Parallel Explicit MPC for Hardware with Limited Memory

Juraj Oravec * **Yuning Jiang** ** **Boris Houska** **
Michal Kvasnica *

* *Faculty of Chemical and Food Technology, Slovak University of Technology in Bratislava, Slovak Republic. ({juraj.oravec,michal.kvasnica}@stuba.sk)*
** *School of Information Science and Technology, ShanghaiTech University, China. ({jiangyn,borish}@shanghaitech.edu.cn)*

Abstract: This paper proposes a real-time model predictive control (MPC) algorithm for problems with convex quadratic objectives, linear dynamic systems, as well as polytopic state- and control constraints. The method features, on the one hand, explicit model predictive control algorithms and parametric optimization, but, on the other hand, also uses ideas from iterative distributed optimization. This leads to an explicit real-time algorithm with constant and verifiable run-time bounds that scale linearly with the prediction horizon length of the MPC problem. Moreover, unlike standard explicit MPC algorithms, the memory requirements of the proposed algorithm do not dependent on the prediction horizon length. The practical advantages of the method are illustrated with a numerical example.

Keywords: Explicit MPC, Parametric Optimization, Distributed Optimization

1. INTRODUCTION

Explicit model prediction control (MPC), pioneered by Bemporad et al. (2002), is a popular control methodology that is based on pre-computing, off-line, the closed-form solution to a given MPC optimization problem using paramtric programming. When the MPC problem is in the form of a strictly convex quadratic program (QP), the closed-form solution is known to be a piecewise affine (PWA) function over polyhedral critical regions that maps state measurements onto optimal control inputs. The on-line implementation of such solutions then boils down to a mere function evaluation that can be performed quickly even in hardware with limited computational resources. Due to its simple and fast implementation, explicit MPC has found it ways into a plethora of control applications, see the comprehensive overview by Oberdieck et al. (2016) and references therein.

Despite intensive research in the field (see the overview in Alessio and Bemporad (2009)), two limitations of explicit MPC prevail up to these days, both of which are directly related to the size of the MPC problem that can be tackled. First, the inherited computational complexity of parametric optimization techniques only allows to synthesize closed-form solutions for problems of modest size (i.e., for small values of the prediction horizon, the number of manipulated inputs and the number of states). This is due to the fact that the number of critical regions to be discovered off-line grows, in the worst case, exponentially with the prediction horizon [1]. Although the construction of the solution is done off-line, it can still take a prohibitive amount of time. Secondly, once the solution is computed, it is often too complex for typical control hardware in terms of the memory footprint (which is a linear function of the number of critical regions) and the on-line evaluation time.

The first limitation is typically addressed by simplifying the MPC setup, e.g., by using minimum-time formulations (Grieder et al., 2005) or by employing move blocking (Cagienard et al., 2007), also see Oberdieck et al. (2016) and references therein. Although such methods allow to simplify the MPC problem and thus to accelerate the off-line construction of the explicit solution, they induce suboptimality. The second limitation is often tackled by using complexity reduction schemes which, a-posteriori, simplify the explicit solution as to decrease the number of regions and thus to decrease the memory footprint of explicit solutions, see Holaza et al. (2015) and references therein. The common drawback of almost all complexity reduction schemes is that they are, from a practical point of view, limited to fairly low dimensions of the parametric space and typically require a significant amount of processing time.

In this paper we take a different route. We develop an explicit MPC algorithm for standard convex linear-quadratic MPC problems that is, on the one hand, solely based on explicit piecewise affine solution maps, but, on the other hand, uses ideas from iterative distributed optimization in order to break long prediction horizons into smaller ones on which explicit MPC is effective. Since the complexity of the explicit solution developed here does not depend on the prediction horizon, the procedure of

[1] More precisely, the number of critical regions is a function of the total number of constraints (which typically grow linearly with the prediction horizon), and the number of optimized variables (i.e., $n_{\mathrm{u}}N$).

this paper allow explicit MPC to be synthesized, and implemented, even for large horizons (≈ 100).

Notice that there exist a huge amount of literature on distributed convex optimization. One class of such methods are dual decomposition algorithms (Everett, 1963), which have been applied to linear MPC by many authors, e.g., Giselsson et al. (2013), also in combination with explicit MPC (Trnka et al., 2016; Zanarini et al., 2013). Moreover, the alternating direction method of multipliers (ADMM) (Boyd et al., 2011) has been applied to solve optimal control (O'Donoghue et al., 2013) as well as linear MPC problems (Mota et al., 2012). However, the local convergence rate of both dual decomposition in combination with gradient ascent methods as used by Giselsson et al. (2013); Necoara and Suykens (2008); Pu et al. (2014) as well as ADMM is typically at most linear. Therefore, this paper uses a variant of the recently proposed augmented Lagrangian based alternating direction inexact Newton (ALADIN) method (Houska et al., 2016), which can achieve locally quadratic convergence rates. Notice that ALADIN has been found suitable for solving band-structured nonlinear programming problems as arising in the context of optimal control (Kouzoupis et al., 2016).

