Assessing the speed-up achievable by online constraint removal in MPC

Michael Jost, Gabriele Pannocchia and Martin Mönnigmann

***Abstract*— We recently proposed to accelerate online MPC calculations by detecting and removing inactive constraints from the online optimization problems as a function of the current initial state. A number of variants of constraint removal (CR) have been explored, ranging from detecting inactive constraints based on precomputed regions of activity or approximations thereof to online methods that do not require any offline preparation. In typical applications CR can reduce the computing times required for the calculation of the model predictive control laws by 15% to 90%. Since CR is very easy to implement, does not require any additional assumptions to be fulfilled beyond the usual ones for stability, and can be combined with all optimization algorithms, it is very easy to cash in the described acceleration. Moreover, CR may prove useful if an existing, established MPC implementation needs to be accelerated, e.g., in order to use it on an embedded processor, but replacing it altogether is not an option.**

I. Introduction

Model predictive control (MPC) is a powerful tool for controlling constrained, multivariable systems. MPC requires to solve optimal control problems online, which makes it computationally expensive, however.

For the important class of linear systems with linear constraints on the states and inputs the optimal control problem to be solved is a quadratic program. Bemporad et al. [1] showed that the solution to this quadratic program (QP) is given by a piecewise affine function. Several approaches have used the insights into the structure of the state feedback law to solve the QPs more efficiently. Ferreau et al. [2] show that the piecewise structure of the explicit control law can be exploited to accelerate the online computations of MPC. Pannocchia et al. [3] enumerate the active sets which occur most frequently during runtime. Finally, the authors of the present paper showed in [4]–[7] that the time to calculate the control law can be significantly reduced by removing constraints from consideration that are known to be inactive at the optimum *before* solving the quadratic program. We remark that this does not only entail removing constraints that are always inactive (usually referred to as redundant constraints) by preprocessing, but we remove constraints as a function of the current initial state of the receding horizon problem.

In this paper we present a detailed simulation study on the approaches presented in [5]–[7]. First, we introduce the idea of CR, briefly discuss its variants and their online computational complexity. Then, we consider several examples which differ with respect to the number of states, inputs and horizon lengths. Both interior-point and active-set solvers are considered to investigate the influence of the QP solver.

M. Jost and M. Mönnigmann are with Automatic Control and Systems Theory, Department of Mechanical Engineering, Ruhr-Universität Bochum, Germany. G. Pannocchia is with Department of Civil and Industrial Engineering, University of Pisa, Italy. E-mail: michael.s.jost@rub.de, gabriele.pannocchia@unipi.it, martin.moennigmann@rub.de.

II. Problem Statement

We consider linear discrete-time state space systems

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

with states $x(t) \in \mathbb{R}^n$, inputs $u(t) \in \mathbb{R}^m$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and (A,B) stabilizable. State and input constraints read

$$x(t) \in \mathbb{X} \subset \mathbb{R}^n, \quad u(t) \in \mathbb{U} \subset \mathbb{R}^m, \tag{2}$$

for all t, where $\mathbb{U}$ and $\mathbb{X}$ are compact full dimensional polytopes that contain the origin in their interiors. MPC regulates system (1) to the origin by solving the optimal control problem

$$\begin{aligned} \min_{U,X} \quad & \ell_f(x(N)) + \sum_{k=0}^{N-1} \ell(x(k),u(k)) \\ \text{s.t.} \quad & x(k+1) = Ax(k) + Bu(k),\ k=0,\dots,N-1 \\ & x(0) = x, \\ & x(k) \in \mathbb{X}, \qquad k = 1,\dots,N, \\ & u(k) \in \mathbb{U}, \qquad k = 0,\dots,N-1, \\ & x(N) \in \mathbb{X}_f, \end{aligned} \tag{3}$$

on a receding horizon[1], where $X = (x'(1),\dots,x'(N))'$, $U = (u'(0),\dots,u'(N-1))'$, and N is the horizon length. Moreover, $\ell_f(x) = \frac{1}{2}x'\,P\,x$ and $\ell(x,u) = \frac{1}{2}x'\,Q\,x + \frac{1}{2}u'\,R\,u$, where $P \in \mathbb{R}^{n\times n}$, $P \succeq 0$, $Q \in \mathbb{R}^{n\times n}$, $Q \succeq 0$ and $R \in \mathbb{R}^{m\times m}$, $R \succ 0$ are the terminal, state and input weighting matrix, respectively. $\mathbb{X}_f \subset \mathbb{R}^n$ is a full dimensional polyhedral terminal set which contains the origin in its interior. We assume $(Q^{1/2}, A)$ to be detectable. Under these assumptions it can be shown that (3) is a strictly convex optimization problem.

By eliminating the states from (3), the optimal control problem (3) can be rewritten as the following QP

$$\begin{aligned} V(x,U) = \min_U \quad & \frac{1}{2}x'Yx + x'FU + \frac{1}{2}U'HU, \\ \text{s. t.} \quad & GU - Ex \le w, \end{aligned} \tag{$\mathbb{P}(x)$}$$

where $Y \in \mathbb{R}^{n\times n}$, $F \in \mathbb{R}^{n\times mN}$, $H \in \mathbb{R}^{mN\times mN}$, $G \in \mathbb{R}^{q\times mN}$, $w \in \mathbb{R}^q$, $E \in \mathbb{R}^{q\times n}$, and q denotes the number of constraints. H is a positive definite matrix and thus $\mathbb{P}(x)$ has a unique solution if it exists [1].

