arXiv:2002.06835v1 [eess.SY] 17 Feb 2020

On the Quadratic Programming Solution for Model Predictive Control with Move Blocking

Pavel Otta[1], Ondřej Šantin[2], and Vladimír Havlena[1]

***Abstract*— Model Predictive Control (MPC) is a popular optimization-based control technique. MPC is usually formulated as sparse or dense Quadratic Programming (QP). This paper reviews two well-known methods, namely, state condensing and move blocking, and brings them together. Their combination results in generalized QP that serves arbitrarily sparse (or dense) QP for MPC with *move blocking*. The proposed QP can be solved by a specialized solver capable of exploiting a sparsity structure of the problem. Numerical examples give inside in computational and memory requirements.**

***Index Terms*— Model Predictive Control; Box-Constrained Quadratic Programming; Move Blocking; State Condensing**

I. Introduction

Model Predictive Control (MPC) is a popular multivariable control technique that systematically incorporates physical constraints (e.g., potential or flow limits) by design. It is an optimization-based method, i.e., in every sampling period, a finite optimal control problem needs to be solved. For linear dynamics and quadratic costs, the problem to be solved is structured Quadratic Programming (QP). For nonlinear dynamics and nonlinear costs, the problem to be solved is NonLinear Programming (NLP). NLP can be solved by Sequential Quadratic Programming (SQP), which requires a sequence of structured QP to be solved. Therefore, the results presented in this paper can address nonlinear MPC as well.

The two most common QP formulations in which the MPC problem can be written are the *dense* and the *sparse* formulation [9]. In the sparse QP formulation, the minimization variables are inputs and states over the prediction horizon, and they are interconnected by the equality constraint representing system dynamics explicitly. Consequently, the QP problem is large in a number of variables with a specific sparsity pattern. On the other hand, in the dense QP formulation, the minimization variables are inputs over the prediction horizon only, the states are eliminated out, and the interconnection is held implicitly. Consequently, the QP problem is smaller, but with no sparsity pattern.

A comparison between sparse and dense QP formulation in the context of walking motion generation is presented in [4]. This application benefits from the use of sparse QP formulation as parameters change in their model; only a negligible additional computational effort is required. In some other applications (e.g., [15]), it might be beneficial to transform sparse QP to the dense one by a so-called *condensing* procedure. When a model is fixed, the condensing can be performed once offline, which leads to a significant computational saving. Note that condensing can be done with quadratic complexity in horizon length, as proposed in [7].

It should be noted that dense QP can be solved by a generic-purpose solver, which made the dense formulation more popular in the past. Nowadays, several structure-exploiting methods tailored for sparse QP arising in MPC exist (e.g. [6], [13], [19], [20]). As the problem formulation proposed in this work is derived from the sparse QP, these algorithms can be applied with no additional modification required.

When treating dense formulation, the computation complexity can be decreased by the *move-blocking* technique. The idea of this technique is to fix consequent inputs at the same value. Therefore the number of degrees of freedom of the optimization problem decreases. Another strategy to reduce the number of degrees of freedom is to utilize Linear Quadratic Regulator (LQR) [14]. A move-blocking strategy can optimize control performance, robustness, or feasibility when hard state constraints are considered [8], [17], [18]. The idea of using move blocking regardless of the sampling period was proposed in [5].

State condensing has been proposed in [1]. It enables sparsity level control of the QP formulation. The level of sparsity can be controlled smoothly in between the sparse (non-condensed) to dense (fully-condensed) QP formulation. This method was used for sped-up dual Newton step algorithm regarding nonlinear MPC in [11] or combined with the partial sensitivity update in [2] recently.

To the best author's knowledge, a generalization of the state condensing for MPC problem with move blocking has not been reported in the literature. From the other way around, move blocking has not been adapted for the sparse QP formulation yet. This paper attempts to address this issue.

The rest of the paper is organized as follows. In Section II, the MPC problem is introduced. Section III gives a basic QP problem formulation of the MPC. In Section IV and Section V, move-blocking and partial-condensing procedures are described, respectively. Section VI presents a QP transformation combining state condensing with move blocking. Section VII gives an insight in computational and memory requirements of the proposed method based on simulations.

The first author was supported by SGS (the Student Grant Fund) of the Czech Technical University.

