3), we can instead solve $N$ problems of size one iteratively backwards in time. The problem that needs to be solved at each iteration is given by

$$J_k^\star(x_k) = \min_{u_k} \ \ell(x_k, u_k) + J_{k+1}^\star(x_{k+1}) \tag{10a}$$
$$\text{s.t. } x_{k+1} = Ax_k + Bu_k, \tag{10b}$$
$$u_k \in \mathcal{U},\ x_k \in \mathcal{X}, \tag{10c}$$
$$x_{k+1} \in \mathcal{X}_{k+1}, \tag{10d}$$

which for the first iteration (i.e., $k = N-1$), is initialized by $J_{k+1}^\star \equiv \ell_N$ (cf. (5)), and $\mathcal{X}_{k+1} \equiv \mathcal{X}_N$ (see (3d)). Solving (10) parametrically at iteration $k$ yields the cost-to-go function $J_k^\star(x_k)$ as in (9), along with the corresponding feedback law $\mu_k^\star(x_k)$ as in (6). Hence iterating with $k = N-1, N-2, \dots, 0$ yields at the end $\mu_0^\star(x_0)$, the feedback law that is equivalent to (6) obtained by solving (3) directly as a single pQP. We remark that each DP iteration also yields the corresponding feasible set $\mathcal{X}_k$ such that $\mathcal{X}_N \subseteq \mathcal{X}_{N-1} \subseteq \dots \subseteq \mathcal{X}_1 \subseteq \mathcal{X}_0$ where $\mathcal{X}_0$ is the same as in (8).

### B. Approximate Dynamic Programming

The difficulty of applying dynamic programming to solving MPC problems with the objective function in (3a) composed of quadratic stage cost and quadratic terminal penalty as in (4)−(5), stems from the fact that, already in the first iteration with $k = N-1$, $J_{N-1}^\star$ is a PWQ function as in (9). Since $J_{N-1}^\star$ is used as a cost-to-go at the subsequent iteration with $k = N-2$, the objective function in (10a), i.e., $\ell(x_{N-2}, u_{N-2}) + J_{N-1}^\star(x_{N-1})$, becomes a PWQ function and Lemma 2.3 no longer applies[1].

Therefore in this section we show how to formulate the DP iterations (10) in a way that each iteration is a convex parametric quadratic program with a quadratic objective function. This will be achieved by approximating the PWQ cost function $J_k^\star$ by a convex quadratic function $\hat{J}_k$. The approximation is performed in such a way that each DP iteration yields a feedback law $\widetilde{\mu}_k(x_k)$ that is closed-loop stabilizing for the system in (1) subject to constraints in (2).

To construct the approximate feedback law, we apply dynamic programming where the cost-to-go function is replaced by the approximate cost-to-go function $\hat{J} \approx \widetilde{J}$:

$$\widetilde{J}_k(x_k) = \min_{u_k} \ \ell(x_k, u_k) + \hat{J}_{k+1}(x_{k+1}) \tag{11a}$$
$$\text{s.t. } x_{k+1} = Ax_k + Bu_k, \tag{11b}$$
$$u_k \in \mathcal{U},\ x_k \in \mathcal{X}, \tag{11c}$$
$$x_{k+1} \in \mathcal{X}_{k+1}, \tag{11d}$$

[1]Problems of the form (10) with convex PWQ objective function could be solved either as parametric mixed-integer QPs by introducing binary variables, or by multiple pQPs (one pQP for each region), see [16]. However, in the context of linear MPC with quadratic cost, doing so would be a significant overkill both in terms of off-line construction of $\mu^\star$, as well as its on-line implementation.

initialized (for $k = N-1$) with $\hat{J}_N \equiv \ell_N$ from (5). Since in the first iteration the cost-to-go function $\hat{J}_N$ is a quadratic function, solving (11) according to Lemma 2.3 yields a PWQ function $\widetilde{J}_k(x_k)$ and a PWA feedback law $\widetilde{\mu}_k(x_k)$ (the tildes denote that the cost function and the feedback law were obtained by solving (11) for an approximation of the cost-to-go function). Assume now that at each iteration $k = N-1, \dots, 1$ there exists an approximate cost-to-go function $\hat{J}_k$ as a convex quadratic function with

$$\hat{J}_k(x) = x^\intercal \widetilde{H}_k x \tag{12}$$

with $\widetilde{H}_k = \widetilde{H}_k^\intercal \succ 0$ that satisfies

$$\widetilde{J}_k(x) \le \hat{J}_k(x) \le \widetilde{J}_k(x) + \ell(x, \widetilde{\mu}_k(x)), \tag{13}$$

for all $x \in \mathrm{dom}(\widetilde{J}_k)$. The details about constructing $\hat{J}_k$ satisfying (13) will be provided in Section III-C. Then we can prove our first main result:

*Theorem 3.1:* Assume that all conditions of Assumptions 2.1 and 2.2 are satisfied, and consider dynamic programming iterations for $k = N-1, \dots, 0$ where at each iteration we solve (11) with the approximate cost-to-go function $\hat{J}_{k+1} \approx \widetilde{J}_{k+1}$ bounded per (13). Then:

1) each instance of (11) is a convex parametric quadratic program;
2) each cost function $\widetilde{J}_k$ is a convex PWQ function;
3) each optimizer $u_k^\star = \widetilde{\mu}_k(x_k)$ is a PWA function;
4) each feasible set $\mathcal{X}_k$ is positively invariant;
5) each feedback law $\widetilde{\mu}_k$ provides closed-loop stability of (1) subject to (2).

*Corollary 3.2:* Each feedback law $\widetilde{\mu}_k$ is persistently feasible.

*Remark 3.3:* Since the constraints in (11b)−(11d) are the same as in (10b)−(10d), solving (11) at iteration $k$ by Lemma 2.3 yields a polyhedral feasible set $\mathcal{X}_k$ that is identical to that obtained by solving the exact DP problem (10) and hence $\mathrm{dom}(\widetilde{J}_k) = \mathrm{dom}(\hat{J}_k) = \mathrm{dom}(J_k^\star)$. □

We can therefore find an approximate solution to problem (3) by solving the dynamic programming iterations (11) for $k = N-1, \dots, 0$ as parametric quadratic programs. Each iteration yields the cost function $\widetilde{J}_k$, which needs to be approximated by $\hat{J}_k$ such that (13) holds. If this is possible, Theorem 3.1 establishes that $\widetilde{J}_k$ for $k = N-1, \dots, 0$ are Lyapunov functions, thus the associated minimizers $\widetilde{\mu}_k$ will be closed-loop stabilizing for the system in (1) and will enforce the constraints in (2). For the purpose of closed-loop implementation in the receding horizon manner, only $\widetilde{\mu}_0$ is required, hence $\widetilde{\mu}_i$, $i = 1, \dots, N-1$ can be discarded.

In the next section we show how to seek for the approximate quadratic cost-to-go function $\hat{J}_k$ that is bounded as in (13) by solving a convex optimization problem. Moreover, the suboptimality of $\widetilde{\mu}_0$ with respect to its optimal (but complex) counterpart $\mu_0^\star$ depends on the quality of the approximation in (13). The closer $\hat{J}_k$ is to $\widetilde{J}_k$, the smaller is the induced loss of optimality. Therefore we also show how to find $\hat{J}_k$ that is as close as possible to $\widetilde{J}_k$ while still being bounded as in (13).

### C. Cost-to-go Approximation

Let $\widetilde{J}_k$ be a convex PWQ cost function, obtained by solving the problem (11) as a parametric QP at some iteration $k$. Then, according to Lemma 2.3, $\widetilde{J}_k$ takes the form

$$\widetilde{J}_k(x) = x_k^\intercal \widetilde{H}_{k,i} x_k + \widetilde{E}_{k,i}^\intercal x_k + \widetilde{h}_{k,i} \text{ if } x_k \in \widetilde{\mathcal{R}}_{k,i}. \quad (14)$$

Moreover, $\widetilde{\mu}_k$ is the associated PWA feedback law (the parametric representation of the primal optimizer $u_k^\star = \widetilde{\mu}_k(x_k)$) with

$$\widetilde{\mu}_k(x) = \widetilde{F}_{k,i} x_k + \widetilde{f}_{k,i} \text{ if } x_k \in \widetilde{\mathcal{R}}_{k,i}, \quad (15)$$

where the polyhedral regions are given by

$$\widetilde{\mathcal{R}}_{k,i} = \{x_k \mid \widetilde{\Phi}_{k,i} x_k \leq \widetilde{\phi}_{k,i}\}, \quad (16)$$

where $i = 1, \ldots, \widetilde{M}_k$ and $\widetilde{M}_k$ is the number of regions obtained at the $k$-th iteration. To simplify the notation, we will henceforth drop the iteration index $k$ from (14)−(16).