A. Notation

Consider any matrix $M \in \mathbb{R}^{a\times b}$. We denote by M^i the i-th row vector of M. By $\mathcal{X} \subseteq \mathbb{X}$ we denote the set of states for which the $\mathbb{P}(x)$ is feasible. Let $U^\star : \mathcal{X} \to \mathbb{U}^N$ and $u^\star : \mathcal{X} \to \mathbb{U}$ be the optimal solution of $\mathbb{P}(x)$ and its first m elements, respectively. Let $V^\star : \mathcal{X} \to \mathbb{R}$, $V^\star(x) = \frac{1}{2}x'Yx + x'FU^\star(x) + \frac{1}{2}U^{\star\prime}(x)HU^\star(x)$ be the corresponding optimal

[1]The notation $x(0) = x$ is understood to mean that the initial condition for the prediction is set to the current system state. We choose this notation for simplicity here, because the predicted states are eliminated eventually (see $\mathbb{P}(x)$).

value function.
Let $\mathcal{Q} = \{1, \ldots, q\}$. We call a constraint *active* for an arbitrary but fixed $x \in \mathcal{X}$, if $G^i U^\star(x) - E^i x = w^i$ and *inactive*, if $G^i U^\star(x) - E^i x < w^i$. It is meaningful to define the sets of active and inactive constraints for every state,

$$\begin{aligned} \mathcal{A}(x) &= \left\{ i \in \mathcal{Q} \,\middle|\, G^i U^\star(x) - E^i x = w^i \right\}, \\ \mathcal{I}(x) &= \mathcal{Q} \backslash \mathcal{A}(x), \end{aligned} \tag{4}$$

respectively. For later use we note that the solution of the optimization problem $\min\limits_U V(x, U)$, i.e. $\mathbb{P}(x)$ without constraints, is given by

$$U^\star(x) = -H^{-1} F' x. \tag{5}$$

$\mathbb{P}(x)$ is solved by a continuous piecewise affine function of x, defined on a partition of the state space into a finite number of convex polytopes [1]. In other words, there exists $n_\mathcal{P}$ polytopes $\mathcal{P}_i$, gains $\bar{K}_i \in \mathbb{R}^{mN \times n}$ and biases $\bar{b}_i \in \mathbb{R}^{mN}$ such that

$$U^\star(x) = \begin{cases} \bar{K}_1 x + \bar{b}_1 & \text{if} \quad x \in \mathcal{P}_1, \\ \quad \vdots & \qquad \vdots \\ \bar{K}_{n_\mathcal{P}} x + \bar{b}_{n_\mathcal{P}} & \text{if} \quad x \in \mathcal{P}_{n_\mathcal{P}}. \end{cases} \tag{6}$$

III. Constraint Removal in Model Predictive Control

It is the central idea of constraint removal to eliminate constraints that are inactive at the optimal solution from the quadratic program $\mathbb{P}(x)$ *before* solving it. We start by stating the reduced optimal control problem more precisely.

Proposition 1 (Reduced optimization problem) **[7]** *Let $x \in \mathcal{X}$ be arbitrary and let $\tilde{\mathcal{I}} \subseteq \mathcal{I}(x)$ be any subset of the inactive constraints. Consider the reduced optimization problem*

$$\begin{aligned} V(x, \tilde{U}) = \min_{\tilde{U}} \quad & \frac{1}{2} x' Y x + x' F \tilde{U} + \frac{1}{2} \tilde{U}' H \tilde{U}, \\ \text{s. t.} \quad & G^{\mathcal{Q} \backslash \tilde{\mathcal{I}}} \tilde{U} - E^{\mathcal{Q} \backslash \tilde{\mathcal{I}}} x \leq w^{\mathcal{Q} \backslash \tilde{\mathcal{I}}}. \end{aligned} \tag{$\tilde{\mathbb{P}}(x)$}$$

Then $\tilde{\mathbb{P}}(x)$ and $\mathbb{P}(x)$ have the same unique solution $\tilde{U}^\star(x) = U^\star(x)$.

The question arises how the subset $\tilde{\mathcal{I}}$ can be constructed before solving the optimization problem. Several methods exist. We briefly summarize them in Secs. III-A and III-B.

A. Constraint removal based on structural information

Two approaches are discussed in Sec. III-A. The first one is based on so-called regions of activity, the second approach uses the cost function to identify inactive constraints.

Regions of activity

Regions of activity $\mathcal{G}_i$ have been introduced in [4], [5] to identify the set $\tilde{\mathcal{I}}$ before runtime. The region of activity of constraint i is defined as a subset of $\mathcal{X}$ in which the constraint is active, i.e. $\mathcal{G}_i = \{x \in \mathcal{X} | \, i \in \mathcal{A}(x)\}$, or equivalently,

$$\mathcal{G}_i = \{x \in \mathcal{X} | \, G^i U^\star(x) - E^i x = w^i\}. \tag{7}$$

By definition this yields

$$i \in \mathcal{A}(x) \text{ if and only if } x \in \mathcal{G}_i. \tag{8}$$

Assume the regions of activity have been determined before runtime of the MPC, and let $x \in \mathcal{X}$ be arbitrary. Then we can determine, for every constraint $i \in \mathcal{Q}$, if it is active by checking whether $x \in \mathcal{G}_i$.
Unfortunately, the regions of activity are neither convex nor connected in general [5]. Both determining the $\mathcal{G}_i$ offline and carrying out tests of the type $x \in \mathcal{G}_i$ is therefore computationally expensive. Since it may in fact be prohibitive to check whether $x \in \mathcal{G}_i$ for all i online, we suggested to use *convex* outer approximation $\hat{\mathcal{G}}_i \supset \mathcal{G}_i$ of the regions of activity [5]. If, however, we replace $\mathcal{G}_i$ by a convex outer approximation $\hat{\mathcal{G}}_i \supset \mathcal{G}_i$, the equivalence in (8) no longer holds but has to be replaced by the implication

$$i \in \mathcal{A}(x) \quad \text{implies} \quad x \in \hat{\mathcal{G}}_i.$$

This leads to the following lemma.