[1]Czech Technical University in Prague, Technická 2, CZ-16627 Prague 6, Czech Republic ottapav1, havlena@fel.cvut.cz

[2]Garrett Motion Czech Republic s.r.o., Tuřanka 1387/100, CZ-62700 Brno - Slatina, Czech Republic ondrej.santin@garrettmotion.com

II. Model Predictive Control

We are concerned in a discrete-time Linear Time-Varying (LTV) systems in the form

$$\boldsymbol{x}(t+1) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) + \boldsymbol{w}(t), \quad t \geq 0,$$

where $\boldsymbol{A}(t) \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}}$, $\boldsymbol{B}(t) \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{u}}}$, and $\boldsymbol{w}(t) \in \mathbb{R}^{n_{\boldsymbol{x}}}$ at every time t are known. $\boldsymbol{A}(t), \boldsymbol{B}(t), \boldsymbol{w}(t)$ are deterministic, possibly arising from the linearization of nonlinear system. Input and state dimensions are denoted by $n_{\boldsymbol{u}}$ and $n_{\boldsymbol{x}}$, respectively.

Then the problem of the regulator with input box constraints can be treated as the following LTV MPC

$$\begin{aligned}
\min_{\substack{\boldsymbol{u}_0,\dots,\boldsymbol{u}_{N-1} \\ \boldsymbol{x}_1,\dots,\boldsymbol{x}_N}} \quad & \frac{1}{2}\sum_{k=0}^{N-1} \begin{bmatrix}\boldsymbol{u}_k \\ \boldsymbol{x}_k\end{bmatrix}^T \begin{bmatrix}\boldsymbol{R}_k & \boldsymbol{S}_k^T \\ \boldsymbol{S}_k & \boldsymbol{Q}_k\end{bmatrix} \begin{bmatrix}\boldsymbol{u}_k \\ \boldsymbol{x}_k\end{bmatrix} + \begin{bmatrix}\boldsymbol{u}_k \\ \boldsymbol{x}_k\end{bmatrix}^T \begin{bmatrix}\boldsymbol{r}_k \\ \boldsymbol{q}_k\end{bmatrix} \\
& + \frac{1}{2}\boldsymbol{x}_N^T \boldsymbol{Q}_N \boldsymbol{x}_N && \text{(1a)} \\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{w}_k, && \text{(1b)} \\
& \underline{\boldsymbol{u}}_k \leq \boldsymbol{u}_k \leq \overline{\boldsymbol{u}}_k, \ k = 0, \dots, N-1, && \text{(1c)} \\
& \boldsymbol{x}_0 = \hat{\boldsymbol{x}}(t), && \text{(1d)}
\end{aligned}$$

where $\hat{\boldsymbol{x}}(t) \in \mathbb{R}^{n_{\boldsymbol{x}}}$ is a current state measurement (estimation) at time t, $N \in \mathbb{N}$ is the finite prediction horizon length. For the sake of brevity, subscript k denotes time period from the sampling moment t. For example, $\boldsymbol{x}_k = \boldsymbol{x}(t+k) \in \mathbb{R}^{n_{\boldsymbol{x}}}$ and $\boldsymbol{u}_k = \boldsymbol{u}(t+k) \in \mathbb{R}^{n_{\boldsymbol{u}}}$ denote the state and input at stage k on prediction horizon, respectively. The quadratic weights are $\boldsymbol{R}_k \succ \boldsymbol{0}$, $\boldsymbol{Q}_k - \boldsymbol{S}_k \boldsymbol{R}_k^{-1} \boldsymbol{S}_k^T \succeq \boldsymbol{0}$, and the terminal weight $\boldsymbol{Q}_N \succeq \boldsymbol{0}$. The optimizer of problem (1) is a unique input sequence with associated state trajectory.

As there is new measurement $\hat{\boldsymbol{x}}(t)$ and time-varying model $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{w}_k, k = 0, \dots, N-1$ along the prediction horizon is available at each sampling instant, the problem (1) has to be reoptimized. Hence, the *receding horizon* concept is established, i.e., the plan of control inputs $\boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1}$ is recomputed at each sampling instant parametrized by measured system state $\hat{\boldsymbol{x}}$ and only the first control move $\boldsymbol{u}_0$ is actually applied to the system, cf. [12]. The need for re-computation requires a fast solver for problem (1), to have a solution ready by the next sampling time.

III. Sparse QP Formulation

Problem (1) can be rewritten straightforwardly as a box-constrained QP in the following form

$$\begin{aligned}
\min_{\mathbf{u},\mathbf{x}} \quad & \frac{1}{2}\begin{bmatrix}\mathbf{u} \\ \mathbf{x}\end{bmatrix}^T \begin{bmatrix}\mathbf{R} & \mathbf{S}^T \\ \mathbf{S} & \mathbf{Q}\end{bmatrix} \begin{bmatrix}\mathbf{u} \\ \mathbf{x}\end{bmatrix} + \begin{bmatrix}\mathbf{u} \\ \mathbf{x}\end{bmatrix}^T \begin{bmatrix}\mathbf{r}(\boldsymbol{x}_0) \\ \mathbf{q}\end{bmatrix} \\
& + c(\boldsymbol{x}_0) && \text{(2a)} \\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{u} + \mathbf{w}(\boldsymbol{x}_0), && \text{(2b)} \\
& \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}, && \text{(2c)}
\end{aligned}$$