The main contribution of the paper is a parallelizable explicit MPC algorithm based on ALADIN. This algorithm can be implemented on embedded hardware, where only static memory and very limited storage space is available. The case study illustrates that the proposed strategy leads to a significant reduction of the off-line computational complexity (given as the time required to construct explicit maps) and the on-line storage complexity.

2. BACKGROUND ON PARAMETRIC QUADRATIC PROGRAMMING

The problem

$$\min_z \ \frac{1}{2} z^\top H z + \theta^\top J z + c^\top z, \tag{1a}$$

$$\text{s.t. } Gz \leq w + E\theta, \tag{1b}$$

with the optimization variable $z \in \mathbb{R}^{n_z}$ and a vector of parameters $\theta \in \mathbb{R}^{n_\theta}$ is referred to as a *parametric quadratic program* (pQP). Here, $H \in \mathbb{R}^{n_z \times n_z}$, $J \in \mathbb{R}^{n_\theta \times n_z}$, $c \in \mathbb{R}^{n_z}$, $G \in \mathbb{R}^{n_c \times n_z}$, $w \in \mathbb{R}^{n_c}$, $E \in \mathbb{R}^{n_c \times n_\theta}$ are known problem data. Moreover, we denote by $\Psi = \{\theta \in \mathbb{R}^{n_\theta} \mid \exists z \text{ s.t. } Gz \leq w + E\theta\}$ the set of all parameters θ for which (1) is feasible. Note, $\mathbb{R}^n$ and $\mathbb{R}^{n \times m}$ denote the set of n-dimensional real vectors and $n \times m$ real matrices, respectively.

It is well known (see, e.g., Bemporad et al. (2002)) that for strictly convex pQPs (1), the optimizer $z^\star : \Psi \to \mathbb{R}^{n_z}$ is a continuous polyhedral piecewise affine (PWA) function of the parameters θ:

$$z^\star(\theta) = \begin{cases} F_1\theta + f_1 & \text{if } \theta \in \mathcal{P}_1, \\ \ \vdots & \\ F_{N_\mathrm{R}}\theta + f_{N_\mathrm{R}} & \text{if } \theta \in \mathcal{P}_{N_\mathrm{R}}, \end{cases} \tag{2}$$

where $F_i \in \mathbb{R}^{n_z \times n_\theta}$, $f_i \in \mathbb{R}^{n_z}$, and $\mathcal{P}_i = \{\theta \mid P_i\theta \leq p_i\}$ are so-called polyhedral *critical regions* with $P_i \in \mathbb{R}^{c_i \times n_\theta}$, $p_i \in \mathbb{R}^{c_i}$, and c_i is the number of defining half-spaces of the i-th region for $i = 1, \ldots, N_\mathrm{R}$. Here, N_R refers to the total number of critical region in the parametric solution (2) and its value is entirely problem-dependent. The data of (2), i.e., the gains F_i, f_i, along with the half-spaces of critical regions captured in P_i, p_i are computed by *parametric programming*. In what follows we provide a concise overview of available approaches. For a more elaborate description, the reader is referred to Oberdieck et al. (2016).

Most of available pQP solvers operate as active set methods with a complete factorization of the Karush-Kuhn-Tucker (KKT) optimality conditions. Specifically, let $\mathcal{A} \subseteq \{1, \ldots, n_c\}$ be an index set of constraints that are active in (1b) for some $\theta \in \Psi$. Then the minimizer $z^\star$ to (1) is identical to the minimizer to (1a) subject to $G_\mathcal{A} z = w_\mathcal{A} + E_\mathcal{A}\theta$ where $G_\mathcal{A}$ is a matrix created by retaining only the rows indexed by $\mathcal{A}$. The Lagrangian of this equality-constrained problem is $\mathcal{L}(z, \lambda) = 1/2 \ z^\top H z + \theta^\top J z + c^\top z + \lambda^\top (G_\mathcal{A} z - w_\mathcal{A} - E_\mathcal{A}\theta)$ and its stationarity condition $\nabla\mathcal{L}(z^\star, \lambda^\star) = 0$ yields

$$\begin{bmatrix} H & G_\mathcal{A}^\top \\ G_\mathcal{A} & 0 \end{bmatrix} \begin{bmatrix} z^\star \\ \lambda^\star \end{bmatrix} = \begin{bmatrix} -c - J^\top\theta \\ w_\mathcal{A} + E_\mathcal{A}\theta \end{bmatrix}. \tag{3}$$

Under the assumption that H is invertible and $G_\mathcal{A}$ is of full row rank, solving for $z^\star$, $\lambda^\star$ from (3) gives

$$z^\star = F(\mathcal{A})\theta + f(\mathcal{A}), \ \lambda^\star = L(\mathcal{A})\theta + \ell(\mathcal{A}), \tag{4}$$

with

$$F(\mathcal{A}) = -H^{-1}\left(G_\mathcal{A}^\top M_\mathcal{A}(G_\mathcal{A} H^{-1} J^\top + E_\mathcal{A}) + J^\top\right), \tag{5a}$$

$$f(\mathcal{A}) = -H^{-1}\left(G_\mathcal{A}^\top M_\mathcal{A}(G_\mathcal{A} H^{-1} c + w_\mathcal{A}) + c\right), \tag{5b}$$