Lemma 1 **[5], [7]** *Let $x \in \mathcal{X}$ be arbitrary, let $i \in \mathcal{Q}$ be an arbitrary constraint and let $\mathcal{G}_i$ be as in (7). Consider any outer approximation $\hat{\mathcal{G}}_i \supset \mathcal{G}_i$ of the region of activity $\mathcal{G}_i$. Then*

$$x \notin \hat{\mathcal{G}}_i \Rightarrow i \in \mathcal{I}(x),$$

i.e. constraint i is inactive at the optimal solution to $\mathbb{P}(x)$.

Several convex outer approximations can be envisioned, for example, ellipsoids or hyperrectangles. In the remainder of the paper we denote these two cases by $\mathcal{E}_i$ and $\mathcal{H}_i$, respectively. See [5] for their construction.

Constraint removal based on the cost function

We showed in [7] that inactive constraints can also be detected using the optimal cost function $V^\star(x)$. To do so, a lower bound $\sigma_i^\star$ on $V^\star(x)$ for all $x \in \mathcal{G}_i$ is calculated before runtime for each constraint. Formally, $\sigma_i^\star$ satisfies

$$\sigma_i^\star < V^\star(x) \text{ for all } x \in \mathcal{G}_i, \tag{9}$$

where $\mathcal{G}_i$ is as in (7). If the optimal cost function of the closed-loop system is a Lyapunov function, it is strictly decreasing along any trajectory of the closed-loop system. Thus, if the cost function drops below the precalculated value, the corresponding constraint will remain inactive along the (nominal) trajectory of the system. This is summarized in the following lemma.

Lemma 2 **[7]** *Let $x(t_0) \in \mathcal{X}$ be arbitrary, let $i \in \mathcal{Q}$ be an arbitrary constraint and assume $\sigma_i^\star \in \mathbb{R}$ satisfies (9). Then*

$$V^\star(x(t_0)) < \sigma_i^\star \Rightarrow i \in \mathcal{I}(x(t)), \quad t \geq t_0,$$

i.e. constraint i remains inactive along the trajectory of the closed-loop system.

Lemma 2 does not only provide a criterion for the detection of inactive constraints, but also guarantees that the constraint remains inactive for all future time steps $t \geq t_0$. This is obviously very different from the inclusion tests given in Lemma 1, which have to be repeated in every time step. The lower bounds $\sigma_i^\star$ can be calculated by solving a strictly convex optimization problem of the same size as $\mathbb{P}(x)$ (see [7] for details). Since the value of $V^\star(x(t_0))$ must be known to identify inactive constraints, Lemma 2 cannot be applied at the initial state. Since $\sigma_i^\star = \infty$ results for redundant constraints [7], they can be removed even at the initial state.

B. Online constraint removal

The methods presented so far require to calculate some information *offline*, which is then used to identify the inactive constraints *online*. Some of these calculations are computationally expensive. In addition, the precalculated information depends on parameters of the MPC formulation (3) such as weighting matrices or horizons. Thus, a recalculation is

TABLE I

CALCULATIONS AND COMPUTATIONAL EFFORT REQUIRED TO IDENTIFY INACTIVE CONSTRAINTS.

Name	Description	Inclusion test	Online Effort
hyp-MPC	Hyperrectangular approximation of $\mathcal{G}_i$	$x \notin \mathcal{H}_i \Rightarrow i \in \mathcal{I}(x)$	$\mathcal{O}\left(n^2\right)$
ell-MPC	Ellipsoidal approximation of $\mathcal{G}_i$	$x \notin \mathcal{E}_i \Rightarrow i \in \mathcal{I}(x)$	$\mathcal{O}\left(n^2\right)$
pre-lyap-MPC	Upper bounds on the cost function	$V^\star(x) < \sigma_i \Rightarrow i \in \mathcal{I}(x)$	$\mathcal{O}\left(1\right)$
lyap-MPC	Online constraint removal	$\|G^i\| c_U < E^i x^+ + w^i \Rightarrow i \in \mathcal{I}(x)$	$\mathcal{O}\left(n + {}^{(n^2+nmN)}/_{q}\right)$

necessary if these parameters are adjusted during controller tuning.

This motivates the search for methods that identify inactive constraints without precalculated information. Such a method is presented in [6]. The method is essentially based on two ideas. For one, if a bound $c_U \in \mathbb{R}$ is known on the norm of the optimal input vector, i.e. $\|U^\star(x)\|_2 \leq c_U$, this bound can be used to detect inactivity. This is stated more precisely in the following lemma.

