Parallel Explicit Model Predictive Control

Yuning Jiang\textsuperscript{a,b,d}, Juraj Oravec\textsuperscript{c}, Boris Houska\textsuperscript{a}, Michal Kvasnica\textsuperscript{c}

\textsuperscript{a}School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China.
\textsuperscript{b}Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China
\textsuperscript{c}Institute of Information Engineering, Automation, and Mathematics, Faculty of Chemical and Food Technology, Slovak University of Technology in Bratislava, Radlinského 9, SK812-37 Bratislava, Slovak Republic.
\textsuperscript{d}University of Chinese Academy of Sciences, Beijing 100049, China

Abstract

This paper is about a real-time model predictive control (MPC) algorithm for large-scale, structured linear systems with polytopic state and control constraints. The proposed controller receives the current state measurement as an input and computes a sub-optimal control reaction by evaluating a finite number of piecewise affine functions that correspond to the explicit solution maps of small-scale parametric quadratic programming (QP) problems. We provide recursive feasibility and asymptotic stability guarantees, which can both be verified offline. The feedback controller is suboptimal on purpose because we are enforcing real-time requirements assuming that it is impossible to solve the given large-scale QP in the given amount of time. In this context, a key contribution of this paper is that we provide a bound on the sub-optimality of the controller. Our numerical simulations illustrate that the proposed explicit real-time scheme easily scales up to systems with hundreds of states and long control horizons, system sizes that are completely out of the scope of existing, non-suboptimal Explicit MPC controllers.

Key words: model predictive control, parametric optimization, fixed memory utilization.

1 Introduction

The advances of numerical optimization methods over the last decades \cite{13}, in particular, the development of efficient quadratic programming problem (QP) solvers \cite{16}, have enabled numerous industrial applications of MPC \cite{34}. Modern real-time optimization and control software packages \cite{22,29} achieve run-times in the milli- and microsecond range by generating efficient and reliable C-code that implements problem-tailored MPC algorithms \cite{12,40}. However, as much as these algorithms and codes perform well on desktop computers or other devices with comparable computation power, the number of successful implementations of MPC on embedded industrial hardware such as programmable logic controllers (PLC) and field-programmable gate arrays (FPGA) remains limited \cite{24}. Here, the main question is what can be done if an embedded device has simply not enough computational power or storage space to solve the exact MPC problem online and in real-time.

Many researchers have attempted to address this question. For example, the development of Explicit MPC \cite{31} aims at reducing both the online run-time and the memory footprint of MPC by optimizing pre-computed solution maps of multi-parametric optimization problems. However, Explicit MPC has the disadvantage that the number of polytopic regions over which the piecewise affine solution map of a parametric linear or quadratic programming problem is defined, grows, in the worst case, exponentially with the number of constraints. Some authors \cite{18} have suggested addressing this issue by simplifying the MPC problem formulation, e.g., by using move-blocking \cite{9}, but the associated control reactions can be sub-optimal by a large margin. Other authors \cite{27} have worked on reducing the memory footprint of Explicit MPC—certainly making considerable progress yet
failing to meet the requirement of many practical systems with more than just a few states. In fact, despite all these developments in Explicit MPC, these methods are often applicable to problems of modest size only. As soon as one attempts to scale up to larger systems, an empirical observation is that Explicit MPC is often outperformed by iterative online solvers such as active set [16] or interior point methods for MPC [29]. In this context, we mention that [25] has recently proposed a heuristic for reducing the number of regions of Explicit MPC by using neural-network approximations. The corresponding controller does, however, only come along with guarantees on the feasibility, stability, and performance if the exact explicit solution map happens to be recovered by the deep-learning approach.

A recent trend in optimization based control is to solve large MPC problems by breaking them into smaller ones. This trend has been initiated by the enormous research effort in the field of distributed optimization [7]. There exists a large number of related optimization methods, including dual decomposition [14], ADMM [7], or ALADIN [23], which have all been applied to MPC in various contexts and by many authors [11,17,26,30,32,36]. Additionally, applications of accelerated variants of ADMM to MPC can be found in [15,37].

However, modern distributed optimization methods, such as ADMM or ALADIN, typically converge to an optimal solution in the limit, i.e., if the number of iterations tends to infinity. Thus, if real-time constraints are present, one could at most implement a finite number of such ADMM or ALADIN iterations returning a control input that may be infeasible or sub-optimal by a large margin. But, unfortunately, for such heuristic implementations of real-time distributed MPC, there are, at the current status of research, no stability, feasibility, and performance guarantees available.

Therefore, this paper asks the question whether it is possible to approximate an MPC feedback law by a finite code list, whose input is the current state measurement and whose output, the control reaction, is obtained by evaluating a constant, finite number of pre-computed, explicit solution maps that are associated to MPC problems of a smaller scale. Here, a key requirement is that recursive feasibility, uniform asymptotic stability, and performance guarantees of the implemented closed-loop controller have to be verifiable offline. Notice that such an MPC code would have major advantages for an embedded hardware system, as it has a constant run-time using static memory only, while, at the same time, feasibility, stability, and performance guarantees are available.

The contribution of this paper is the development of a controller, which meets these requirements under the restricting assumption that the original MPC problem is a strongly convex (but potentially large-scale) QP, as introduced in Section 2. The control scheme itself is presented in the form of Algorithm 1 in Section 3. This algorithm alternates—similar to the distributed optimization method ALADIN [23]—between solving explicit solution maps that are associated with small-scale decoupled QPs and solving a linear equation system of a larger scale. However, in contrast to ALADIN, ADMM or other existing distributed optimization algorithms, Algorithm 1 performs only a constant number of iterations per sampling time.

The recursive feasibility, stability, and performance properties of Algorithm 1, which represent the main added value compared to our preliminary work [33], are summarized in Section 3.3, 3.4, and 3.5, respectively. Instead of relying on existing analysis concepts from the field of distributed optimization, the mathematical developments in this paper rely on results that find their origin in Explicit MPC theory [4]. In particular, the technical developments around Theorem 1 make use of the solution properties of multi-parametric QPs in order to derive convergence rate estimates for Algorithm 1. Moreover, Theorem 2 establishes an asymptotic stability guarantee of the presented real-time closed-loop scheme. This result is complemented by Corollary 1, which provides bounds on the sub-optimality of the presented control scheme.

Finally, Section 4 discusses implementation details with a particular emphasis on computational and storage complexity exploiting the fact that the presented scheme can be realized by using static memory only while ensuring a constant run-time. A spring-vehicle-damper benchmark is used to illustrate the performance of the proposed real-time scheme. Section 6 concludes the paper.

## 2  Linear-Quadratic MPC

This paper concerns differential-algebraic model predictive control problems in discrete-time form,

\[
J(x_0) = \min_{x,u,z} \sum_{k=0}^{N-1} \ell(x_k,u_k,z_k) + \mathcal{M}(x_N) \\
\text{s.t. } \forall k \in \{0,\ldots,N-1\},
\]

\[
x_{k+1} = Ax_k + Bu_k + Cz_k,
\]

\[
0 = Dx_k + Ez_k,
\]

\[
x_k \in \mathcal{X}, \ u_k \in \mathcal{U}, \ z_k \in \mathcal{Z}, \ x_N \in \mathcal{X}_N,
\]

with strictly convex quadratic stage and terminal cost,

\[
\ell(x,u,z) = x^\top Q x + u^\top R u + z^\top S z
\]

and \(\mathcal{M}(x) = x^\top P x\).
Here, \( x_k \in \mathbb{R}^{n_x} \) denotes the predicted state at time \( k \), \( z_k \in \mathbb{R}^{n_z} \) an associated algebraic state, and \( u_k \in \mathbb{R}^{n_u} \) the associated control input. The matrices

\[
A, P, Q \in \mathbb{R}^{n_x \times n_x}, \quad B \in \mathbb{R}^{n_x \times n_u}, \quad C \in \mathbb{R}^{n_x \times n_z}, \\
D \in \mathbb{R}^{n_n \times n_x}, \quad E, S \in \mathbb{R}^{n_n \times n_n}, \quad R \in \mathbb{R}^{n_u \times n_u}
\]

are given and constant. Notice that (1) is a parametric optimization problem with respect to the current state measurement \( x_0 \), i.e., the optimization variable \( x = [x_1, x_2, \ldots, x_N] \) includes all but the first element of the state sequence. The control sequence \( u = [u_0, u_1, \ldots, u_{N-1}] \) and algebraic state sequence \( z = [z_0, z_1, \ldots, z_N] \) are defined accordingly.