$$L(\mathcal{A}) = M_\mathcal{A}(G_\mathcal{A} H^{-1} J^\top + E_\mathcal{A}), \tag{5c}$$

$$\ell(\mathcal{A}) = M_\mathcal{A}(G_\mathcal{A} H^{-1} c + w_\mathcal{A}), \tag{5d}$$

where $M_\mathcal{A} = -(G_\mathcal{A} H^{-1} G_\mathcal{A}^\top)^{-1}$. Therefore, as long as $\mathcal{A}$ remains the optimal active set, the primal as well as the dual optimizer are affine functions of θ. The subset of Ψ where $\mathcal{A}$ is optimal is obtained by plugging (4) into primal and dual feasibility conditions $G_\mathcal{N} z^\star < w_\mathcal{N} + E_\mathcal{N}\theta$ and $\lambda^\star \geq 0$, respectively, where $\mathcal{N} = \{1, \ldots, n_c\} \setminus \mathcal{A}$ is the index set of inactive constraints. This yields the critical region

$$\mathcal{P}(\mathcal{A}) = \{\theta \mid G_\mathcal{N}\left(F(\mathcal{A})\theta + f(\mathcal{A})\right) < w_\mathcal{N} + E_\mathcal{N}\theta, \\ L(\mathcal{A})\theta + \ell(\mathcal{A}) \geq 0 \ \}. \tag{6}$$

Since all constraints in (6) are linear in θ, the set $\mathcal{P}(\mathcal{A}) \subseteq \Psi$ is a polyhedron and describes the set of parameters for which $\mathcal{A}$ is the optimal active set and for which $z^\star$ in (4) is the optimizer to (1).

The full parametric solution in (2) is then obtained by first determining all distinct optimal active sets $\mathscr{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_{N_\mathrm{R}}\}$, followed by computing the corresponding local affine optimizers $F_i := F(\mathcal{A}_i)$, $f_i := f(\mathcal{A}_i)$, and the associated critical regions $\mathcal{P}_i := \mathcal{P}(\mathcal{A}_i)$ for $i = 1, \ldots, N_\mathrm{R}$ as described above. Depending on the way all optimal active sets are enumerated, we can divide available parametric solvers into two big groups: geometric solvers and enumerative solvers.

Choosing a particular algorithm should be done based on two quantities: the number of parameters n_θ and the number of optimization variables n_z. If $n_z \leq 10$, enumerative solvers perform well irrespective of the number of parameters and outperform geometric solvers, see, e.g., Drgoňa et al. (2016). Therefore they are, in our experience, the preferred way to go. If $n_z > 10$ but $n_\theta \leq 10$, geometric

algorithms are preferred. Cases with $n_\theta > 10$ and $n_z > 10$ have, in our experience, no clear winner.

Once the parametric optimizer in (2) is constructed, its evaluation for a particular value of the parameter is a two step procedure. First, the index $i^\star$ of the critical region that contains θ needs to be determined. This task is referred to as the *point location problem*. Once $i^\star$ is known, $z^\star$ is computed by $z^\star = F_{i^\star}\theta + f_{i^\star}$. Various point location algorithms exist, ranging from a simple (but inefficient) traversal of the list of all critical regions in a linear fashion, through building of various search structures such as the binary search tree (Tøndel et al., 2003), or using descriptor functions (Baotic et al., 2008). A more elaborate discussion of available method, however, goes beyond the scope of this paper.

3. PARALLEL EXPLICIT MPC

This paper concerns linear-quadratic MPC problems,

$$\begin{aligned} \min_{x,u} \; & \sum_{k=0}^{N-1} \left\{ \|x_k\|_Q^2 + \|u_k\|_R^2 \right\} + \|x_N\|_P^2, \\ \text{s.t.} \; & \begin{cases} \forall k \in \{0,\dots,N-1\}, \\ x_{k+1} = Ax_k + Bu_k \,,\; x_0 = \hat{x} \\ (x_k, u_k) \in \mathbb{X} \times \mathbb{U} \,. \end{cases} \end{aligned} \tag{7}$$

Here, $x_k \in \mathbb{R}^{n_\mathrm{x}}$ and $u_k \in \mathbb{R}^{n_\mathrm{u}}$ denote the states and inputs and N is the prediction horizon. The matrices $A \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$, $B \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{u}}$, $Q \in \mathbb{S}_+^{n_\mathrm{x}}$, $R \in \mathbb{S}_{++}^{n_\mathrm{u}}$, and $P \in \mathbb{S}_{++}^{n_\mathrm{x}}$ are assumed to be constant and given. [2] Here, $\mathbb{S}_{++}^n$ ($\mathbb{S}_+^n$) denotes the set of real symmetric and positive (semi-)definite $n \times n$ matrices. For any given $x \in \mathbb{R}^n$, $Q \in \mathbb{S}_+^n$, $\|x\|_Q^2 = x^\top Q x$ is the weighted squared 2-norm.

Assumption 3.1. The state and input constraint sets, $\mathbb{X}$ and $\mathbb{U}$, are non-empty closed and convex polyhedra.

Assumption 3.2. We have $(0,0) \in \mathbb{X} \times \mathbb{U}$.