Lemma 3 **[6]** *Let $x \in \mathcal{X}$ be arbitrary and assume there exists $c_U \geq 0$ such that $\|U^\star(x)\|_2 \leq c_U$. Then*

$$\left\|G^i\right\|_2 c_U < E^i x + w^i \Rightarrow i \in \mathcal{I}(x), \tag{10}$$

i.e. constraint i is inactive at the optimal solution to $\mathbb{P}(x)$.

Secondly, appropriate bounds c_U can be calculated from geometric properties of the bounds and the decay of the optimal cost function along any closed-loop trajectory. Assume that the optimal cost function $V^\star(x)$ is a Lyapunov function of the closed-loop system. Since $V^\star(x)$ is strictly decreasing under this assumption, the current state x and any candidate $U \in \mathbb{R}^{mN}$ have to satisfy $V(x,U) < V^\star(x^-)$, or, equivalently

$$\frac{1}{2}x'Yx + x'FU + \frac{1}{2}U'HU < V^\star(x^-), \tag{11}$$

where x^- is the preceding state, i.e. $x = Ax^- + Bu^\star(x^-)$. We showed in [6] that (11) defines an ellipsoid in the augmented input space, which can be rewritten as

$$\mathcal{U} = \left\{ U \in \mathbb{R}^{mN} \,\middle|\, \|U + H^{-1}Fx\|^2_{\frac{1}{2}H} \leq \rho(x^-) \right\}, \tag{12}$$

where $\rho(x^-)$ is a strictly positive function [6]. Thus, the optimal solution to $\mathbb{P}(x)$ is restricted to lie inside the ellipsoid (12), i.e. $U^\star(x) \in \mathcal{U}$. The ellipsoid $\mathcal{U}$ implies an upper bound c_U on $\|U^\star(x)\|_2$, which can be used together with (10) to detect inactive constraints [6]. We stress that the bound c_U cannot be applied at the initial state, since $V^\star(x^-)$ is unknown. However, we show in [6] how to combine c_U with a second bound that does not depend on x^- and therefore is available at the initial state.

IV. SOME NOTES ON THE COMPUTATIONAL COMPLEXITY

We describe the computational effort required to detect inactive constraints with the methods summarized in the previous section. We use the symbol $\mathcal{O}\left(\cdot\right)$ to denote the order of the number of elementary floating point arithmetic operations such as additions, multiplications and comparisons necessary for a given calculation. For example, the inner product $a'b$ for $a \in \mathbb{R}^n$, $b \in \mathbb{R}^n$ is of order $\mathcal{O}\left(n\right)$, since it requires n multiplications and $n-1$ additions.

Hyperrectangular and ellipsoidal approximations of $\mathcal{G}_i$

We have to check whether $x \in \mathcal{H}_i$ (resp. $x \in \mathcal{E}_i$) to determine whether the constraint i is active. It is easy to show that $\mathcal{O}\left(n^2\right)$ operations are required if $\mathcal{H}_i$ (resp. $\mathcal{E}_i$) is a hyperrectangle (resp. ellipsoid) [5].

Upper Bound on $V^\star(x)$ on $\mathcal{G}_i$

According to Lemma 2 we have to test whether $V^\star(x) < \sigma_i^\star$. Since this is a comparison of two real numbers, this test is of order $\mathcal{O}\left(1\right)$.

Online constraint removal

We have to test whether $\|G^i\| c_U < E^i x + w^i$ to check if constraint i is inactive. Since the inner product $E^i x$ is involved, $\mathcal{O}\left(n\right)$ operations are needed. Determining c_U requires $\mathcal{O}\left(n^2 + nmN\right)$ operations, which yields $\mathcal{O}\left((n^2+nmN)/q\right)$ operations per constraint on average. The number of arithmetic operations is therefore of order $\mathcal{O}\left(n + (n^2+nmN)/q\right)$ for each constraint.

Table I gives a summary.

V. SIMULATION STUDY

Four MPC problems (3) serve as examples in this simulation study. These four examples are summarized in Tab. II; some details on the system are given in the Appendix. We combine the examples with four different QP solvers. Specifically these are the interior-point and active-set solver of the Matlab Optimization Toolbox [8] (int-pnt-cvx and act-set for short, respectively) and the interior-point solver qpip and the active-set solver qpas from the QPC library [9][2].

Combining four different examples and four different QP solvers obviously results in 16 example-solver-combinations. For each of these 16 combinations, we run 5 MPC variants:

- **full-MPC:** MPC without constraint removal.
- **hyp-MPC:** Constraint removal with hyperrectangular approximation of the regions of activity.
- **ell-MPC:** Constraint removal with ellipsoidal approximation of the regions of activity.
- **pre-lyap-MPC:** Constraint removal with precalculated bounds on the objective function.
- **lyap-MPC:** Online constraint removal.