**Assumption 1** We assume that

a) the constraint sets \( \mathbb{U} \subseteq \mathbb{R}^{n_u}, \mathbb{X}, \mathbb{X}_N \subseteq \mathbb{R}^{n_x}, \) and \( \mathbb{Z} \subseteq \mathbb{R}^{n_z} \) are convex and closed polyhedra satisfying \( 0 \in \mathbb{U}, 0 \in \mathbb{X}, 0 \in \mathbb{X}_N \), and \( 0 \in \mathbb{Z} \);
b) the matrices \( Q, R, S, \) and \( P \) are all symmetric and positive definite;
c) and the square-matrix \( E \) is invertible.

Notice that the latter assumption implies that, at least in principle, one could eliminate the algebraic states, as the algebraic constraints in (1) imply

\[
z_k = -E^{-1} Dx_k .
\]

However, the following algorithmic developments exploit the particular structure of (1). This is relevant in the context of large-scale interconnected control systems, where the matrix \( D \) is potentially dense while all other matrices have a block-diagonal structure, as explained in the sections below.

**Remark 1** Assumption 1c) also implies that the equality constraints in (1) are linearly independent. Thus, the associated co-states (dual solutions) are unique. Similarly, Assumption 1a) and 1b) imply strong convexity such that the primal solution of (1) is unique whenever it exists.

**Remark 2** The assumption that the matrices \( Q, R, S, \) and \( P \) are positive definite can be satisfied in practice by adding suitable positive definite regularization to the stage cost.

### 2.1 Interconnected systems

Many control systems of practical interest have a particular structure, as they can be divided into \( I \in \mathbb{N} \) subsystems that are interconnected. A prototypical example for such an interconnected system is shown in Figure 1, which consists of \( I \) vehicles with mass \( m \) that are linked by springs and dampers. Such systems can be modelled by a linear discrete-time recursion of the form

\[
[x_{k+1}]_i = A_{i,i} [x_k]_i + B_i [u_k]_i + \sum_{j \in \mathcal{N}_i} A_{ij} [x_k]_j 
\]

for all \( k \in \{1, \ldots, N\} \). Here, \( [x_k]_i \in \mathbb{X}_i \) denotes the state, \( [u_k]_i \in \mathbb{U}_i \) the control input of the \( i \)-th subsystem, and \( A_{ij} \) and \( B_i \) system matrices of appropriate dimension. The index \( j \) runs over \( \mathcal{N}_i \), the set of neighbors of the \( i \)-th node in the system’s connectivity graph.

**Example 1** For the spring-vehicle-damper system in Figure 1 the connectivity graph can be specified by setting

\[
\mathcal{N}_i = \{i-1, i+1\}
\]

for \( i \in \{2, 3, \ldots, I-1\} \) and \( \mathcal{N}_1 = \{2\} \) and \( \mathcal{N}_I \) for the first and last trolley. In this example, the \( i \)-th subblock of the state at time \( k \),

\[
[x_k]_i = ([p_k]_i, [v_k]_i)^T,
\]

consists of the position and velocity of the \( i \)-th trolley, all relative to their equilibrium values. The control input is the force at the last trolley. A corresponding system model is then obtained by setting

\[
A_{i,i} = I + h \begin{pmatrix} 0 & 1 \\ -2k/m & -2d/m \end{pmatrix}, \quad B_i = \begin{pmatrix} 0 \\ 0 \end{pmatrix},
\]

\[
A_{i-1,i} = A_{i,i+1} = h \begin{pmatrix} 0 & 0 \\ k/m & d/m \end{pmatrix},
\]

for all \( i \in \{1, \ldots, I-1\} \). In this context, \( m > 0 \) denotes the mass, \( k > 0 \) the spring constant, \( d \geq 0 \) a damping coefficient, and \( h > 0 \) the step-size of the Euler discretization.

System (3) can be reformulated by introducing auxiliary variables of the form

\[
[z_k]_i = \sum_{j \in \mathcal{N}_i} A_{ij} [x_k]_j .
\]
The advantage of introducing these algebraic auxiliary variables is that the dynamic recursion (3) can be written in the form

\[
\begin{align*}
x_{k+1} &= Ax_k + Bu_k + Cz_k, \\
0 &= Dx_k + Ez_k,
\end{align*}
\]

with \( A = \text{diag} \left( A_{1,1}, \ldots, A_{I,I} \right) \), \( B = \text{diag} \left( B_1, \ldots, B_I \right) \), \( C = I \),

\[
D_{i,j} = \begin{cases} 
A_{i,j} & \text{if } j \in \mathcal{N}_i \\
0 & \text{otherwise,}
\end{cases}
\]

and \( E = -I \). After this reformulation, all matrices in the algebraic discrete-time system are block diagonal except for the matrix \( D \), which may, however, still be sparse depending on the particular definition (graph structure) of the sets \( \mathcal{N}_i \). Notice that the associated state and control constraint sets \( X = X_1 \times \ldots \times X_I \) and \( U = U_1 \times \ldots \times U_I \) have a separable structure, too.

2.2 Recursive feasibility and asymptotic stability

Notice that the stability and recursive feasibility properties of MPC controllers have been analyzed exhaustively [35]. As long as Assumption 1 holds, these results can be applied one-to-one for (1) after eliminating the algebraic states explicitly.

**Definition 1** A set \( X \subseteq \mathbb{X} \) is called control invariant if

\[
X \subseteq \left\{ x \in \mathbb{R}^n_x \mid \exists x^+ \in \mathbb{X}, \exists u \in \mathbb{U}, \exists z \in \mathbb{Z}, \quad x^+ = Ax + Bu + Cz, \quad 0 = Dx + Ez \right\}.
\]

We assume that the terminal set \( \mathbb{X}_N \) is control invariant.

It is well known that (1) is recursively feasible if Assumption 1 and Assumption 2 hold recalling that one can always eliminate the algebraic states in order to apply the results from [35]. Another standard assumption can be formulated as follows.

**Assumption 2** We assume that the terminal set \( \mathbb{X}_N \) is control invariant.

Moreover, tailored methods for constructing block separable terminal costs and terminal constraint sets are available in the literature, too [10].

3 Suboptimal real-time MPC

In this section we propose and analyze a real-time algorithm for finding approximate, suboptimal solutions of (1).

3.1 Preliminaries

Let us introduce the stacked vectors

\[
y_0 = [u_0^T, z_0^T]^T, \quad y_k = [x_k^T, u_k^T, z_k^T]^T, \quad y_N = x_N,
\]

and their associated constraint sets

\[
\mathbb{Y}_k = \left\{ y_k \left| \begin{array}{l}
Ax_k + Bu_k + Cz_k \in \mathbb{X}, \\
0 = Dx_k + Ez_k,
\end{array} \right. \right\}, \quad \mathbb{Y}_N = \mathbb{X}_N,
\]

for all \( k \in \{0, \ldots, N-1\} \). Moreover, it is convenient to introduce the shorthand notation

\[
F_k(y_k) = \ell(x_k, u_k, z_k), \quad k \in \{0, \ldots, N-1\}
\]

as well as the matrices

\[
G_k = \begin{pmatrix} I & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad G_N = I,
\]

for \( k \in \{1, \ldots, N-1\} \) and

\[
H_0 = \begin{pmatrix} B & C \\ 0 & E \end{pmatrix}, \quad H_k = \begin{pmatrix} A & B & C \\ \_ & D & 0 & E \end{pmatrix},
\]

as well as \( h_0 = [(Ax_0^T 0)^T, h_k = \mathbf{0} \) for all indices \( k \in \{1, \ldots, N-1\} \). The advantage of this notation is that (1) can be written in the form

\[
J(x_0) = \min_y \sum_{k=0}^N F_k(y_k) \quad \text{subject to} \quad \begin{cases}
\forall k \in \{0, \ldots, N-1\}, \\
g_{k+1} y_{k+1} = H_k y_k + h_k \mid \lambda_k, \\
y_k \in \mathbb{Y}_k, \quad y_N \in \mathbb{Y}_N.
\end{cases}
\]