In the context of MPC, problem (7) is solved whenever a new state-estimate $\hat{x}$ becomes available (Rawlings and Mayne, 2009). The explicit solution map $u_0^\star(\hat{x})$ of the regular parametric quadratic program (7) could be obtained by applying the techniques from the previous section. However, for large N this is not recommendable, since storing the explicit solution map may in this case become very expensive.

3.1 Cost-to-travel functions

The cost-to-travel function

$$V : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R} \cup \{\infty\}$$

is defined as the optimal value of the optimization problem

$$\begin{aligned} V(a,b) = \min_{x,u} \; & \sum_{k=0}^{n-1} \left\{ \|x_k\|_Q^2 + \|u_k\|_R^2 \right\}, \\ \text{s.t.} \; & \begin{cases} \forall k \in \{0,1,\dots,n-1\}, \\ x_{k+1} = Ax_k + Bu_k, \\ (x_k, u_k) \in \mathbb{X} \times \mathbb{U}, \\ x_0 = a \;,\; b = x_n \in \mathbb{X}, \end{cases} \end{aligned} \tag{8}$$

for all $a, b \in \mathbb{R}^{n_x}$ and a given strictly positive integer $n \in \mathbb{N}$. We set $V(a,b) = \infty$ if the above optimization problem is infeasible. If Assumptions 3.1 and 3.2 hold, V is a closed, proper, and convex function (Boyd and Vandenberghe, 2004). The value $V(a,b)$ can be interpreted as the cost to travel from the point a to the point b in n steps. In the following, we denote by $u_0^\star(a,b)$ the optimal solution for u_0 of (8).

Proposition 3.3. If Assumption 3.1 is satisfied, the domain

$$\mathbb{D}_n = \{(a,b) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \mid V(a,b) < \infty\}$$

is a closed convex polyhedron and V is for every given $n \in \mathbb{N}$ a piecewise quadratic function on $\mathbb{D}_n$. The function $u_0^\star$ is piecewise affine on $\mathbb{D}_n$.

Proof. The statement of this proposition follows immediately from the fact that (8) is a parametric quadraric program, see (Borrelli et al., 2011, Chapter 12) for details. □

Parametric quadratic programming can be used to store the function V explicitly. For many practical problems, however, the storage complexity increases dramatically with larger n. Of course, if $N = Mn$ for a moderately small n and $M \in \mathbb{N}$, one could write (7) in the equivalent form

$$\min_z \; \sum_{i=0}^{M-1} \{V(z_i, z_{i+1})\} + \|z_M\|_P^2 \quad \text{s.t.} \quad z_0 = \hat{x} \,.$$

Here, $z = \left(x_0^\top, x_n^\top, x_{2n}^\top, \dots, x_{Mn}^\top\right)^\top$ is a stacked vector that collects all coupling variables at the boundaries of the shorter horizons. This problem could in principle be solved by storing the function V explicitly and then using a suitable NLP solver to solve the above optimization problem. However, such a construction cannot be recommended for practical use, as the function V is non-differentiable. In the following, the notation

$$\begin{aligned} W(a,b) = \min_{x,u} \; & \sum_{k=0}^{n-1} \left\{ \|x_k\|_Q^2 + \|u_k\|_R^2 \right\}, \\ \text{s.t.} \; & \begin{cases} \forall k \in \{0,1,\dots,n-1\}, \\ x_{k+1} = Ax_k + Bu_k, \\ x_0 = a \;,\; b = x_n, \end{cases} \end{aligned} \tag{9}$$

is used to an approximate cost-to-travel function.

3.2 Augmented cost-to-travel functions

As technical preparation for developing a parallel explicit MPC scheme, the following definition of augmented cost-to-travel functions [3] is introduced:

$$V_P(\theta) = \min_{a,b} \left\{ V(a,b) + \begin{bmatrix} -\theta_1 \\ \theta_2 \end{bmatrix}^\top \begin{bmatrix} a \\ b \end{bmatrix} + \left\| \begin{bmatrix} a - \theta_3 \\ b - \theta_4 \end{bmatrix} \right\|_P^2 \right\}. \tag{10}$$

Here, $\theta = (\theta_1, \theta_2, \theta_3, \theta_4) \in \mathbb{R}^{n_x \times 4}$ is a parameter, whose first two rows denote multipliers of the coupling constraints and whose last two rows denote the center of the

[2] The matrix P can optionally be chosen by solving an algebraic Ricatti equation such that the terminal cost equals the infinite horizon LQR cost, see Rawlings and Mayne (2009).

[3] The definition of augmented cost-to-travel functions is equivalent to the definition of augmented Lagrangian functions in Houska et al. (2016).

regularization term of the augmented cost. Additionally, the notation

$$\forall c, d \in \mathbb{R}^{n_x}, \quad \left\| \begin{bmatrix} c \\ d \end{bmatrix} \right\|_P^2 = \|c\|_P^2 + \|d\|_P^2 \,,$$

is used. Notice that V_P is a piecewise quadratic function.