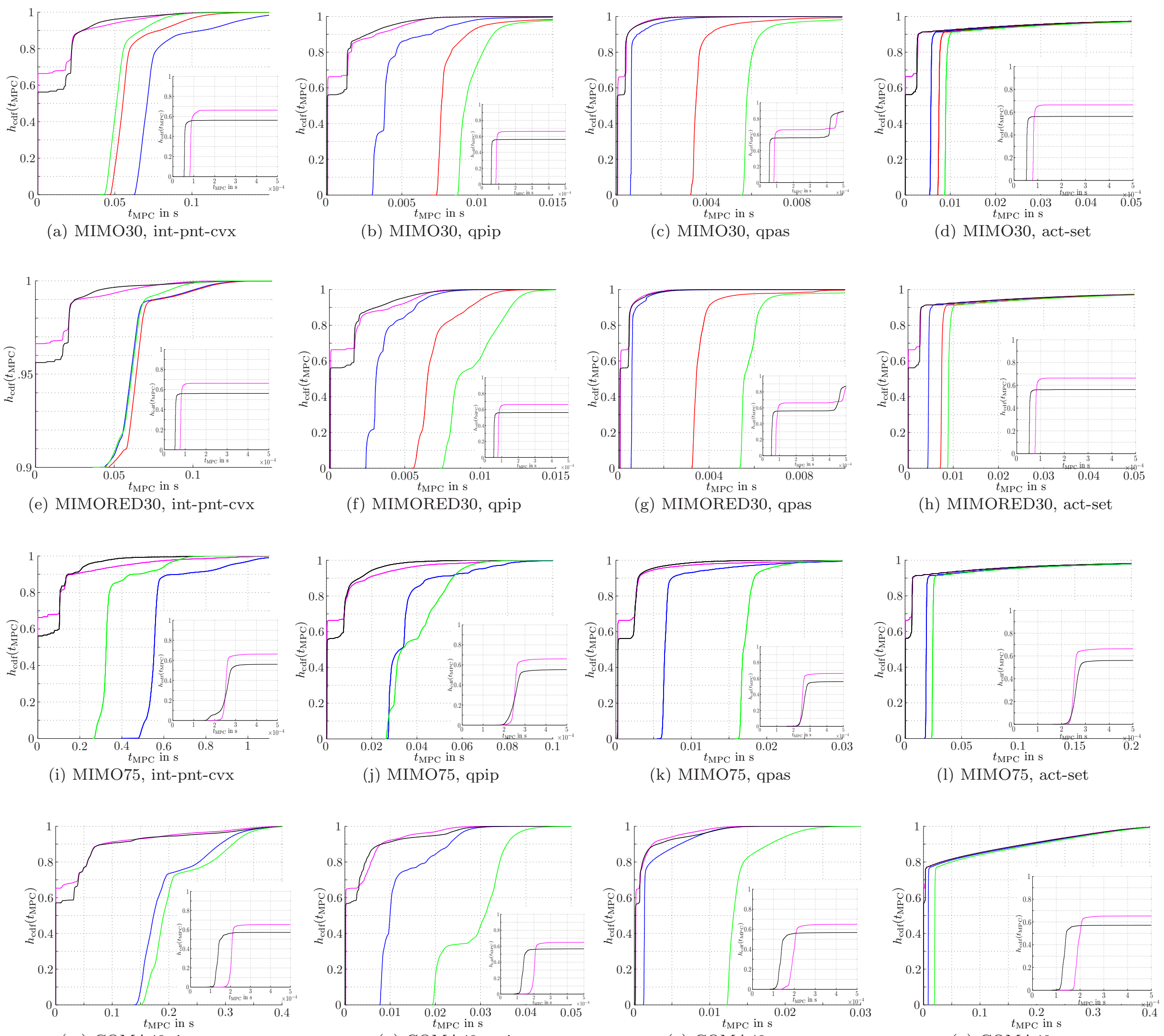
We refer to each of the $5 \times 16 = 80$ MPC implementations as a *case* for short. We generate random initial values $x \in \mathcal{X}$ for every system and calculate trajectories for the MPC-controlled system until $\|x(t)\| \leq 10^{-3}$ for every initial condition. The specific numbers of initial conditions and QPs are summarized in Tab. II. Note that more than 10^6 QPs are solved in every case. The same initial conditions are used in all cases for a given example. We implemented the QPs in

[2]All QPs were implemented in Matlab and solved on an Intel i5-3570 CPU with 8Gb RAM running Suse Linux with Kernel 3.11.10-7.

TABLE II

SUMMARY OF THE SAMPLE SYSTEMS

Name	n	m	N	mN	q	$\#x_0$	#QPs
MIMO30	10	3	30	90	780	6747	1110569
MIMORED30	10	3	30	90	556	6900	1136212
MIMO75	10	3	75	225	1950	6900	1136212
COMA40	12	3	40	120	1200	8098	1155202


(a) MIMO30, int-pnt-cvx (b) MIMO30, qpip (c) MIMO30, qpas (d) MIMO30, act-set

(e) MIMORED30, int-pnt-cvx (f) MIMORED30, qpip (g) MIMORED30, qpas (h) MIMORED30, act-set

(i) MIMO75, int-pnt-cvx (j) MIMO75, qpip (k) MIMO75, qpas (l) MIMO75, act-set

(m) COMA40, int-pnt-cvx (n) COMA40, qpip (o) COMA40, qpas (p) COMA40, act-set

Fig. 1. Cumulative distribution function $h_{\text{cdf}}(t_{\text{MPC}})$ for the examples MIMO30 (a-d), MIMO75 (e-h), MIMORED30 (i-l) and COMA40 (m-p). Results for hyp-MPC, ell-MPC, lyap-MPC and pre-lyap-MPC are in green, red, magenta and black, respectively, and results for full-MPC are in blue. Note that ellipsoidal outer approximations of the regions of activity could not be determined for MIMO75 and COMA40.

the form $\tilde{\mathbb{P}}(x)$, i.e., after eliminating the predicted states. We claim without giving details that the proposed approach can also be applied to (3) directly.