Here, \( \lambda_0, \ldots, \lambda_{N-1} \) denote the multipliers of the affine constraints in (7). It is helpful to keep in mind that both
the function $F_0$ and the vector $h_0$ depend on $x_0$ even if we hide this parametric dependence for a moment in order to arrive at a simpler notation. In addition, we introduce a shorthand for the objective in (7),

$$F(y) = \sum_{k=0}^{N} F_k(y_k).$$

Moreover, the convex conjugate function of $F$ is denoted by

$$F^*(\lambda) = \max_y \left\{ -F(y) - (H^T \lambda_k)^T y_0 + \sum_{k=1}^{N-1} \left( G_k^T \lambda_k - H_k^T \lambda_k \right)^T y_k + \lambda_k N^y_0 G_N y_N \right\}.$$ 

Notice that the functions $F$ and $F^*$ are strongly convex quadratic forms with $F(0) = 0$ and $F^*(0) = 0$ as long as Assumption 1 is satisfied. The optimal primal and dual solutions of (7) are denoted by $x^*$ and $\lambda^*$, respectively. It is well-known that $x^*$ and $\lambda^*$ are continuous and piecewise affine functions of $x_0$ on the polytopic domain $\mathcal{X} = \{x_0 | J(x_0) < \infty\}$, see [5].

### 3.2 Algorithm

The main idea for solving (7) approximately and in real time is to consider the unconstrained auxiliary optimization problem

$$J(x_0) = \min_y \sum_{k=1}^{N-1} F_k(y_k - y^\text{ref}_k) + F_N(y_N - y^\text{ref}_N) \quad (8)$$

subject to

$$\forall k \in \{0, \ldots, N-1\}, \quad G_k y_k = H_k y_k + h_k \quad | \lambda_k,$$

which is a standard tracking optimal control problem without inequality constraints. Here, we have introduced the reference trajectory $y^\text{ref}_k$. For the special case that $y^\text{ref} = y^*$ is equal to the minimizer of (7), Problems (7) and (8) are equivalent. Notice that the main motivation for introducing the coupled QP (8) is that this problem approximates (7) without needing inequality constraints. Thus, this problem can be solved by using a sparse linear algebra solver; see Section 4.2 for implementation details.

Let us assume that $y^m$ and $\lambda^m$ are the current approximations of the primal and dual solution of (7). Now, the main idea of Algorithm 1 is to construct the next iterate $y^{m+1}$ and $\lambda^{m+1}$ by performing two main operations. First, we solve augmented Lagrangian optimiza-

### Algorithm 1 Parallel real-time MPC

**Initialization:**

Choose initial $y^1 = [y_0^1, \ldots, y_N^1]$ and $\lambda^1 = [\lambda_0^1, \ldots, \lambda_{N-1}^1]$ and a constant $\gamma > 0$.

**Online:**

1. Wait for the state measurement $x_0$ and compute the constant

$$f^1 = F(y^1) + F^*(\lambda^1).$$

If $f^1 \geq \gamma^2 x_0^T Q x_0$, rescale

$$y^1 \leftarrow y^1 \sqrt{\gamma^2 \|x_0\|_Q^2} \quad \text{and} \quad \lambda^1 \leftarrow \lambda^1 \sqrt{\gamma^2 \|x_0\|_Q^2},$$

where $\|x_0\|_Q^2 \triangleq x_0^T Q x_0$.

2. For $m = 1 \rightarrow m$

   a) solve the small-scale decoupled QPs in parallel

   $$\min_{\xi \in \mathcal{X}} F_0(\xi^m_0) - (H_0^T \lambda^m_0)^T \xi^m_0 + F_0(\xi^m_0 - y^m_0)$$

   $$\min_{\xi \in \mathcal{X}} F_k(\xi^m_k) + (G_k^T \lambda^m_{k-1} - H_k^T \lambda^m_k)^T \xi^m_k + F_k(\xi^m_k - y^m_k)$$

   $$\min_{\xi \in \mathcal{X}} F_N(\xi^m_N) + (G_N^T \lambda^m_{N-1})^T \xi^m_N + F_N(\xi^m_N - y^m_N)$$

   for all $k \in \{1, \ldots, N-1\}$ and denote the optimal solutions by $\xi^m = [\xi^m_0, \xi^m_1, \ldots, \xi^m_N]$.

   b) Solve the coupled QP

   $$\min_{y} \sum_{k=0}^{N} F_k(y_k^{m+1} - 2\xi_k^m + y_k^m) \quad (9)$$

   subject to

   $$\forall k \in \{0, \ldots, N-1\}, \quad G_k y_k^{m+1} = H_k y_k^{m+1} + h_k \quad | \delta_k^m,$$

   and set $\lambda^{m+1} = \lambda^m + \delta^m$.

**End**

3. Send $u_0 = (0 \ 1 \ 0) \xi_0^m$ to the real process.

4. Set $y^1 = [y_0^m, \ldots, y_N^m, 0]$, $\lambda^1 = [\lambda_0^m, \ldots, \lambda_{N-1}^m, 0]$ and go to Step 1.
with
\[
G = \begin{pmatrix}
H_0 - G_0 & 0 \\
H_1 - G_1 & \ddots & \ddots \\
\vdots & \ddots & \ddots \\
0 & \cdots & H_{N-1} - G_{N-1}
\end{pmatrix}.
\]

The focus of the following analysis is on the case that the weighting matrix \( Q = \nabla^2 F(0) \) is such that
\[
\|\xi^m - y^m\|^2_Q = F(\xi^m - y^m)
\]
recalling that \( F \) is a centered positive-definite quadratic form. And second, we solve QP (8) for the reference point
\[
y^{\text{ref}} = 2\xi^m - y^m,
\]
which can be interpreted as weighted average of the solution of (10) and the previous iterate \( y^m \). These two main steps correspond to Step 2a) and Step 2b) in Algorithm 1. Notice that the main motivation for introducing the augmented Lagrangian problem (10) is that this optimization problem is separable, as exploited by Step 2a). As explained in more detail in Section 4.1, the associated smaller-scale QPs can be solved by using existing tools from the field of Explicit MPC.

Additionally, in order to arrive at a practical procedure, Algorithm 1 is initialized with guesses,
\[
y^1 = [y^1_0, \ldots, y^1_N] \quad \text{and} \quad \lambda^1 = [\lambda^1_0, \ldots, \lambda^1_N],
\]
for the primal and dual solution of (7). Notice that Algorithm 1 receives a state measurement \( x_0 \) in every iteration (Step 1) and returns a control input to the real process (Step 3). Moreover, Step 1) rescales \( y^1 \) and \( \lambda^1 \) based on a tuning parameter \( \gamma > 0 \), which is assumed to have the following property.

**Assumption 4** The constant \( \gamma \) in Algorithm 1 is such that
\[
F(y^*) + F^*(\lambda^*) \leq \gamma^2 x_0^T Q x_0.
\]

Notice that such a constant \( \gamma \) exists and can be precomputed offline, which simply follows from the fact that \( y^* \) and \( \lambda^* \) are Lipschitz continuous and piecewise affine functions of \( x_0 \), as established in [5]. The rationale behind this rescaling is that this step prevents initializations that are too far away from 0. Intuitively, if the term \( f^1 = F(y^1) + F^*(\lambda^1) \) is much larger than \( x_0^T Q x_0 \), then \( (y^1, \lambda^1) \) can be expected to be far away from the optimal solution \((y^*, \lambda^*)\) and it is better to rescale these variables such that they have a reasonable order of magnitude. In the following section, we will provide a theoretical justification of the rescaling factor \( \sqrt{\frac{\|x_0\|^2_Q}{f^1}} \). Notice that

if we would set this rescaling factor to 1, Algorithm 1 is unstable in general. In order to see this, consider the scenario that a user initializes the algorithm with an arbitrary \((y^1, \lambda^1) \neq 0\). Now, if the first measurement happens to be at \( x_0 = 0 \), of course, the optimal control input is at \( u^* = 0 \). But, if we run Algorithm 1 with \( \bar{m} < \infty \), it returns an approximation \( u_0 \approx u^* = 0 \), which will introduce an excitation as we have \( u_0 \neq 0 \) in general. Thus, if we would not rescale the initialization in Step 1), it would not be possible to establish stability.