parametrized by $\boldsymbol{x}_0 \in \mathbb{R}^{n_{\boldsymbol{x}}}$ where $\mathbf{u} \in \mathbb{R}^{N \cdot n_{\boldsymbol{u}}}$ denotes box-constrained inputs sequence and $\mathbf{x} \in \mathbb{R}^{N \cdot n_{\boldsymbol{x}}}$ denotes states trajectory vectors stacked as $\mathbf{x} = \left[\boldsymbol{x}_1^T, \dots, \boldsymbol{x}_{N-1}^T, \boldsymbol{x}_N^T\right]^T$, $\mathbf{u} = \left[\boldsymbol{u}_0^T, \dots, \boldsymbol{u}_{N-2}^T, \boldsymbol{u}_{N-1}^T\right]^T$

Presented class of problems defined by (2) *does impose* box input constraints. It is motivated by the fact that algorithm controls actuators, in common, which operate in a limited range (e.g., valve position) or limited range for rate of change (e.g., valve transition speed). The benefit is that box-constrained QP can be solved faster than a generally-constrained one. The presented approach *does not impose* hard state constraints. It prevents the feasibility issue. However, state limits can be imposed as soft constraints using the penalty method [10]. We believe that the class of problems defined by (2) is wide enough to cover the majority of industrial problems.

The assumption that have been made is $\begin{bmatrix}\mathbf{Q} & \mathbf{S}^T \\ \mathbf{S} & \mathbf{R}\end{bmatrix}$ is positive semidefinite. The individual vectors and matrices in (2a) are composed as follows

$$\mathbf{R} = \begin{bmatrix}\boldsymbol{R}_0 & & & \\ & \boldsymbol{R}_1 & & \\ & & \ddots & \\ & & & \boldsymbol{R}_{N-1}\end{bmatrix}, \ \mathbf{r}(\boldsymbol{x}_0) = \begin{bmatrix}\boldsymbol{S}_0\boldsymbol{x}_0 + \boldsymbol{r}_0 \\ \vdots \\ \boldsymbol{r}_{N-2} \\ \boldsymbol{r}_{N-1}\end{bmatrix}, \ \mathbf{q} = \begin{bmatrix}\boldsymbol{q}_1 \\ \vdots \\ \boldsymbol{q}_{N-1} \\ \boldsymbol{q}_N\end{bmatrix},$$

$$\mathbf{S} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{S}_1 & & \\ & \ddots & & \\ & & \boldsymbol{S}_{N-1} & \\ & & & \boldsymbol{0}\end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix}\boldsymbol{Q}_1 & & & \\ & \ddots & & \\ & & \boldsymbol{Q}_{N-1} & \\ & & & \boldsymbol{Q}_N\end{bmatrix},$$

with a constant term $c(\boldsymbol{x}_0) = \frac{1}{2}\boldsymbol{x}_0^T \boldsymbol{Q}_0 \boldsymbol{x}_0 + \boldsymbol{q}_0^T \boldsymbol{x}_0$ which does not influence the minimizer. The system dynamics (2b) and box-constraints (2c) are given by

$$\mathbf{A} = \begin{bmatrix}\boldsymbol{I} & & & \\ -\boldsymbol{A}_1 & \boldsymbol{I} & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{A}_{N-1} & \boldsymbol{I}\end{bmatrix}, \ \mathbf{w}(\boldsymbol{x}_0) = \begin{bmatrix}\boldsymbol{A}_0\boldsymbol{x}_0 + \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_{N-1}\end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix}\boldsymbol{B}_0 & & & \\ & \boldsymbol{B}_1 & & \\ & & \ddots & \\ & & & \boldsymbol{B}_{N-1}\end{bmatrix}, \ \underline{\mathbf{u}} = \begin{bmatrix}\underline{\boldsymbol{u}}_0 \\ \underline{\boldsymbol{u}}_1 \\ \vdots \\ \underline{\boldsymbol{u}}_{N-1}\end{bmatrix}, \ \overline{\mathbf{u}} = \begin{bmatrix}\overline{\boldsymbol{u}}_0 \\ \overline{\boldsymbol{u}}_1 \\ \vdots \\ \overline{\boldsymbol{u}}_{N-1}\end{bmatrix}.$$

Notice that $\mathbf{A} \in \mathbb{R}^{N \cdot n_{\boldsymbol{x}} \times N \cdot n_{\boldsymbol{x}}}$ is invertible by construction.

In the following, move-blocking and partial-condensing procedures are recalled. Then they are both incorporated in transformation in Section VI, leading to the generalized QP formulation.