We aim at constructing the convex quadratic function

$$\hat{J}(x) = x^\intercal \hat{H} x, \quad (17)$$

with $\hat{H} = \hat{H}^\intercal \succ 0$ such that it is lower-bounded by $\widetilde{J}(x)$, and upper-bounded by $\widetilde{J}(x) + \ell(x, \widetilde{\mu}(x))$, where $\ell(\cdot, \cdot)$ is the stage cost in (3a), i.e.:

$$\widetilde{J}(x) \leq \hat{J}(x) \leq \widetilde{J}(x) + \ell(x, \widetilde{\mu}(x)), \ \forall x \in \mathrm{dom}(\widetilde{J}). \quad (18)$$

The difficulty of solving for $\hat{J}(x)$ stems from the fact that the inequality has to hold for all $x \in \mathrm{dom}(\widetilde{J})$, i.e., for an infinite number of points. In what follows we show how to search for $\hat{H}$ in (17) by solving a convex sum-of-squares (SOS) problem.

Consider a particular region $\widetilde{\mathcal{R}}_i = \{x \mid \widetilde{\Phi}_i x \leq \widetilde{\phi}_i\}$ for which $\widetilde{\mu}(x) = \widetilde{F}_i x + \widetilde{f}_i$ and $\ell(x, \widetilde{\mu}(x)) = x^\intercal Q_\mathrm{x} x + (\widetilde{F}_i x + \widetilde{f}_i)^\intercal Q_\mathrm{u} (\widetilde{F}_i x + \widetilde{f}_i)$. Then (18) holds $\forall x \in \widetilde{\mathcal{R}}_i$ if and only if

$$\forall x \in \widetilde{\mathcal{R}}_i : \underline{p}_i(x, \hat{H}) \geq 0, \ \overline{p}_i(x, \hat{H}) \geq 0, \quad (19)$$

with

$$\underline{p}_i(x, \hat{H}) = x^\intercal \hat{H} x - (x^\intercal \widetilde{H}_i x + \widetilde{E}_i^\intercal x + \widetilde{h}_i), \quad (20a)$$

$$\overline{p}_i(x, \hat{H}) = x^\intercal \widetilde{H}_i x + \widetilde{E}_i^\intercal x + \widetilde{h}_i + x^\intercal Q_\mathrm{x} x + \\ + (\widetilde{F}_i x + \widetilde{f}_i)^\intercal Q_\mathrm{u} (\widetilde{F}_i x + \widetilde{f}_i) - x^\intercal \hat{H} x, \quad (20b)$$

where (20a) represents the lower bound and (20b) corresponds to the upper bound. Clearly, $\underline{p}_i$, $\overline{p}_i$ are polynomials in $x$ and $\hat{H}$. To enforce the validity of (19), and thus of (18), for all $x \in \widetilde{\mathcal{R}}_i$, we employ the well-known Positivstellensatz:

*Lemma 3.4 (Positivstellensatz [20]):* Let $h(z)$, $g_j(z)$, $j = 1, \ldots, c$ be polynomials in $z \in \mathbb{R}^n$. Then $h(z) \geq 0$ holds $\forall z \in \{z \mid g_j(z) \geq 0, \ j = 1, \ldots, c\}$ if there exist non-negative polynomials $s_j(z) \geq 0$, $j = 1, \ldots, c$ such that

$$h(z) - \sum_{j=1}^{c} s_j(z) g_j(z) \geq 0. \quad (21)$$

□

To see the relation between Lemma 3.4 and the problem in (19), first note that $x \in \widetilde{\mathcal{R}}_i$ can be written as $\widetilde{\phi}_i - \widetilde{\Phi}_i x \geq 0$ using the fact that $\widetilde{\mathcal{R}}_i$ is a polyhedron. Next, denote $g_j(x) := \widetilde{\phi}_i^{(j)} - \widetilde{\Phi}_i^{(j)} x$, where $Z^{(j)}$ is the $r$-th row of the vector/matrix, and let $h(z) := [\underline{p}_i, \overline{p}_i]^\intercal$. Then (19) holds for all $x \in \widetilde{\mathcal{R}}_i$, thus $\hat{J}(x)$ being bounded per (18), follows directly from Lemma 3.4, provided there exist non-negative polynomials $s_j(x, \hat{H})$, $j = 1, \ldots, n_r$. Next we report our second main result:

*Theorem 3.5:* Given are: (i) the polyhedral regions $\widetilde{\mathcal{R}}_i$ as in (16), (ii) the convex PWQ function $\widetilde{J}$ as in (14), and the PWA feedback law $\widetilde{\mu}$ as in (15). There exists a convex quadratic function $\hat{J}$ as in (17), lower/upper bounded as in (18) for all $x \in \mathrm{dom}(\widetilde{J}) = \cup_i \widetilde{\mathcal{R}}_i$ if there is a feasible solution to

$$\text{find } \hat{H}, \underline{s}_{j,i}(x, \hat{H}), \overline{s}_{j,i}(x, \hat{H}) \quad (22a)$$

$$\text{s.t. } \underline{p}_i(x, \hat{H}) - \sum_{j=1}^{c_i} \underline{s}_{j,i}(x, \hat{H}) g_{j,i}(x) \geq 0, \quad (22b)$$

$$\overline{p}_i(x, \hat{H}) - \sum_{j=1}^{c_i} \overline{s}_{j,i}(x, \hat{H}) g_{j,i}(x) \geq 0, \quad (22c)$$

$$g_{j,i}(x) = \phi_i^{(j)} - \Phi_i^{(j)} x, \quad (22d)$$

$$\underline{s}_{j,i}(x, \hat{H}) \geq 0, \overline{s}_{j,i}(x, \hat{H}) \geq 0, \quad (22e)$$

$$\hat{H} \succ 0, \quad (22f)$$

where all constraints have to hold for $i = 1, \ldots, \widetilde{M}$, and for each $i$, $j = 1, \ldots, c_i$, where $c_i$ is the number of defining half-spaces of the $i$-th polyhedron.

*Remark 3.6:* The problem in (22) can be formulated and solved as a convex semi-definite program. To see this, recall (see, e.g., [21]) that a polynomial $p(z)$ is globally non-negative if it can be written as sum of squares, i.e., $p(v) = v^\intercal Q v$ where $v$ is a vector of monomials and $Q$ is a symmetric, positive definite matrix. It follows that each non-negativity constraint in (22) can in turn be translated into a linear matrix inequality constraint, and (22) can be formulated by YALMIP [22] and solved using semi-definite programming packages, such as SeDuMi or MOSEK. □

Clearly, the approximation $\hat{J}$ satisfying (18) is not unique. In the context of approximate dynamic programming, it is desirable for $\hat{J}$ to be "as close as possible" to the lower bound, i.e., to $\widetilde{J}$ in some measure. By minimizing the distance of $\hat{J}$ to $\widetilde{J}$ we minimize the induced loss of optimality. We propose two viable, cheap to implement options to achieve this goal. The first one is based on replacing the feasibility objective in (22a) by minimization of the trace of $\hat{H}$. By doing so, we inderectly minimize the quadrature of $\hat{J}$ in (17) and bring it closer to the lower bound $\widetilde{J}$. The second option is to directly minimize the approximation error, for instance the worst-case error between $\hat{J}$ and $\widetilde{J}$. This can be done by minimizing a scalar $t$ subject to $t \geq \hat{J}(x) - \widetilde{J}(x) \ \forall x \in \widetilde{\mathcal{R}}_i$, $i = 1, \ldots, \widetilde{M}$. By using the PWQ nature of $\widetilde{J}$, this is equivalent to

$$t - \underline{p}_i(x, \hat{H}) \geq 0, \ \forall x \in \widetilde{\mathcal{R}}_i, \ i = 1, \ldots, \widetilde{M}, \quad (23)$$

where $\underline{p}_i$ is as in (20a). Thus by introducing the variable $t$ into (22), and by adding (23) into the constraints, one obtains the approximation that minimizes the worst-case

approximation error. The constraint (23) can be enforced for all $x \in \widetilde{\mathcal{R}}_i$ via Lemma 3.4 and converted to an SDP constraint according to Remark 3.6.