### 3.3 Convergence properties of Algorithm 1

This section provides a concise overview of the theoretical convergence properties of Algorithm 1. We start with the following theorem, which is proven in Appendix A.

**Theorem 1** Let Assumption 1 be satisfied and let (7) be feasible, i.e., such that a unique minimizer \( y^* \) and an associated dual solution \( \lambda^* \) exist. Then there exists a positive constant \( \kappa < 1 \) such that
\[
F(y^{m+1} - y^*) + F^*(\lambda^{m+1} - \lambda^*) \leq \kappa (F(y^m - y^*) + F^*(\lambda^m - \lambda^*))
\]
for all \( m \geq 2 \).

Because Theorem 1 establishes contraction, an immediate consequence is that the iterates of Algorithm 1 would converge to the exact solution of (7), if we would set \( \bar{m} = \infty \), i.e.,
\[
\lim_{m \to \infty} \xi^m = y^* \quad \text{and} \quad \lim_{m \to \infty} \lambda^m = \lambda^*.
\]

Moreover, the proof of Theorem 1 provides an explicit procedure for computing the constant \( \kappa < 1 \).

### 3.4 Recursive feasibility and asymptotic stability of Algorithm 1

The goal of this section is to establish recursive feasibility and asymptotic stability of Algorithm 1 on \( X \). Because we send the control input \( u_0 = (0 \ I \ 0) \xi_0^m \) to the real process, the next measurement will be at
\[
x^+_0 = A x_0 + (B \ C) \xi_0^m.
\]

Notice that, in general, we may have
\[
x^+_0 \neq x^*_0 = A x_0 + (B \ C) y^*_0,
\]
since we run Algorithm 1 with a finite \( \bar{m} < \infty \). A proof of the following proposition can be found in Appendix B.

**Proposition 1** Let us assume that
Then Algorithm 1 is recursively feasible in the sense that $x_0 \in X$ implies $x_0^j \in X$. Moreover, the equation $X = X$ holds, i.e., Problem (7) remains feasible.

**Remark 3** The construction of control invariant sets for interconnected systems in the presence of state constraints can be a challenging task and this manuscript does not claim to resolve this problem. An in-depth discussion of how to meet the third requirement of Proposition 1 is beyond the scope of this paper, but we refer to [10], where a discussion of methods for the construction of control invariant sets for network systems can be found.

The following theorem establishes asymptotical stability of Algorithm 1, one of the main contributions of this paper. The corresponding proof can be found in Appendix C.

**Theorem 2** Let Assumptions 1, 2, 3, and 4 be satisfied and let $X_N = X$ be control invariant. Let the constant $\sigma > 0$ be such that the semi-definite inequality
\[
\begin{bmatrix}
B^\top QB & B^\top QC \\
C^\top QB & C^\top QB
\end{bmatrix} \preceq \sigma \begin{bmatrix}
R & 0 \\
0 & S
\end{bmatrix}
\]
holds and let the constants $\eta, \tau > 0$ be such that
\[
|J(x_0^+) - J(x_1^+)| \leq \eta \|x_0^+ - x_1^+\|_Q + \frac{\tau}{2} \|x_0^+ - x_1^+\|^2_Q
\]
If the constant $\overline{m} \in \mathbb{N}$ satisfies
\[
\overline{m} > 2 \frac{\log \left( \eta \sqrt{\sigma} \gamma (1 + \sqrt{\kappa}) + \frac{\tau \sigma \gamma^2 (1 + \sqrt{\kappa})^2}{2} \right)}{\log (1/\kappa)} ,
\]
then the controller in Algorithm 1 is asymptotically stable on $X$.

The constants $\eta, \tau, \sigma, \gamma$, and $\kappa < 1$ in the above theorem depend on the problem data only, but they are independent of the initial state $x_0 \in X$.

**Remark 4** The lower bound (13) is monotonically increasing in $\eta, \tau, \sigma, \gamma$, and $\kappa$. Thus, the smaller these constants are, the tighter the lower bound (13) will be. For small-scale applications, one can compute these constants offline, using methods from the field of approximate dynamic programming [39]. However, if one uses such conservative bounding methods, the lower bound (13) is conservative, too.

### 3.5 Performance of Algorithm 1

The result of Theorem 2 can be extended in order to derive an a-priori verifiable upper bound on the sub-optimality of Algorithm 1.

**Corollary 1** Let the assumption of Theorem 2 hold with
\[
\alpha = 1 - \left[ \eta \sqrt{\sigma} \gamma (1 + \sqrt{\kappa}) + \frac{\tau \sigma \gamma^2 (1 + \sqrt{\kappa})^2}{2} \right] \kappa^\frac{m}{2}
\]
If $y_i^c = (x_i^c, u_i^c, z_i^c)$ denotes the sequence of closed-loop states and controls that are generated by the controller in Algorithm 1, an a-priori bound on the associated infinite-horizon closed-loop performance is given by
\[
\sum_{i=0}^{\infty} \ell(x_i^c, u_i^c, z_i^c) \leq \frac{J(x_0)}{\alpha} .
\]

**Proof.** Theorem 2 together with (C.2) and (C.3) imply that
\[
J(x_i^c) \leq J(x_0) - \alpha F_0(y_i^c),
\]
which yields the inequality
\[
\sum_{i=0}^{\infty} F_0(y_i^c) \leq \frac{1}{\alpha} \sum_{i=0}^{\infty} (J(x_i^c) - J(x_{i+1}^c)) .
\]
The statement of the corollary follows after simplifying the telescoping sum on the right and substituting the equation $F_0(y_i^c) = \ell(x_i^c, u_i^c, z_i^c)$. \qed

### 4 Implementation on hardware with limited memory

This section discusses implementation details for Algorithm 1 with a particular emphasis on run-time aspects and limited memory requirements, as needed for the implementation of MPC on embedded hardware. Here, the implementation of Steps 1), 3), and 4) turns out to be straightforward, as both the CPU time requirements and memory needed for implementing these steps is negligible compared to Step 2). Thus, the focus of the following subsections is on the implementation of Step 2a) and Step 2b).

#### 4.1 Parametric quadratic programming

In Step 2a) of Algorithm 1 decoupled QPs have to be solved on-line. To diminish the induced implementation...
effort, we propose to solve these QPs off-line using multi-parametric programming, i.e., to pre-compute the solution maps

\[
\xi_0^*(\theta_0, x_0) = \arg\min_{\xi_0 \in \mathcal{V}_0} 2F_0(\xi_0) + \theta_0^T \xi_0, \\
\xi_1^*(\theta_1) = \arg\min_{\xi_1 \in \mathcal{V}_1} 2F_1(\xi_1) + \theta_1^T \xi_1, \\
\xi_N^*(\theta_N) = \arg\min_{\xi_N \in \mathcal{V}_N} 2F_N(\xi_N) + \theta_N^T \xi_N,
\]

(14)

with parameters \(\theta_0 \in \mathbb{R}^{n_u+n_x}, \theta_1 \in \mathbb{R}^{n_x+n_u+n_x}, \) and \(\theta_N \in \mathbb{R}^{n_x}.\) Because these QPs are strongly convex, the functions \(\xi_0^*, \xi_1^*, \) and \(\xi_N^*\) are piecewise affine [4]. Here, it should be noted that \(\xi_0^*\) additionally depends on \(x_0\) recalling that this dependency had been hidden in our definition of \(F_0\) and \(Y_0\). In this paper, we use MPT [21] to pre-compute and store the explicit solution maps \(\xi_0^*, \xi_1^*\) and \(\xi_N^*.\) Consequently, Step 2a) in Algorithm 1 can be replaced by:

**Step 2a’)** Compute the parameters

\[
\begin{align*}
\theta_0^m &= -H_0^T \lambda_0^m - 2\Sigma_0 y_0^m, \\
\theta_k^m &= G^T_k \lambda_{k-1}^m - H_k^T \lambda_k^m - 2\Sigma_k y_k^m, \\
\theta_N^m &= G^T_N \lambda_N^m - 2\Sigma_N y_N^m
\end{align*}
\]

(15a-c)

with \(\Sigma_0 = \text{blkdiag}\{R, S\}, \Sigma_k = \text{blkdiag}\{Q, R, S\}\) for all \(k \in \{1, \ldots, N-1\}, \Sigma_N = P\) and set

\[
\xi_0^m = \xi_0^*(\theta_0^m, x_0), \quad \xi_k^m = \xi_1^*(\theta_k^m)
\]

for all \(k \in \{1, \ldots, N\}\) by evaluating the respective explicit solution maps (14). In this paper, we use the enumeration-based multi-parametric QP algorithm from [20] for generating these maps.