Remark 3.4. The function V_P can be interpreted as an augmented Lagrangian function, whose properties have been analysed exhaustively in the nonlinear optimization literature (Bertsekas, 2012a; Hestenes, 1969; Tapia, 1978; Powell, 1969). □

3.3 Generalized gradient maps

The generalized gradient maps

$$\mathcal{G}_\mathrm{a}, \mathcal{G}_\mathrm{b} : \mathbb{R}^{n_x \times 4} \to \mathbb{R}^{n_x}$$

are defined as

$$\forall \theta \in \mathbb{R}^{n_x \times 4}, \quad \left\{ \begin{array}{l} \mathcal{G}_\mathrm{a}(\theta) = \theta_1 + 2P(\theta_3 - a^\star(\theta)), \\ \mathcal{G}_\mathrm{b}(\theta) = -\theta_2 + 2P(\theta_4 - b^\star(\theta)) \,, \end{array} \right\}, \quad (11)$$

where $a^\star(\theta)$ and $b^\star(\theta)$ are the parametric minimizers of (10) as in (2). Since $a^\star$ and $b^\star$ are piecewise affine functions, the functions $\mathcal{G}_\mathrm{a}$ and $\mathcal{G}_\mathrm{b}$ are piecewise affine, too. The first order optimality conditions of (10) imply

$$\mathcal{G}_\mathrm{a}(\theta) \in \partial_a V(a^\star(\theta), b^\star(\theta)) \quad (12)$$

$$\text{and} \quad \mathcal{G}_\mathrm{b}(\theta) \in \partial_b V(a^\star(\theta), b^\star(\theta)) \,, \quad (13)$$

where $\partial_a V$ and $\partial_b V$ denote the sub-differentials (Rockafellar, 1970) of V with respect to its first and second argument, respectively. In this sense, $\mathcal{G}_\mathrm{a}$ and $\mathcal{G}_\mathrm{b}$ can be regarded as generalized gradient maps. Notice that the complexity of storing $\mathcal{G}_\mathrm{a}$ and $\mathcal{G}_\mathrm{b}$ is independent of N. In particular, if n is small, e.g., $n = 1$, the functions $\mathcal{G}_\mathrm{a}$ and $\mathcal{G}_\mathrm{b}$ can be stored without needing much memory.

3.4 Parametric linear-quadratic regulation

Besides the above generalized gradient maps, this paper is based on linear-quadratic consensus steps that can be implemented by solving problems of the form [4]

$$\min_{\xi,\phi} \sum_{k=0}^{M-1} \left\{ W(\xi_k, \phi_k) + q_k^\top \xi_k + r_k^\top \phi_k \right\} + \|\xi_M\|_P^2 + q_M^\top \xi_M,$$
$$\text{s.t.} \left\{ \begin{array}{ll} \forall k \in \{0, 1, \ldots, M-1\}, & \\ \phi_k = c_k + \xi_{k+1}, & \mid \lambda_{k+1}, \\ \xi_0 = 0, & \mid \lambda_0 \,, \end{array} \right. \quad (14)$$

where $q = (q_0, \ldots, q_n)$ and $r = (r_0, \ldots, r_{M-1})$ are parametric gradient corrections and $c = (c_0, \ldots, c_{M-1})$ are parametric consensus gaps. Recall that the function W has been defined in (9). Problem (14) can be interpreted as an affinely parameterized linear-quadratic regulator (LQR). Consequently, affine solution maps $\mathcal{F}_\xi, \mathcal{F}_\phi$ and $\mathcal{F}_\lambda$ can be found such that

$$\xi^\star = \mathcal{F}_\xi(c, q, r) \,, \; \phi^\star = \mathcal{F}_\phi(c, q, r) \,, \; \lambda^\star = \mathcal{F}_\lambda(c, q, r)$$

is a primal-dual solution of (14). The complexity of constructing, storing, and evaluating these functions is of order $\mathcal{O}(M)$, see Bertsekas (2012b).

[4] Problem (14) is an equality constrained QP, whose definition is analogous to the coupled QP (3.3) in (Houska et al., 2016, pp. 7).

3.5 Implementation of parallel explicit MPC

Algorithm 1 proposes a parallel explicit MPC scheme. Notice that this algorithm is entirely based on explicit solution maps: the functions $\mathcal{G}_\mathrm{a}, \mathcal{G}_\mathrm{b}, u_0^\star$ are piecewise affine, independent of the prediction horizon N, and can be precomputed offline by using parametric quadratic programming (see Section 2). Their evaluation and storage cost is of order $\mathcal{O}(N_\mathrm{R})$ where N_R is the number of critical regions. Notice that the state and control constraints are in this notation hidden in the definiton of the generalized gradient maps. These constraints affect the complexity of the corresponding PWA representation. The evaluation and storage cost of the affine functions $\mathcal{F}_\xi, \mathcal{F}_\phi$ and $\mathcal{F}_\lambda$ is of order $\mathcal{O}(M)$.