### D. Complete Algorithm

The approximated stabilizing PWA feedback law $\widetilde{\mu}_0$ that solves Problem 2.5 can be obtained in a dynamic programming fashion as follows:

- Initialization: set $\hat{J}_N \equiv \ell_N$ and let $\mathcal{X}_N$ be as in (3d), cf. Assumption 2.2.
- DP iterations: for each $k = N-1, \dots, 1$ do
  1) solve the pQP (11) with the approximate cost-to-go function $\hat{J}_{k+1}$ and the terminal set $\mathcal{X}_{k+1}$; obtain the PWQ cost function $\widetilde{J}_k$, the PWA feedback law $\widetilde{\mu}_k$, and the feasible set $\mathcal{X}_k = \cup_i \widetilde{\mathcal{R}}_{k,i}$;
  2) solve problem (22) as a convex SDP according to Remark 3.6, obtain $\hat{H}_k$ and let $\hat{J}_k = x^\intercal \hat{H}_k x$.
- Final step: solve the pQP (11) with $\hat{J}_1$ as the cost-to-go function and $\mathcal{X}_1$ as the terminal set, obtain the receding horizon feedback law $\widetilde{\mu}_0$.

Therefore we need to solve $N$ pQP problems of the form (11), together with $N-1$ approximation problems (22) required to find $\hat{J}_k$ for $k = N-1, \dots, 1$ (notice that the approximation is not needed in the final step since the last one that is used is $\hat{J}_1$).

## IV. Case Study

We aim to control an inverted pendulum mounted at a cart, shown in Fig. 1 and described in more details in [23]. The continuous-time dynamics of the system, linearized around the upright unstable equilibrium point, is given by

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{3g}{2l} & -b & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{3}{2l} \\ 0 \\ 1 \end{bmatrix} u(t) \qquad (24)$$

with the state vector $x = [\Theta \ \dot{\Theta} \ d \ \dot{d}]^\intercal$. Here, $\Theta$ is the angle of pendulum from its upright position in radians, $\dot{\Theta}$ is the angular velocity in radians per second, $d$ is the cart's position in metres, $\dot{d}$ is the cart's velocity in metres per second, and $u$ is acceleration of the cart in $\mathrm{m\,s^{-2}}$. Moreover, $g = 9.8067\,\mathrm{m\,s^{-2}}$ represents gravitation acceleration, $l = 0.21\,\mathrm{m}$ is the length of the pendulum, and $b = 1\,\mathrm{s^{-1}}$ is the friction factor that was identified experimentally. The dynamics in (24) was subsequently discretized with sampling time $T_\mathrm{s} = 0.02\,\mathrm{s}$ and converted to the form of (1).

The system operates under hard state and input constraints

$$\begin{bmatrix} -0.1920 \\ -3.1416 \\ -0.2500 \\ -0.8500 \end{bmatrix} \leq x(t) \leq \begin{bmatrix} 0.1920 \\ 3.1416 \\ 0.2500 \\ 0.8500 \end{bmatrix}, \ -10 \leq u(t) \leq 10. \qquad (25)$$

Since the pendulum's angle is expressed in radians, the first constraint translates to $-11^\circ \leq \Theta \leq 11^\circ$, which is the range of validity of the linearization in (24).

The control objective is to reject disturbances and to stabilize the states of the system towards the origin. Two MPC controllers were considered, both with prediction horizon $N = 4$, $Q_\mathrm{x} = \mathrm{diag}(10^4, 1, 10^2, 1)$ and $Q_\mathrm{u} = 10$ in (4), $Q_\mathrm{N}$ chosen as the solution of the discrete-time Riccati equation, and $\mathcal{X}_N$ being the positively invariant subset subject to the corresponding LQR controller. With this setting, $Q_\mathrm{N}$ and $\mathcal{X}_N$ satisfy all conditions of Assumption 2.2.

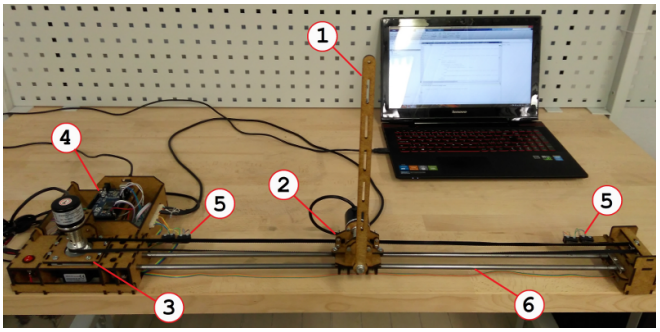
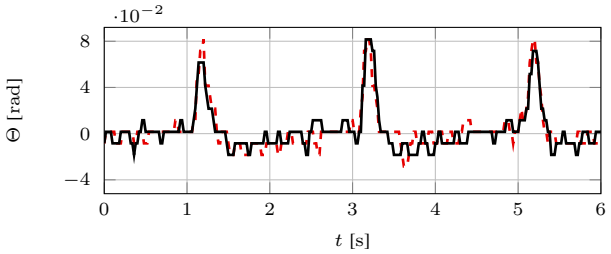



Fig. 1. Inverted pendulum on a cart. 1: pendulum, 2: moveable cart with incremental rotary encoder, 3: stepper motor as an actuator and incremental rotary encoder attached on it, 4: micro-controller board and stepper motor driver, 5: limit switches, 6: leading rods.