Notice that the complexity of pre-processing the small-scale QPs (14) solely depends on the maximum number \(N_R = \max\{N_{R,0}, N_{R,1}, \ldots, N_{R,N}\}\) of critical regions over which the PWA optimizers \(\xi_0^*, \xi_1^*, \) and \(\xi_N^*\) are defined [1], but \(N_R\) is independent of the prediction horizon \(N\) as summarized in Table 1; see also [33,8,6]. Here, we assume that each parametric QP is post-processed, off-line, to obtain binary search trees [38] in \(O(N_R^3)\) time. Once the trees are constructed, they provide for a fast evaluation of the solution maps in (14) in time that is logarithmic in the number of regions, thus establishing the \(O(N \log^2(N_R))\) on-line computational bound. The memory requirements are directly proportional to the number of regions \(N_R\) with each region represented by a finite number of affine half-spaces.

### 4.2 Sparse linear algebra

In Step 2b) of Algorithm 1 the large-scale, coupled QP (9) must be solved. Because this QP has equality constraints only, (9) is equivalent to a large but sparse system of equations. Moreover, all matrices in (9) are given and constant during the online iterations. This means that all linear algebra decompositions can be precomputed offline. If one uses standard Riccati recursions for exploiting the band-structure of (9), the computational complexity for all offline computations is at most of order \(O(N^5)\), where \(n = n_u + n_x\), while the online implementation has complexity \(O(N^3)\) [2]. If one considers interconnected systems this run-time result may be improved—in many practical cases, e.g., in the example that is discussed in Section 5, one may set \(n = (n_u + n_x)/I\). However, in general, the choice of the linear algebra solver and its computational complexity depend on the particular topology of the network [6].

In summary, Algorithm 1 can be implemented by using static memory only allocating at most \(O(N_R + N^2)\) floating point numbers. Here, the explicit solutions maps of both the uncoupled QPs in Step 2a) as well as the coupled QP in Step 2b) can be pre-computed offline. Because Theorem 2 provides an explicit formula for computing a constant number of iterations \(\bar{m}\) such that a stable and recursively feasible controller is obtained, the online run-time of Algorithm 1 is constant and of order \(O(N \log^2(N_R) + N^2)\). Thus, Algorithm 1 may be called an explicit MPC method—in the sense that it has a constant run-time and constant memory requirements for any given \(N\) and \(I\) while stability and recursive feasibility can be verified offline. Its main advantage compared to existing MPC controllers is that it scales up easily for large scale interconnected networks of systems as well as long prediction horizons \(N\), as illustrated below.

### 5 Numerical example

This section applies Algorithm 1 to a spring-vehicle-damper control system, which has been introduced in Example 1 with state and control and constraints

\[
X = X_1 \times \ldots \times X_I, \quad U = [-2, 0.5], \quad Z = \mathbb{R}^{2I},
\]

where \(X_1 = \ldots = X_I = [-0.5, 1.5] \times [-0.5, 1]\).

The weighting matrices of the MPC objective are set to

\[
Q = 10I, \quad R = I, \quad S = 10^{-2}I.
\]
The numerical values for the mass $m$, spring constant $k$, and damping constant $d$ are listed below.

| Parameter                        | Symbol | Value [Unit] |
|----------------------------------|--------|--------------|
| sampling time                    | $h$    | 0.1 [s]      |
| spring constant                  | $k$    | 3 [N/m]      |
| mass of vehicle                  | $m$    | 1 [kg]       |
| viscous damping coefficient      | $d$    | 3 [N s/m]    |

Last but not least the matrix $P$ is computed by solving an algebraic Riccati equation, such that the terminal cost is equal to the unconstrained infinite horizon cost if none of the inequality constraints are active [35].

We have implemented Algorithm 1 in Matlab R2018a using YALMIP [28] and MPT 3.1.5 [21]. Here, the solution maps of the QPs (14) were pre-computed using the geometric parametric LCP solver of MPT 3.1.5 [21]. By exploiting the separability of the cost function and constraints, the storage space for the parametric solutions can be reduced to 287 kB, which corresponds to 432 critical regions. This memory requirement is independent of the number of vehicles $\bar{I}$ and the length of the prediction horizon $N$. In contrast to this, the number of regions for standard Explicit MPC depends on both $\bar{I}$ and $N$:

| $(\bar{I}, N)$ | # of regions | memory |
|----------------|--------------|--------|
| (1, 10)        | 58           | 14 [kB]|
| (1, 20)        | 84           | 40 [kB]|
| (1, 50)        | 144          | 169 [kB]|
| (2, 10)        | 2244         | 877 [kB]|
| (3, 10)        | 4247         | 2324 [kB]|

Notice that the number of regions of standard Explicit MPC explode quickly, as soon as one attempts to choose $N \geq 10$ or more than 3 vehicles. In contrast to this, our approach scales up easily to hundreds of vehicles and very long horizons. Here, only the evaluation of the solution map of (9) depends on $\bar{I}$ and $N$. For example, if we set $\bar{I} = 3$ and $N = 30$, we need 10 kB to store this map—for larger values this number scales up precisely as predicted by Table 1.

Figure 2 shows the total number of active constraints of all distributed QP solvers during the MPC iterations for different choices of $\bar{m}$. $\bar{m} = 1$ still leads to many wrongly chosen active sets—especially during the first 10 MPC iterations. However, for $\bar{m} \geq 10$ a reasonably accurate approximation of the optimal number of active constraints is maintained during all iterations.

![Fig. 2. The total number of active constraints of all distributed QP solvers during the MPC iterations for different choices of $\bar{m}$](image)

Notice that the number of regions of standard Explicit MPC explode quickly, as soon as one attempts to choose $N \geq 10$ or more than 3 vehicles. In contrast to this, our approach scales up easily to hundreds of vehicles and very long horizons. Here, only the evaluation of the solution map of (9) depends on $\bar{I}$ and $N$. For example, if we set $\bar{I} = 3$ and $N = 30$, we need 10 kB to store this map—for larger values this number scales up precisely as predicted by Table 1.

Figure 2 shows the total number of active constraints of all distributed QP solvers during the MPC iterations for different choices of $\bar{m}$. In order to visualize the performance of the proposed sub-optimal controller in terms of the number of correctly detected active constraint indices, the number of active constraints of non-suboptimal MPC (corresponding to $\bar{m} = \infty$) are shown in the form of red crosses in Figure 2. If we compare these optimal red crosses with the blue diamonds, which are obtained for $\bar{m} = 1$, we can see that the choice

Finally, Figure 3 shows the sub-optimality of Algorithm 1 in dependence on $\bar{m}$ for a representative case study with $\bar{I} = 3$ and $N = 30$—for other values of $\bar{I}$ and $N$ the convergence looks similar.