A. *Interpretation of the results*

We compare computational times using the cumulative distribution functions (cdf) $h_{\text{cdf}}(t_{\text{MPC}})$. The cumulative distribution function $h_{\text{cdf}}(t)$ is defined as the fraction of QPs in which the control law is found in time t or less. For each of the 16 combinations we determine the cumulative distribution function that results with hyp-MPC, lyap-MPC, pre-lyap-MPC and full-MPC. We only determined the cdf that results with ell-MPC for the examples MIMO30 and MIMORED30, since the calculation of the ellipsoidal approximations of the regions of activity was not possible for the examples MIMO75 and COMA40 in any reasonable time.

Computational times for the cases with constraint removal comprise the time required for constructing the set $\mathcal{J}(x)$, to set up the reduced quadratic program $\tilde{\mathbb{P}}(x)$, and to solve $\tilde{\mathbb{P}}(x)$.

The cdfs for all 80 cases are shown in Fig. 1(a-p) grouped by the 16 example-solver-combinations. Intervals for t_{MPC} are chosen such that approximately the range $[0, 0.99]$ of the cdf is visible. By definition of the cdf, the time t_{MPC} such that $h_{\text{cdf}}(t_{\text{MPC}}) = 1$ is the maximal computational time obtained for the respective case.

Results for MIMO30, Fig. 1(a–d)

- Consider the results achieved by pre-lyap-MPC and lyap-MPC for the MIMO30 example first, cf. Fig. 1(a–d), black and magenta curve. Approximately 55% and 65% of the QPs are detected to be unconstrained by pre-lyap-MPC and lyap-MPC, respectively. While full-MPC solves a QP, no optimization problem is solved at all by pre-lyap-MPC and lyap-MPC, but the optimal control law of the unconstrained case (5) can be evaluated immediately. Consequently, pre-lyap-MPC and lyap-MPC provide the optimal input sequence very quickly. This gives rise to the leftmost shoulders on all black and magenta curves in Figs 1(a–d).
- For all solvers pre-lyap-MPC and lyap-MPC outperform

TABLE III

RELATIVE DIFFERENCE OF THE AVERAGE COMPUTATION TIMES OF LYAP-MPC, PRE-LYAP-MPC, ELL-MPC AND HYP-MPC COMPARED TO FULL-MPC. THE VALUES IN EACH FIELD CORRESPOND TO LYAP-MPC, PRE-LYAP-MPC, ELL-MPC AND HYP-MPC, RESPECTIVELY.

Name	int-pnt-cvx				qpip				qpas				act-set			
MIMO30	-82.8%	-84.1%	-23.6%	-30.7%	-76.6%	-76.6%	89.4%	127.2%	-65.2%	-63.2%	358.4%	635.8%	-51.1%	-48.8%	19.2%	35.5%
MIMORED30	-82.8%	-80.9%	5.3%	-1.9%	-68.9%	-68.5%	94.6%	162.9%	-58.8%	-54.9%	437.5%	763.2%	-44.2%	-44.2%	36.1%	55.3%
MIMO75	-87.3%	-89.1%	–	-39.7%	-82.3%	-84.5%	–	10.6%	-76.5%	-78.0%	–	136.1%	-55.3%	-54.9%	–	17.5%
COMA40	-82.8%	-80.3%	–	8.5%	-75.3%	-73.5%	–	132.3%	-55.2%	-48.3%	–	489.7%	-16.5%	-15.3%	–	24.0%

full-MPC in the sense that the cdfs for both pre-lyap-MPC and lyap-MPC always lie to the left and above that for full-MPC. This is not the case for ell-MPC and hyp-MPC. Here, only for the int-pnt-cvx solver the cdfs lies to the left and above that for full-MPC. For the solvers qpip, qpas and act-set the cdfs of both ell-MPC and hyp-MPC lie to the right of that for full-MPC, which implies that ell-MPC and hyp-MPC have larger calculation times than full-MPC.

• The maximal computation time required by pre-lyap-MPC and lyap-MPC is smaller than that required by full-MPC for all QP solvers in the MIMO30 example. The maximal computation times required by ell-MPC and hyp-MPC are only smaller for the int-pnt-cvx solver, and larger than that required by full-MPC for qpas, qpip and act-set.

Results for MIMORED30, Fig. 1(e–h)

• Consider the results achieved by pre-lyap-MPC and lyap-MPC first, cf. Fig. 1(e–h), black and magenta curves. Again, approximately 55% and 65% of the QPs are detected to be unconstrained by pre-lyap-MPC and lyap-MPC, respectively. See the first comment on the MIMO30 results for a more detailed discussion.

• Again, pre-lyap-MPC and lyap-MPC outperform full-MPC in the sense stated in the second comment on the MIMO30 results. Full-MPC is faster than hyp-MPC and ell-MPC for the solvers qpip, qpas and act-set. The cdf of full-MPC and both hyp-MPC and ell-MPC are similar for the combination of MIMORED30 with int-pnt-cvx.