Thus, the method can be scaled up easily for long horizons $N = Mn$. Algorithm 1 can be initialized with $\Theta = 0$, but this is only affecting the first step. In online mode, Θ is initialized with the solution of the previous MPC step. Optionally, Θ can be shifted in Step 3 after sending the control to the process in order to take into account that the horizon is moving. Notice that the inner loop in Step 2 is equivalent to ALADIN applied to (7), which has been reviewed in the introduction of this paper. The only difference of Algorithm 1 compared to the ALADIN

Algorithm 1 Parallel Explicit MPC using ALADIN

Offline:

- Use parametric quadratic programming in order to find explicit expressions for the piecewise affine gradient maps $\mathcal{G}_\mathrm{a}$ and $\mathcal{G}_\mathrm{b}$. Also store the piecewise affine function $u_0^\star$.
- Use standard LQR theory to precompute the affine solution maps $\mathcal{F}_\xi, \mathcal{F}_\phi$ and $\mathcal{F}_\lambda$.

Initialization:

Initial guesses $\Theta^0, \Theta^1, \ldots, \Theta^M \in \mathbb{R}^{n_x \times 4}$.

Online:

(1) Wait for the next state estimate $\hat{x}_0$ and set $\Theta_3^0 = \hat{x}_0$.

(2) **For** $m = 1, \ldots, m_{\max}$**:**

(a) Evaluate the explicit gradient maps

$$\forall k \in \{0, \ldots, M\}, \quad \left\{ \begin{array}{l} q_k = \mathcal{G}_\mathrm{a}(\Theta^k) \\ r_k = \mathcal{G}_\mathrm{b}(\Theta^k) \end{array} \right\} .$$

(b) Set $c_k = \Theta_3^{k+1} - \Theta_4^k$ for all $k \in \{0, \ldots, M-1\}$.

(c) Set

$$\begin{aligned} \Theta_1 &\leftarrow \mathcal{F}_\lambda(c, q, r), \\ \Theta_2^k &\leftarrow \Theta_1^{k+1}, \;\; k \in \{0, \ldots, M-1\}, \\ \Theta_3 &\leftarrow \Theta_3 + \mathcal{F}_\xi(c, q, r), \\ \Theta_4 &\leftarrow \Theta_4 + \mathcal{F}_\phi(c, q, r) \end{aligned}$$

and $m \leftarrow m + 1$.

End.

(3) Send $u_0^\star(\Theta_3^0, \Theta_4^0)$ to the real process and go to Step 1.

version, which has been proposed in Houska et al. (2016), is that the decoupled NLPs as well as the coupled QP are solved offline and replaced by explicit solution maps. This implies in particular that the convergence conditions for ALADIN from Houska et al. (2016) can be applied one-to-one in order to establish convergence and optimality of Algorithm 1 for large $m_{\max}$. In other words, $u_0^\star(\Theta_3^0, \Theta_4^0)$ is a numerically accurate approximation of the associated optimal control input of the original MPC problem (7). Thus, the online routine of Algorithm 1 can be implemented by using static memory only. Moreover, for fixed $m_{\max}$ the run-time of the online-loop can be verified offline, as it remains constant while running the algorithm. In practice, one can choose a small $m_{\max}$; in our case study we even choose $m_{\max} = 1$. Unfortunately, for this particularly agressive choice of $m_{\max}$ neither convergence, nor stability, nor feasibilty guarantees are available yet, but the empirical observation is that Algorithm 1 operates near optimal and is (independent of this initialization) surprisingly stable if we choose $m_{\max} = 1$. A deeper analysis of this observation will be part of future work.

4. NUMERICAL EXAMPLE

In order to illustrate the performance of Algorithm 1, this paper considers a double integrator system of the form

$$x_{k+1} = \begin{bmatrix} 1.0 & 1.0 \\ 0 & 1.0 \end{bmatrix} x_k + \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix} u_k. \tag{15}$$

Input and state constraints are given by

$$\mathbb{X} = \left\{ x \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} -1000 \\ -5 \end{bmatrix} \preceq x \preceq \begin{bmatrix} 1000 \\ 5 \end{bmatrix} \right\} \tag{16}$$
$$\text{and} \quad \mathbb{U} = \{ u \in \mathbb{R} \mid -2.5 \leq u \leq 1 \} .$$

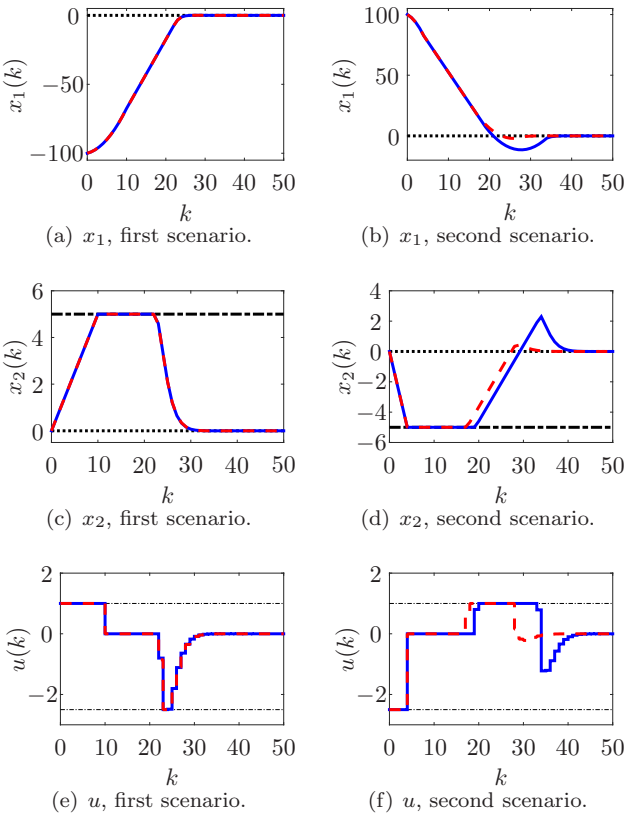
The weighting matrices of the MPC setup are given by