6 Conclusions

This paper has introduced a parallelizable and real-time verifiable MPC scheme, presented in the form of Algorithm 1. This control algorithm evaluates at every sampling time a finite number of pre-computed, explicit piecewise affine solution maps that are associated with parametric small-scale QPs. Because solving large-scale QPs in real-time may be impossible, the presented algorithm returns suboptimal control reaction on purpose—in order to be able to meet challenging real-time and limited memory requirements. The theoretical contributions of this paper have been presented in Theorem 2.
and Corollary 1, which provide both asymptotic stability guarantees as well as bounds on sub-optimality. The presented explicit MPC approach can be used to reduce the storage and run-time of explicit MPC by orders of magnitude, as illustrated by applying Algorithm 1 to a spring-mass-vehicle benchmark problem.

Acknowledgements

The authors thank three anonymous reviewers and the associate editor for their precise and substantial remarks, which have helped to significantly improve this article. 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. The work of Yuning Jiang and Boris Houska was supported by the National Science Foundation China (NSFC), No. 61473185, as well as ShanghaiTech University, Grant-No. F-020314-012. Juraj Oravec and Michal Kvasnica acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants 1/0085/19, 1/0112/16, and the Slovak Research and Development Agency under the project APVV-15-0007.

References

[1] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. Automatica, 38(1):3–20, 2002.
[2] D.P. Bertsekas. Dynamic Programming and Optimal Control. Athena Scientific Dynamic Programming and Optimal Control, Belmont, Massachusetts, 3rd edition, 2012.
[3] F. Blanchini. Set invariance in control — A survey. Automatica, 35:1747–1767, 1999.
[4] F. Borrelli. Constrained Optimal Control Of Linear And Hybrid Systems, volume 290 of Lecture Notes in Control and Information Sciences. Springer, 2003.
[5] F. Borrelli, A. Bemporad, and M. Morari. Geometric algorithm for multiparametric linear programming. Journal of optimization theory and applications, 118(3):515–540, 2003.
[6] F. Borrelli, A. Bemporad, and M. Morari. Predictive control for linear and hybrid systems. Cambridge University Press, 2017.
[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundation Trends in Machine Learning, 3(1):1–122, 2011.
[8] S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, 2004.
[9] R. Cagniard, P. Grieder, E.C. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. Journal of Process Control, 17(6):563–570, 2007.
[10] C. Conte, C.N. Jones, M. Morari, and M.N. Zeilinger. Distributed synthesis and stability of cooperative distributed model predictive control for linear systems. Automatica, 69:117–125, 2016.
[11] C. Conte, T. Summers, M.N. Zeilinger, M. Morari, and C.N. Jones. Computational aspects of distributed optimization in model predictive control. In Proceedings of the 51st IEEE Conference on Decision and Control, 2012, pages 6819–6824, 2012.
[12] M. Diehl, H.G. Bock, J.P. Schlöder, R.Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. Journal of Process Control, 12(4):577–585, 2002.
[13] M. Diehl, H.J. Ferreau, and N. Haverbeke. Efficient numerical methods for nonlinear MPC and moving horizon estimation. In L. Magni, M.D. Raimondo, and F. Allgöwer, editors, Nonlinear model predictive control, volume 384 of Lecture Notes in Control and Information Sciences, pages 391–417. Springer, 2009.
[14] H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Operations Research, 11(3):399–417, 1963.
[15] L. Ferranti, G. Stathopoulos, C. N. Jones, and T. Keviczky. Constrained LQR using online decomposition techniques. In IEEE Conference on Decision and Control, pages 2339–2344, Las Vegas, USA, 2016.
[16] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. qpoases: A parametric active-set algorithm for quadratic programming. Mathematical Programming Computation, 6(4):327–363, 2014.
[17] P. Gieslsson, M. Damg Doan, T. Keviczky, B. De Schutter, and A. Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. Automatica, 49(3):829–833, 2013.
[18] P. Grieder and M. Morari. Complexity reduction of receding horizon control. In IEEE Conference on Decision and Control, Maui, Hawaii, USA, 2003.
[19] L. Grüne. Analysis and design of unconstrained nonlinear mpc schemes for finite and infinite dimensional systems. SIAM Journal on Control and Optimization, 48(2):1206–1228, 2009.
[20] M. Herceg, C. N. Jones, M. Kvasnica, and M. Morari. Enumeration-based approach to solving parametric linear complementarity problems. Automatica, 62:243–248, 2015.
[21] M. Herceg, M. Kvasnica, C. Jones, and M. Morari. Multi-parametric toolbox 3.0. In 2013 European Control Conference, pages 502–510, 2013.
[22] B. Houska, H.J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. Automatica, 47:22792285, 2011.
[23] B. Houska, J. Frasch, and M. Diehl. An augmented Lagrangian based algorithm for distributed non-convex optimization. SIAM Journal on Optimization, 26(2):1101–1127, 2016.
[24] D. Ingole and M. Kvasnica. FPGA implementation of explicit model predictive control for closed loop control of depth of anesthesia. In 5th IFAC Conference on Nonlinear Model Predictive Control, pages 484–489, 2015.
[25] B. Karg and S. Lucia. Efficient representation and approximation of model predictive control laws via deep learning. arXiv.org/math.OC, abs/1806.10644, 2018.
[26] A. Kozma, J.V. Frasch, and M. Diehl. A distributed method for convex quadratic programming problems arising in optimal control of distributed systems. In Proceedings of the IEEE Conference on Decision and Control (CDC), 2013.
[27] M. Kvasnica, B. Takács, J. Holaza, and S. Di Cairano. On region-free explicit model predictive control. In Conference on Decision and Control (CDC), pages 3669–3674, 2015.
[28] J. Löfberg. YALMIP, 2004. Available from http://users.isy.liu.se/johanl/yalmip/.
[29] J. Mattingley and S. Boyd. Automatic code generation for real-time convex optimization. Convex optimization in signal processing and communications, pages 1–41, 2009.
[30] I. Necoara and J.A.K. Suykens. Application of a smoothing technique to decomposition in convex optimization. IEEE Transactions on Automatic Control, 53(11):2674–2679, 2008.
A Proof of Theorem 1

As the proof of Theorem 1 is not entirely straightforward, we first establish an intermediate results on the convergence properties of Algorithm 1, which is then, in a second step, used to obtain a convergence rate estimate. Therefore, this proof is divided into two subsections: Subsection A.1 analyzes the general convergence properties of Algorithm 1 and Subsection A.2 presents a complete proof of Theorem 1 by using these properties.

A.1 A closer look at the convergence properties of Algorithm 1

The goal of the section is to establish the following technical result.

Lemma 1 Let Assumption 1 be satisfied and let (7) be feasible, i.e., such that a unique minimizer $y^*$ and an associated dual solution $\lambda^*$ exist. Then the iterates of Algorithm 1 satisfy

$$\sum_{m=\hat{m}}^{\infty} F(\xi^m - y^*) \leq \frac{F(y^{\hat{m}} - y^*) + F^*(\lambda^{\hat{m}} - \lambda^*)}{4}$$

for all $\mathbf{m} \geq \hat{m}$ and all $\mathbf{n} \geq 2$.