• Consider the cdfs of the full-MPC cases (blue curves in Fig. 1(e–f)). These cdfs have approximately the same shape as the corresponding ones for MIMO30 (blue curves in Fig. 1(a–d)), but they are shifted to smaller values of t_{MPC} here. This shift results, since the redundant constraints have been removed here. It appears to be small for the active-set solvers at first sight, but closer inspection reveals that it is still considerable. The leftmost shoulders are located at $0.65 \cdot 10^{-3}$s and $0.56 \cdot 10^{-3}$s for the solver qpas in Fig. 1(c) and Fig. 1(g), respectively. This amounts to an acceleration by about 20%. For the solver act-set the leftmost shoulders are located at $5.6 \cdot 10^{-3}$s and $4.4 \cdot 10^{-3}$s (see Fig. 1(d) and Fig. 1(h), respectively), which amounts to an acceleration by about 14%.

• Consider the cdfs of lyap-MPC and pre-lyap-MPC next. Note that these cdfs also have approximately the same shape as the corresponding cdfs resulting the MIMO30 example, but the shift is much smaller compared to that resulting for full-MPC. As an example consider the cdfs resulting for the act-set solver. The cdf of lyap-MPC and pre-lyap-MPC equals 0.7 at $2.71 \cdot 10^{-3}$s and $2.63 \cdot 10^{-3}$s in Fig. 1(a), and the cdfs equal 0.7 at $2.69 \cdot 10^{-3}$s and $2.67 \cdot 10^{-3}$s in Fig. 1(e). This amounts to a shift of about 0.7% and 1.5%, respectively. Thus, very similar calculation times result from applying lyap-MPC and pre-lyap-MPC to both MIMO30 and MIMORED30, which suggests that both lyap-MPC and pre-lyap-MPC remove many of the redundant constraints during runtime.

• The maximal computation time required by pre-lyap-MPC and lyap-MPC is smaller than that required by full-MPC for all QP solvers in this example. The difference is less pronounced for the active-set solvers than for the interior-point solvers. The maximal computation times required by hyp-MPC is larger than that required by full-MPC for qpas, qpip and act-set, and smaller than that required by full-MPC for int-pnt-cvx. The maximal computation times required by ell-MPC is larger in all cases.

Results for MIMO75, Fig. 1(i-l)

• Again, approximately 55% and 65% of the QPs are detected to be unconstrained by pre-lyap-MPC and lyap-MPC (black and magenta curves in Figs. 1(i-l)), respectively. See the first comment on the MIMO30 results for a more detailed discussion.

• MPC with constraint removal outperforms MPC without constraint removal for all solvers in the sense stated in the second comment on the MIMO30 results.

• All curves have approximately the same shape as the corresponding ones for MIMO30, but they are shifted to larger values of t_{MPC} here. This result is consistent with the fact that both MIMO30 and MIMO75 are based on the same system (1) and the same constraints (2), but the number of constraints is larger here due to the larger horizon.

• The maximal computation time required by pre-lyap-MPC and lyap-MPC is smaller than that required by full-MPC for all QP solvers in this example. The maximal computation times required by hyp-MPC is larger than that required by full-MPC for qpas and act-set, and smaller than that required by full-MPC for int-pnt-cvx and qpip.

Results for COMA40, Fig. 1(m-p)

• Approximately 55% and 65% of the QPs are detected to be unconstrained by pre-lyap-MPC and lyap-MPC, respectively. See the first comment on the MIMO30 results for a more detailed discussion.

• Pre-lyap-MPC and lyap-MPC outperform full-MPC for all solvers (cf. Fig. 1(m-p)) in the sense discussed in the second comment on the MIMO30 results. In this example full-MPC is always faster than hyp-MPC.

• The maximal computation time required by pre-lyap-MPC and lyap-MPC is smaller than full-MPC. The same observation holds as stated in the third comment on the MIMO30 example.

Table III summarizes the relative reduction of the average computation times for lyap-MPC, pre-lyap-MPC, ell-MPC and hyp-MPC compared to full-MPC. For example, a relative reduction of -82.8% can be achieved by applying lyap-MPC on the MIMO30 example in combination with the int-pnt-cvx solver. The data confirm that a large reduction results from applying lyap-MPC and pre-lyap-MPC to all examples (between -68% and -89% for interior-point solvers and between -17% and -78% for active-set solvers). Hype-MPC and ell-MPC increase the computation time in several cases (cf. Tab III, combination MIMO30 with qpas solver).

VI. Conclusion

Considerable reductions of the online computational effort result for the Lyapunov-function-based variants of constraint removal (lyap-MPC and pre-lyap-MPC, see the beginning of Sec. V for an explanation of these abbreviations). While the computational effort for detecting inactive constraints is higher for lyap-MPC than for pre-lyap-MPC (see Sec. IV), both methods result in very similar overall online computation time. This indicates that lyap-MPC detects more inactive constraints. This conjecture is corroborated by the fact the lyap-MPC detects unconstrained MPC problems most frequently (leftmost shoulders in Fig. 1). Since lyap-MPC does not involve expensive offline calculations, it is the method of choice for constraint removal. The other variants (ell-MPC and hyp-MPC) only result in reductions if combined with one of the interior point solvers (int-pnt-cvx). The reduction achieved in these cases are always smaller than for lyap-MPC and pre-lyap-MPC. Moreover, the maximal computation time required by ell-MPC and hyp-MPC increases for several example-solver combinations (see for example Fig. 1(c)). In these cases the effort required to identify the inactive constraints is larger than the savings obtained from solving $\tilde{\mathbb{P}}(x)$ instead of $\mathbb{P}(x)$.

The achieved reductions are in general larger for the interior-point solvers than for the active-set solvers. This is consistent with the fact that interior-point-solvers always involve all constraints in their calculations. Consequently, removing some of them in advance has a large effect. Active-set solvers, in contrast, operate on a subset of the constraints and therefore may not encounter a particular constraint, regardless of whether it is removed in advance or not.