$$P = \begin{bmatrix} 1.8085 & 0.2310 \\ 0.2310 & 2.6489 \end{bmatrix}, \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 1, \tag{17}$$

where P has been found by solving an algebraic Riccati equation that approximates the infinite horizon cost-to-go function, as suggested in Rawlings and Mayne (2009).

Algorithm 1 has been implemented in Matlab R2013a using YALMIP (Löfberg, 2004) and MPT 3.1.5 (Herceg et al., 2013). First, the generalized gradient maps $\mathcal{G}_{\mathrm{a}}$ and $\mathcal{G}_{\mathrm{b}}$, as well as the primal optimizer $u_0^\star$ were pre-computed using the geometric parametric LCP solver. As this example has $n_{\mathrm{x}} = 2$ states, all parametric representations are operating in an $4n_{\mathrm{x}} = 8$ dimensional space. In particular, the complete parametric solution for $m_{\max} = 1$ consisted of 51 critical regions that, together with the associated local affine functions in (2) require 10 629 double-precision floating point numbers that approximately correspond to 85 kilobytes for their storage. This parametric solution was obtained in 3.6 seconds on a 1.7 GHz machine with 8 GB of memory.

To put the complexity of the parametric representations of generalized gradient maps $\mathcal{G}_{\mathrm{a}}$, $\mathcal{G}_{\mathrm{b}}$ and the primal optimizer $u_0^\star$ into perspective, we have also computed the full explicit solution by solving (7) for $N = 10, 20, \ldots, 100$ as a parametric QP using the parametric LCP solver contained



(a) x_1, first scenario. (b) x_1, second scenario.
(c) x_2, first scenario. (d) x_2, second scenario.
(e) u, first scenario. (f) u, second scenario.

Fig. 1. Closed-loop control performance of parallel MPC (solid blue) vs traditional MPC (dashed red) represented by (7) for $N = 10$. The dotted line denotes the regulation objective (the origin), dash-dotted lines denote constraints.

Table 1. "Standard" explicit MPC complexity.

N	off-line runtime [secs]	number of regions [−]	memory footprint [kB]
10	8	347	42
20	36	1071	129
50	309	3593	432
100	1838	7793	936

in the MPT toolbox. We remark that the explicit optimizer in this case was defined over n_{x}-dimensional Euclidian space with $n_{\mathrm{x}} = 2$, while the explicit solution maps employed in Algorithm 1 are in 8D. Despite this difference, the proposed parallel scheme outperforms the "standard" explicit MPC in two key measures. First, the off-line construction runtime of the proposed parallel scheme is up to three orders magnitude faster compared to traditional explicit MPC, see the second column in Table 1. Secondly, and more importantly, the size of the explicit solution maps employed by Algorithm 1 is up to one order of magnitude smaller compared to traditional explicit MPC.

To assess the performance of Algorithm 1 we have performed closed-loop simulations for two initial conditions using the prediction horizon $N = 10$. For $x_0 = [-100\ 0]^\top$, the difference to the optimal closed-loop trajectory is negligible, cf. Figs. 1(a), 1(c), 1(e). For $x_0 = [100\ 0]^\top$, Algorithm 1 generates suboptimal trajectories that are depicted in Figs. 1(b), 1(d), 1(f). In both cases, however, the feedback controller represented by Algorithm 1

achieved the regulation goal of steering all states to the origin. The suboptimality is induced by using $m_{\max} = 1$ in Algorithm 1, i.e., by performing just a single iteration at each time step. We remark that the suboptimality can be reduced by using $m_{\max} > 1$ as discussed in Section 3.5. The implications of suboptimality on stability and feasibility will be part of future work.

5. CONCLUSION

This paper has presented a novel parallel explicit MPC scheme, which is based on the distributed optimization algorithm ALADIN. The implementation of the online algorithm is solely based on piecewise affine explicit solution maps that can be pre-computed offline by using parametric quadratic programming tools. The main advantage of the proposed scheme compared to existing explicit MPC solvers is its significantly reduced and real-time verifyable online run-time as well as a much reduced memory footprint that does scale-up linearly with the prediction horizon of the MPC controller. Numerical examples for a double integrator example indicate a promising performance. More complex case studies will be part of future work.

ACKNOWLEDGEMENTS

All authors were supported via a bilateral grant agreement between China and Slovakia (SK-CN-2015-PROJECT-6558, APVV SK-CN-2015-0016), which is gratefully acknowledged. Additionally, the work of Yuning Jiang and Boris Houska was supported by the National Science Foundation China (NSFC), Nr. 61473185, as well as ShanghaiTech University, Grant-Nr. F-0203-14-012. J. Oravec and M. Kvasnica also gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grant 1/0403/15 and the Slovak Research and Development Agency under the project APVV-15-0007.