Proof. Let us introduce the auxiliary functions

$$\mathcal{F}_0(\phi_0) = F_0(\phi_0) - (H_0^T \phi_0)^T \phi_0 + \nabla F_0(\phi_0)^T \phi_0,$$

$$\mathcal{F}_k(\phi_k) = F_k(\phi_k) + (G_k^T \lambda_k - H_k^T \phi_k)^T \phi_k + \nabla F_k(\phi_k)^T \phi_k.$$

Because $\xi^m_k$ is a minimizer of the $k$-th decoupled QP in Step 2a) of Algorithm 1, it must also be a minimizer of $\mathcal{F}_k$ on $Y_k$. Thus, because $\mathcal{F}_k$ is strongly convex with Hessian $\nabla^2 \mathcal{F}_k$, we must have

$$\sum_{k=0}^{N} \mathcal{F}_k(\xi^m_k) + \sum_{k=0}^{N} F_k(\xi^m_k - y_k^*) \leq \sum_{k=0}^{N} \mathcal{F}_k(y_k^*).$$

On the other hand, due to duality, we have

$$\sum_{k=0}^{N} F_k(\xi^m_k) + \langle \lambda^*, y^* \rangle + \sum_{k=0}^{N} F_k(\xi^m_k - y_k^*) \leq \sum_{k=0}^{N} \mathcal{F}_k(\xi^m_k) + \langle \lambda^*, \xi^m \rangle,$$

where the shorthand notation

$$\langle \lambda, y \rangle = -(H_k^T \lambda_k)^T y_k + \sum_{k=1}^{N} (G_k^T \lambda_{k-1} - H_k^T \lambda_k)^T y_k$$

is used to denote a weighted (non-symmetric) scalar product of primal and dual variables. Adding both inequalities and collecting terms yields

$$0 \geq \sum_{k=0}^{N} \nabla F_k(\xi^m_k - y_k^*)^T (\xi^m_k - y_k^*) + 2 \sum_{k=0}^{N} F_k(\xi^m_k - y_k^*)$$

$$+ \langle \lambda^m - \lambda^*, \xi^m - y^* \rangle.$$

Let us introduce the matrices

$$Q = \nabla^2 \left( \sum_{k=0}^{N} F_k \right), A = \nabla_{\lambda, y} \langle \lambda, y \rangle$$

such that the above inequality can be written in the form

$$0 \geq (\xi^m - y^*)^T Q (\xi^m - y^*) + 2 \sum_{k=0}^{N} F_k(\xi^m_k - y_k^*)$$

$$+ \langle \lambda^m - \lambda^*, \xi^m - y^* \rangle. \quad (A.1)$$

Similarly, the stationarity condition QP (9) can be written as

$$Q(y^{m+1} - 2\xi^m + y^m) + A^T \delta^m = 0.$$

Because $Q$ is positive definite, we solve this equation with respect to $\xi^m$ finding

$$\xi^m = \frac{1}{2} Q^{-1} A^T (\lambda^{m+1} - \lambda^m) + \frac{y^m + y^{m+1}}{2}. \quad (A.2)$$
Here, we have additionally substituted the relation
\[ \delta^m = \lambda^{m+1} - \lambda^m. \]

Notice that we have \( Ay^m = Ay^{m+1} = Ay^\ast \) for all \( m \geq 2 \), because the solutions of the QP (9) must satisfy the equality constraints in (7). If we substitute this equation and the expression for \( \xi^m \) in (A.1), we find that
\[
-2F(\xi^m - y^*) \\
\geq (\xi^m - y^*)^T Q(\xi^m - y^*) + (\lambda^m - \lambda^*)^T A(\xi^m - y^*) \\
= \frac{1}{4}(\lambda^{m+1} - \lambda^m)^T AQ^{-1} A^T (\lambda^{m+1} - \lambda^m) \\
+ \frac{1}{4}(y^{m+1} - y^m)^T Q(y^{m+1} - 2y^* + y^{m+1}) \\
+ \frac{1}{2}(\lambda^m - \lambda^*)^T AQ^{-1} A^T (\lambda^m - \lambda^*) \\
= \frac{1}{2}(F(y^{m+1} - y^*) - F(y^m - y^*)) \\
+ \frac{1}{2}(F^*(\lambda^{m+1} - \lambda^*) - F^*(\lambda^m - \lambda^*))
\]
for all \( m \geq 2 \). Now, the statement of Lemma 1 follows by summing up the above inequalities for \( m = \hat{m} \) to \( m = m \) and using that the last element in the telescoping sum on the right hand,
\[ \frac{F(y^{\hat{m}+1} - y^*) + F^*(\lambda^{\hat{m}+1} - \lambda^*)}{2} \geq 0 \]
is non-negative.

### A.2 Analysis of the convergence rate of Algorithm 1

The goal of this section is to prove the statement of Theorem 1 by using the intermediate result from the previous section. Let \( \hat{Y}_k \) denote the intersection of all active supporting hyperplanes at the solutions of the small scale QPs of Step 2a) in Algorithm 1 for \( k \in \{0, \ldots, N\} \) at a given iteration \( m \). We construct the auxiliary optimization problem
\[
\min_{\hat{y}} \sum_{k=0}^{N} F_k(\hat{y}_k) \\
\text{s.t.} \begin{cases}
\forall k \in \{0, \ldots, N-1\}, \\
G_{k+1} \hat{y}_{k+1} = H_k \hat{y}_k + h_k | \hat{\lambda}_k , \\
0 = H_N \hat{y}_N | \hat{\lambda}_N , \\
\hat{y}_k \in \hat{Y}_k , \hat{y}_N \in \hat{Y}_N ,
\end{cases}
\]
and denote optimal primal and dual solutions of this problem by \( \hat{y}^\ast \) and \( \hat{\lambda}^\ast \). Next, we also construct the auxiliary QPs
\[
\min_{\xi^\ast \in \hat{Y}_k} F_k(\xi^\ast) = (G_k^T \lambda^\ast_k - H_k^T \lambda^\ast_k)^T \xi^\ast + F_k(\xi^\ast - y^\ast_k) , \\
\min_{\xi^0 \in \hat{Y}_0} F_0(\xi^0) = (H_0^T \lambda^0_0)^T \xi^0 + F_0(\xi^0 - y^0_k) ,
\]
Because these QPs have equality constraints only, their parametric solutions must be affine. Thus, there exists a matrix \( T_1 \) such that
\[
\xi^m - \hat{y}^\ast = T_1 \begin{pmatrix} y^m - \hat{y}^\ast \\ \lambda^m - \hat{\lambda}^\ast \end{pmatrix} .
\]
Similarly, the coupled QP (9) has equality constraints only, i.e., there exists a matrix \( T_2 \) such that
\[
\begin{pmatrix} y^{m+1} - \hat{y}^\ast \\ \lambda^{m+1} - \hat{\lambda}^\ast \end{pmatrix} = T_2 \begin{pmatrix} \xi^m - \hat{y}^\ast \\ y^m - \hat{y}^\ast \end{pmatrix} .
\]
Now, we use the equation \( \lambda^{m+1} - \lambda^\ast = \lambda^m - \lambda^\ast + \delta \) and substitute the above equations finding that
\[
\begin{pmatrix} y^{m+1} - \hat{y}^\ast \\ \lambda^{m+1} - \hat{\lambda}^\ast \end{pmatrix} = T \begin{pmatrix} y^m - \hat{y}^\ast \\ \lambda^m - \hat{\lambda}^\ast \end{pmatrix}
\]
with
\[
T = \begin{pmatrix} T_2 & T_1 \\ (I_0) & (0 I) \end{pmatrix}.
\]
Next, we know from Lemma 1 that if we would apply Algorithm 1 to the auxiliary problem (A.4), the corresponding primal and dual iterates would converge to \( \hat{y}^\ast \) and \( \hat{\lambda}^\ast \). In particular, inequality (A.3) from the proof of Lemma 1 can be applied finding that
\[
\begin{pmatrix} y^{m+1} - \hat{y}^\ast \\ \lambda^{m+1} - \hat{\lambda}^\ast \end{pmatrix}^T \begin{pmatrix} y^{m+1} - \hat{y}^\ast \\ \lambda^{m+1} - \hat{\lambda}^\ast \end{pmatrix} \\
+ \begin{pmatrix} \lambda^{m+1} - \hat{\lambda}^\ast \end{pmatrix}^T AQ^{-1} A^T \begin{pmatrix} \lambda^{m+1} - \hat{\lambda}^\ast \\ \lambda^m - \hat{\lambda}^\ast \end{pmatrix} < 0
\]
whenever \( \begin{pmatrix} y^m - \hat{y}^\ast \\ \lambda^m - \hat{\lambda}^\ast \end{pmatrix} \neq 0 \). By substituting the linear equation (A.5), we find that this is only possible if
\[
T^T \begin{pmatrix} Q & 0 \\ 0 & AQ^{-1} A^T \end{pmatrix} \leq \kappa_A I
\]
for a constant \( \kappa_A < 1 \). Now, one remaining difficulty is that the constant \( \kappa_A \) (as well as the matrix \( T \)) depends on the particular set \( A \) of active supporting hyperplanes in the small-scale QPs. Nevertheless, because there exists only a finite number of possible active sets, the maximum
\[
\kappa = \max_A \kappa_A
\]
must exist and satisfy \( \kappa < 1 \). Now, the equation
\[
\begin{pmatrix} y^{m+1} - \hat{y}^\ast \\ \lambda^{m+1} - \hat{\lambda}^\ast \end{pmatrix} = T \begin{pmatrix} y^m - \hat{y}^\ast \\ \lambda^m - \hat{\lambda}^\ast \end{pmatrix}
\]
holds only for our fixed $m$ and the associated matrix $T$ for a particular active set, but the associated descent condition

\[
(y^{m+1} - y^*)^\top Q (y^{m+1} - y^*) + (\lambda^{m+1} - \lambda^*)^\top AQ^{-1} A^\top (\lambda^{m+1} - \lambda^*) \leq \kappa \left((y^m - y^*)^\top Q (y^m - y^*) + (\lambda^m - \lambda^*)^\top AQ^{-1} A^\top (\lambda^m - \lambda^*)\right),
\]  