The effect of disturbances and plant-model-mismatch remains to be investigated. Since constraint removal is based on conservative approximations of the regions of activities of the constraints, we expect there exists an inherent robustness, which will, however, be difficult to quantify.

Acknowledgment

Support by the Deutsche Forschungsgemeinschaft (DFG) under grant MO 1086/11-1 is gratefully acknowledged.

Appendix

MIMO30: We consider the state space system (1) resulting from discretizing the continuous-time transfer function

$$G(s) = \begin{pmatrix} \frac{-5s+1}{36s^2+6s+1} & \frac{0.5s}{8s+1} & 0 \\ 0 & \frac{0.1(-10s+1)}{s(8s+1)} & \frac{-0.1}{(64s^2+6s+1)s} \\ \frac{-2s+1}{12s^2+3s+1} & 0 & \frac{2(-5s+1)}{16s^2+2s+1} \end{pmatrix}, \quad (13)$$

with zero-order hold and sampling time $T_s = 1$s. After removing uncontrollable states from (13), the resulting state space model has 10 states and 3 inputs. The resulting system matrices are give in [10]. We consider the state and input constraints (2) $-10 \leq x_i(t) \leq 10, \forall i \in \big(1, \cdots, 10\big)$, $-1 \leq u_j(t) \leq 1, \forall j \in \big(1, \cdots, 3\big)$, respectively. We choose the weighting matrices as $Q = I^{n\times n}$, $R = 0.25 I^{m\times m}$ and P as the solution of the discrete-time algebraic Riccati equation (DARE). The horizon is $N = 30$. In summary, the resulting quadratic program has 90 decision variables and 780 constraints. We reparametrize the inputs with the LQR controller proposed in [11]. A condition number $\kappa(H) = 2.51$ results from this reparameterization.

MIMORED30: System matrices, constraints, weighting matrices and the input reparametrization are the same as in MIMO30. In contrast to MIMO30, we here remove redundant constraints from $\mathbb{P}(x)$. A constraint is called redundant if it never becomes active (see for example [12, Sec. 4.1.1, p. 128 ff.] or [13, Def. 5, p. 492 ff.]).

MIMO75: MIMO75 differs from MIMO30 only with respect to the horizon length, which is set to $N = 75$ here. System matrices, constraints, weighting matrices and the input reparametrization are as in MIMO30. The resulting optimization problem $\mathbb{P}(x)$ has $mN = 225$ decision variables and $q = 1950$ inequality constraints.

COMA40: This system models a linear chain of six masses connected to each other by springs, and to rigid walls on both ends of the chain [14], [15]. All masses and all spring constants are set to unity. There exist three inputs u_1, u_2, u_3 that model forces between the first and second, third and fifth, and fourth and sixth mass, respectively. The resulting system has 12 states and 3 inputs. Discretizing with zero-order hold and sample time $T_s = 0.5$s yields a system of the form (1). The state space matrices are given in [15]. The state and input constraints read $-4 \leq x_i(t) \leq 4, \quad i = 1, \ldots, 12,$, $-0.5 \leq u_j(t) \leq 0.5, \quad j = 1, \ldots, 3$, respectively. We choose $Q = I^{n\times n}$, $R = I^{m\times m}$ and set P to the solution of the DARE. The resulting quadratic program $\mathbb{P}(x)$ has $mN = 120$ decision variables and $q = 1200$ inequality constraints for a horizon of $N = 40$. After reparameterization with the LQR controller proposed in [11], a condition number of $\kappa(H) = 1.47$ results.

References

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

[2] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, pp. 816–830, 2008.

[3] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 43, pp. 852–860, 2007.

[4] M. Jost and M. Mönnigmann, "Accelerating online MPC with partial explicit information and linear storage complexity in the number of constraints," in *Proc. of the 2013 European Control Conference*, Zurich, Switzerland, 2013, pp. 35–40.

[5] ——, "Accelerating model predictive control by online constraint removal," in *Proc. of the 52th IEEE Conference on Decision and Control*, Florence, Italy, 2013, pp. 5764–5769.

[6] M. Jost, G. Pannocchia, and M. Mönnigmann, "Online constraint removal: accelerating MPC with a Lyapunov function," *Automatica*, 2015, Accepted.

[7] ——, "Accelerating linear model predictive control by constraint removal," *submitted*, 2014.

[8] *MathWorks Matlab 2013a Documentation Center: Quadratic Programming Algorithms.*

[9] D. A. Wills, *QPC - Quadratic Programming in C*, School of Electrical Engineering and Computer Science, University of Newcastle, 2009-08-11. [Online]. Available: http://sigpromu.org/quadprog/index.html

[10] M. Jost and M. Mönnigmann, "Minimal state space representation of an example system for online model predictive control," Ruhr-Universität Bochum, Tech. Rep., 2014.

[11] J. A. Rossiter, G. Kouvaritakis, and M. J. Rice, "A numerically robust state-space approach to stable-predictive control strategies," *Automatica*, vol. 34, pp. 65–73, 1998.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2009.

[13] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," *Automatica*, vol. 39, pp. 489 – 497, 2003.

[14] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[15] ——, "fast_mpc: code for fast model predictive control," 2008. [Online]. Available: http://stanford.edu/∼ boyd/fast_mpc/