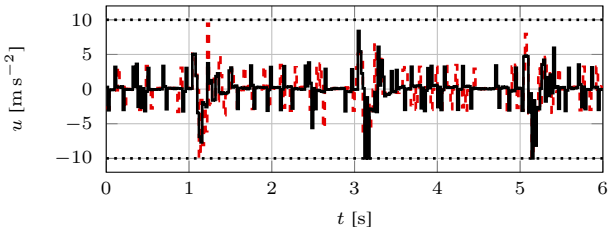
The first controller, denoted as $\mu_0^\star$, was computed by solving the CFTOC problem (3) according to Lemma 2.3 by a parametric programming solver contained in the Multi-Parametric Toolbox [24]. The optimal parametric receding horizon feedback law $\mu^\star$ was obtained after 36 seconds and it consisted of 351 polyhedral regions $\mathcal{R}_i$. The total memory footprint of the optimal feedback law, i.e., the space occupied by vectors and matrices $\Phi_i$, $\phi_i$, $F_i$, $f_i$, was 73 kilobytes, cf. (6) and (7).

The second controller, denoted by $\widetilde{\mu}_0$, was obtained by applying approximate dynamic programming per the procedures of Section III. The SDPs were solved by Mosek. The total time of the overal approximate dynamic programming procedure was $82\,\mathrm{s}$ and the resulting approximate, yet stabilizing feedback law $\widetilde{\mu}_0$ consisted of only 47 regions. Thus the complexity, measured in terms of number of regions, was reduced by a factor of 7.5 from 351 to 47 regions. The total memory footprint of $\widetilde{\mu}_0$ was 13.6 kilobytes, a reduction by a factor of 5.4 compared to $\mu_0^\star$.

Both controllers were then used to gather experimental data to asses their respective performance. To this end, evaluation of the PWA feedback law $\mu_0^\star$ and $\widetilde{\mu}_0$ for a given value of the state measurements was implemented in Matlab using the sequential search approach described in Remark 2.4. The calculated optimal control inputs were then transmitted to the pendulum via a serial interface. Each experiment started with the pendulum in its upright unstable equilibrium and with the cart in rest. Subsequently, a disturbance of $5\,\mathrm{m\,s^{-2}}$ was added to the control action every $2\,\mathrm{s}$ for the duration of 4 sampling periods, i.e., for $80\,\mathrm{ms}$. The experimental results are depicted in Figure 2. One can observe that performance of the optimal controller $\mu_0^\star$ and its approximate counterpart $\widetilde{\mu}_0$ are very similar. Both controllers reject disturbances and stabilize the pendulum in its upright position while respecting constraints. The worst-case implementation time of $\mu_0^\star$, i.e., the time required to calculate $u_0^\star = \mu_0^\star(x(t))$ by evaluating the PWA function in (6) via sequential search, was $3.23\,\mathrm{ms}$.




(a) Profile of the pendulum's angle $\Theta(t)$.




(b) Acceleration of the cart $u(t)$.

Fig. 2. State and input profiles for the approximate feedback law $\widetilde{\mu}_0$ obtained via approximate dynamic programming (solid lines) and for the optimal controller constraints on the control input.

On the other hand, the worst-case runtime of $\widetilde{\mu}_0$ was only 1.04 ms. This decrease is due to the fact that $\widetilde{\mu}_0$ is defined over fewer regions and the sequential search of Remark 2.4 converges faster. We remark that the chattering visible in the profiles in Fig. 2 is caused by rotational encoders having only a finite precision.

## V. Conclusions

We have shown that QP-based MPC problems can be solved in dynamic programming fashion by considering a quadratic approximation of the cost-to-go function. Moreover, when the approximation is bounded as in (13) in each DP iteration, the resulting approximate feedback law provides guarantees of closed-loop stability and recursive feasibility *by construction*. Moreover, since each DP iteration technically corresponds to solving an MPC problem with the prediction horizon equal to one, the approximate feedback law is simpler than its optimal counterpart. By means of an experimental case study we have demonstrated that 1) a considerable reduction of the controller complexity could indeed be achieved both in terms of memory as well as the on-line implementation effort, and 2) that the approximate feedback law, for the specific case study of this paper, features low suboptimality.