REFERENCES

Alessio, A. and Bemporad, A. (2009). A survey on explicit model predictive control. In *Nonlinear model predictive control*, 345–369.

Baotic, M., Borrelli, F., Bemporad, A., and Morari, M. (2008). Efficient On-Line Computation of Constrained Optimal Control. *SIAM Journal on Control and Optimization*, 47(5), 2470–2489.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1), 3–20.

Bertsekas, D.P. (2012a). *Constrained Optimization and Lagrange Multiplier Methods*, volume II. Academic Press, Belmont, Massachusetts, 2nd edition.

Bertsekas, D.P. (2012b). *Dynamic Programming and Optimal Control*. Athena Scientific Dynamic Programming and Optimal Control, Belmont, Massachusetts, 3rd edition.

Borrelli, F., Bemporad, A., and Morari, M. (2011). Predictive control for linear and hybrid systems. *Cambridge February*.

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

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1), 1–122.

Cagienard, R., Grieder, P., Kerrigan, E., and Morari, M. (2007). Move Blocking Strategies in Receding Horizon Control. *Journal of Process Control*, 17(6), 563–570.

Drgoňa, J., Klaučo, M., Janeček, F., and Kvasnica, M. (2016). Optimal control of a laboratory binary distillation column via regionless explicit MPC. *Computers & Chemical Engineering*.

Everett, H. (1963). Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3), 399–417.

Giselsson, P., Doan, M.D., Keviczky, T., Schutter, B.D., and Rantzer, A. (2013). Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3), 49(3).

Grieder, P., Kvasnica, M., Baotić, M., and Morari, M. (2005). Stabilizing low complexity feedback control of constrained piecewise affine system. *Automatica*, 41, 1683–1694.

Herceg, M., Kvasnica, M., Jones, C., and Morari, M. (2013). Multi-parametric toolbox 3.0. In *2013 European Control Conference*, 502–510.

Hestenes, M. (1969). Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4, 302–320.

Holaza, J., Takács, B., Kvasnica, M., and Di Cairano, S. (2015). Nearly optimal simple explicit MPC controllers with stability and feasibility guarantees. *Optimal Control Applications and Methods*, 35(6).

Houska, B., Frasch, J., and Diehl, M. (2016). An augmented lagrangian based algorithm for distributed non-convex optimization. *SIAM Journal on Optimization*, 26(2), 1101–1127.

Kouzoupis, D., Quirynen, R., Houska, B., and Diehl, M. (2016). A block based aladin scheme for highly parallelizable direct optimal control. In *Proceedings of the 2016 American Control Conference, 2016*, 1124–1129. Bostion, USA.

Löfberg, J. (2004). YALMIP. Available from `http://users.isy.liu.se/johanl/yalmip/`.

Mota, J., Xavier, J., Aguiar, P., and Püschel, M. (2012). Distributed admm for model predictive control and congestion control. In *Proceedings of the 51st IEEE Conference on Decision and Control, 2012*, 5110–5115.

Necoara, I. and Suykens, J. (2008). Application of a smoothing technique to decomposition in convex optimization. *IEEE Transactions on Automatic Control*, 53(11), 2674–2679.

Oberdieck, R., Diangelakis, N., Nascu, I., Papathanasiou, M., Sun, M., Avraamidou, S., and Pistikopoulos, E. (2016). On multi-parametric programming and its applications in process systems engineering. *Chemical Engineering Research and Design*.

O'Donoghue, B., Stathopoulos, G., and Boyd, S. (2013). A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 21(6), 2432–2442.

Powell, M. (1969). A method for nonlinear constraints in minimization problems. *In Optimization, (R. Fletcher, ed.), Academic Press*.

Pu, Y., Zeilinger, M., and Jones, C. (2014). Inexact fast alternating minimization algorithm for distributed model predictive control. In *Proceedings of the 53rd IEEE Conferenceon Decision and Control(CDC), 2014*. Los Angeles, USA.

Rawlings, J. and Mayne, D. (2009). *Model Predictive Control: Theory and Design*. Madison, WI: Nob Hill Publishing.

Rockafellar, R. (1970). *Convex Analysis*. Princeton University Press.

Tapia, R. (1978). Quasi-newton methods for equality constrained optimization: Equivalence of existing methods and a new implementation. In *In Nonlinear Programming 3, O. Mangasarian, R. Meyer, S. Robinson, eds, Academic Press*, 125–164. New York, NY.

Tøndel, P., Johansen, T., and Bemporad, A. (2003). Evaluation of piecewise affine control via binary search tree. *Automatica*, 39(5), 945–950.

Trnka, P., Havlena, V., and Pekar, J. (2016). Distributed MPC with parametric coordination. In *Proceedings of American Control Conference*, 6253–6258.

Zanarini, A., Jafargholi, M., and Peyrl, H. (2013). Exploiting parallelization in explicit model predictive control. In *Proceedings of International Conference on Information, Communication and Automation Technologies*, 1–7.