(A.8)

holds independently of the active set of the QPs in the $m$-th iteration and is indeed valid for all $m$. After re-introducing the functions $F$ and $F^*$, we obtain the statement of the theorem.

B Proof of Proposition 1

Because $\xi^{\text{opt}}_{\text{岳}}$ is a feasible solution of the first small-scale decoupled QP, we have $\xi^{\text{opt}}_{\text{岳}} \in \mathbb{Y}_0$. Notice that such a feasible solution exists due to the particular construction of the set $\mathbb{Y}_0$ and our assumption that the set $X$ is control invariant. Consequently, the next iterate for the state,

\[ x_0^+ = Ax_0 + (B \ C)\xi^{\text{opt}}_{\text{岳}} \]

must satisfy $x_0^+ \in X$ by construction. This is the first statement of Proposition 1. In order to establish the second statement, we observe that the particular construction of the sets $\mathbb{Y}_k$ implies that there exists a feasible point of (7) for any choice of $x_0 \in X$, because we choose $X_N = X$, but $X$ is control invariant. Thus, we must have $X \subseteq \mathcal{X}$. The other way around, if $x_0 \not\in X$ the state constraints are violated, i.e., we also have $X \not\subseteq \mathcal{X}$. Thus, we have $X = \mathcal{X}$ and, consequently, $\mathcal{X}$ is a control invariant set, too. This is sufficient to establish recursive feasibility of Algorithm 1.

C Proof of Theorem 2

Recall that stability proofs for standard MPC proceed by using the inequality

\[ J(x_1^+) \leq J(x_0) - F_0(y_0^*) , \]

(C.1)

which holds if Assumption 3 is satisfied (see, e.g., [19] for a derivation of this inequality), i.e., $J$ is a global Lyapunov function on $\mathcal{X}$. Now, if we implement Algorithm 1 with a finite $\overline{m}$, we have

\[ J(x_0^+) \leq J(x_0) - (F_0(y_0^*) - J(x_0^+) + J(x_1^+)) , \]

(C.2)

i.e., $J$ can still be used as a Lyapunov function proving asymptotic stability as long as we ensure that

\[ F_0(y_0^*) - J(x_0^+) + J(x_1^+) \geq \alpha F_0(y_0^*) , \]

(C.3)

for a constant $\alpha > 0$. In order to show that such an inequality can indeed be established for a finite $\overline{m}$, we need the following technical result.

Lemma 2 The iterate $x_0^+$ satisfies the inequality

\[ (x_0^+ - x_1^+)^\top Q (x_0^+ - x_1^+) \leq \sigma \gamma^2 (1 + \sqrt{\kappa})^2 \kappa \overline{m} F_0(y_0^*) . \]

Proof. We start with the equation

\[ x_0^+ - x_1^+ = (B \ C)(\xi_{\text{opt}}^{\text{岳}} - y_0^*) = PA\left(\xi_{\text{opt}}^{\text{岳}} - y^*\right) , \]

which holds for the projection matrix $P = \text{diag}(I, 0, \ldots, 0)$ that filters out the first block component of the equality constraint residuum $A(\xi_{\text{opt}}^{\text{岳}} - y^*)$. Next, we substitute (A.2), which yields

\[ x_0^+ - x_1^+ = PA\left[\frac{Q^{-1} A^\top (\lambda_{\text{opt}}^{m+1} - \lambda^{m})}{2} + \frac{y_{\text{opt}}^{m+1} + y_{\text{opt}}^{m}}{2} - y^*\right] = \frac{1}{2} PA Q^{-1} A^\top (\lambda_{\text{opt}}^{m+1} - \lambda^{m}) , \]

(C.4)

where we have used that $0 = A(y_{\text{opt}}^{m} - y^*)$ and $0 = A(y_{\text{opt}}^{m+1} - y^*)$ recalling that these equations follow from the equality constraints in the coupled QP in Step 2b) of Algorithm 1. The particular definition of $\sigma$ implies that

\[ 4 (x_0^+ - x_1^+) \leq \lambda^{m+1} - \lambda^m , \]

\[ Q^{-1} A^\top (\lambda_{\text{opt}}^{m+1} - \lambda^{m}) \leq \sigma (\lambda_{\text{opt}}^{m+1} - \lambda^{m}) Q^{-1} A^\top (\lambda_{\text{opt}}^{m+1} - \lambda^{m}) . \]

Now, we can use the result of Theorem 1 to find

\[ (\lambda_{\text{opt}}^{m+1} - \lambda^{m}) Q^{-1} A^\top (\lambda_{\text{opt}}^{m+1} - \lambda^{m}) \leq \lambda^m (\lambda_{\text{opt}}^{m+1} - \lambda^{m}) + F^*(\lambda_{\text{opt}}^{m+1} - \lambda^{m}) + 2 \sqrt{\lambda_{\text{opt}}^{m+1} - \lambda^m} \lambda^m + \kappa (1 + \sqrt{\kappa})^2 \left(F(y^m - y^*) + F^*(\lambda^m - \lambda^{m})\right). \]

It remains to use the inequalities

\[ F(y^m) + F^*(\lambda^m) \leq \gamma^2 x_0^m Q x_0^m \leq \gamma^2 F_0(y_0^*) , \]

\[ F(y^1) + F^*(\lambda^1) \leq \gamma^2 x_0^1 Q x_0^1 \leq \gamma^2 F_0(y_0^*) , \]

which hold due to Assumption 4 and the particular construction in Step 1 of Algorithm 1, arriving at the inequality

\[ 4 (x_0^+ - x_1^+) \leq \sigma (\lambda_{\text{opt}}^{m+1} - \lambda^{m}) Q^{-1} A^\top (\lambda_{\text{opt}}^{m+1} - \lambda^{m}) \leq \sigma \kappa^m (1 + \sqrt{\kappa})^2 \left(F(y^m - y^*) + F^*(\lambda^m - \lambda^{m})\right) \leq 4 \sigma \kappa^m (1 + \sqrt{\kappa})^2 \gamma^2 F_0(y_0^*) . \]

Dividing by 4 on both sides yields the statement of the lemma. \qed

By combining the inequality from the above lemma and inequality (12) we find

\[ |J(x_0^+) - J(x_1^+)| \leq \left[\eta \sqrt{\sigma \gamma} (1 + \sqrt{\kappa}) + \frac{\tau \sigma \gamma^2 (1 + \sqrt{\kappa})^2}{2}\right] \kappa \overline{m} F_0(y_0^*) . \]

(C.5)

Now, the statement of Theorem 2 follows directly from (C.5), as the construction of $\overline{m}$ ensures that the Lyapunov descent condition (C.3) holds on $\mathcal{X}$ with

\[ \alpha = 1 - \left[\eta \sqrt{\sigma \gamma} (1 + \sqrt{\kappa}) + \frac{\tau \sigma \gamma^2 (1 + \sqrt{\kappa})^2}{2}\right] \kappa \overline{m} > 0 , \]

which is sufficient to establish asymptotic stability.