## Acknowledgments

The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grant no. 1/0403/15, and the financial support of the Slovak Research and Development Agency under the project APVV-15-0007.

## References

[1] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967 – 2986, 2014.
[2] J. Maciejowski, *Predictive Control with Constraints*. Prentice-Hall, 2001.
[3] 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, Jan. 2002.
[4] L. Willner, "On Parametric Linear Programming," *SIAM Journal on Applied Mathematics*, vol. 15, no. 5, pp. 1253–1257, Sep. 1967.
[5] T. Gal and J. Nedoma, "Multiparametric linear programming," *Management Science*, vol. 18, pp. 406–442, 1972.
[6] F. Borrelli, *Constrained Optimal Control of Linear and Hybrid Systems*, ser. lncis. Springer-Verlag, 2003, vol. 290.
[7] 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, Sep. 2008.
[8] M. Kvasnica, B. Takács, J. Holaza, and S. Di Cairano, "On region-free explicit model predictive control," in *54rd IEEE Conference on Decision and Control*, vol. 54, Osaka, Japan, December 15-18, 2015 2015, pp. 3669–3674.
[9] 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.
[10] T. Geyer, F. D. Torrisi, and M. Morari, "Optimal complexity reduction of polyhedral piecewise affine systems," *Automatica*, vol. 44, no. 7, pp. 1728 – 1740, 2008.
[11] D. Ingole, M. Kvasnica, H. De Silva, and J. Gustafson, "Reducing memory footprints in explicit model predictive control using universal numbers," in *Preprints of the 20th IFAC World Congress, Toulouse, France*, vol. 20, July 9-14, 2017 2017, pp. 12 100–12 105.
[12] G. Valencia-Palomo and J. Rossiter, "Using Laguerre functions to improve efficiency of multi-parametric predictive control," in *Proceedings of the American Control Conference*, Baltimore, USA, 2010, pp. 4731–4736.
[13] M. Kvasnica, J. Löfberg, and M. Fikar, "Stabilizing polynomial approximation of explicit MPC," *Automatica*, vol. 47, no. 10, pp. 2292–2297, 2011.
[14] M. Jones, Colin N.; Morari, "Polytopic approximation of explicit model predictive controllers," *IEEE Transactions on Automatic Control*, vol. 55, 11 2010.
[15] T. A. Johansen and A. Grancharova, "Approximate explicit constrained linear model predictive control via orthogonal search tree." *IEEE Trans. Automat. Contr.*, vol. 48, no. 5, pp. 810–815, 2003.
[16] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709–1721, 2005.
[17] K. Kouramas, N. Faísca, C. Panos, and E. Pistikopoulos, "Explicit/multi-parametric model predictive control (mpc) of linear discrete-time systems by dynamic and multi-parametric programming," *Automatica*, vol. 47, no. 8, pp. 1638–1645, 2011.
[18] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control*. Cambridge University Press, 2017. [Online]. Available: http://www.mpc.berkeley.edu/mpc-course-material
[19] M. Baotić, "Optimal Control of Piecewise Affine Systems – a Multi-parametric Approach," Dr. sc. thesis, ethz, Zurich, Switzerland, Mar. 2005.
[20] G. Stengle, "A Nullstellensatz and a Positivstellensatz in semialgebraic geometry ," *Math. Ann.*, vol. 207, pp. 87–97, 1974.
[21] P. A. Parrilo, "Sums of squares of polynomials and their applications," in *ISSAC '04: Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, New York, NY, USA, 2004, pp. 1–1.
[22] J. Löfberg, "YALMIP : A Toolbox for Modeling and Optimization in MATLAB," in *Proc. of the CACSD Conference*, Taipei, Taiwan, 2004.
[23] P. Bakaráč, M. Kalúz, and Ľ. Čirka, "Design and development of a low-cost inverted pendulum for control education," in *Proceedings of the 21st International Conference on Process Control*, 2017, pp. 398–403.
[24] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the European Control Conference*, Zürich, Switzerland, July 17–19 2013, pp. 502–510, http://control.ee.ethz.ch/~mpt.