IV. Move Blocking

Move Blocking (MB) is commonly used to deal with the computational burden in optimal control. The strategy is to fix an input or a change between two consecutive inputs to be constant for several time-steps [14]. Thus, the number of degrees of freedom in the optimization problem is reduced significantly for dense QP formulation.

The choice of the blocking strategy to provide robust control performance is provided in [17], where a generalized blocked variable-horizon MPC is formulated. The optimal blocking strategy is proposed in [18]. Therein, the optimality is measured regarding controller complexity and region of attraction volume and requires a solution to mixed-integer

programming. Once a move-blocking strategy is chosen, however, it hasn't been shown how to decrease the degree of freedom in the optimization problem for sparse QP formulation.

The level of blocking is parametrized by move-bloking vector $\boldsymbol{m}_{\mathrm{MB}} = [m_1, m_2, \ldots, m_{N_\mathbf{u}}] \in \mathbb{N}^{N_\mathbf{u}}$, where m are sizes of blocking windows and $N_\mathbf{u}$ is number of input vectors after blocking to be optimized, and for which $\mathrm{sum}(\boldsymbol{m}_{\mathrm{MB}}) = N$. For sake of brevity, an auxiliary vector of row indicies $\boldsymbol{j}_{\mathrm{MB}} = \mathrm{cumsum}^1(\boldsymbol{m}_{\mathrm{MB}}) = [j_1, j_2, \ldots, j_{N_\mathbf{u}}]$ is defined. Then *blocking* matrix $\mathbf{T}$ and and input vectors after blocking $\tilde{\mathbf{u}}$ to be optimized are

$$\mathbf{T} = \begin{bmatrix} \boldsymbol{I} & & \\ \vdots & & \\ \boldsymbol{I}_{j_1} & & \\ & \boldsymbol{I} & \\ & \vdots & \\ & \boldsymbol{I}_{j_2} & \\ & & \ddots \end{bmatrix} \in \mathbb{R}^{N\cdot n_{\boldsymbol{u}} \times N_\mathbf{u}\cdot n_{\boldsymbol{u}}}, \quad \tilde{\mathbf{u}} = \begin{bmatrix} \tilde{\boldsymbol{u}}_{j_1} \\ \tilde{\boldsymbol{u}}_{j_2} \\ \vdots \\ \tilde{\boldsymbol{u}}_{j_{N_\mathbf{u}}} \end{bmatrix} \in \mathbb{R}^{N_\mathbf{u}\cdot n_{\boldsymbol{u}}},$$

respectively. The blocking is provided by input transformation

$$\mathbf{u} = \mathbf{T}\tilde{\mathbf{u}}. \tag{3}$$

Note that $\mathbf{T}$ is matrix of ones and zeros only such that $\mathbf{T}^T\mathbf{T} = \mathrm{diag}(\boldsymbol{m}_{\mathrm{MB}}) \otimes \mathbf{I}$. For the admissible $\mathbf{T}$, an identity on each following row must be in the same column or the next right column. Move blocking *approximates* the original problem with one with a lesser number of degrees of freedom. The approximation effect on the control performance has been discussed, e.g., in [16]. Therein, the authors suggest a heuristic method to adapt the blocking strategy online such that control performance remains nearly unchanged. This paper focuses rather on computational and memory aspects.

In this paper, we modify [1] to allow the state condensing works for move-blocked MPC.

V. State condensing

The state condensing was introduced in [1]. The idea is to eliminate not every, but only some of the state vectors from along the prediction horizon. This will result in an optimization problem where equality constraints representing system dynamics are eliminated out only partially. The idea is to take advantage of both sparse and dense QP formulation, as in the partially-condensed problem, some sparsity structure remains, and simultaneously, the number of variables is reduced.

The level of condensing is parametrized by a state condensing vector $\boldsymbol{p}_{\mathrm{PC}} = [p_1, p_2, \ldots, p_{N_\mathbf{x}}] \in \mathbb{N}^{N_\mathbf{x}}$, where p are sizes of condensing windows and $N_\mathbf{x}$ is number of states vectors after condensing. For sake of brevity, a vector of row indicies $\boldsymbol{i}_{\mathrm{PC}} = \mathrm{cumsum}(\boldsymbol{p}_{\mathrm{PC}}) = [i_1, i_2, \ldots, i_{N_\mathbf{x}}]$ is defined. Possible option is $N_\mathbf{x} = 0$, i.e. $\boldsymbol{p}_{\mathrm{PC}}$ is empty vector which results in dense formulation. Then *condensing* matrix and state vectors after condensing to be optimized are

$$\mathbf{E} = \begin{bmatrix} \boldsymbol{I}_{i_1} & & \\ \mathbf{0} & & \\ \vdots & & \\ & \boldsymbol{I}_{i_2} & \\ & \mathbf{0} & \\ & \vdots & \\ & & \ddots \end{bmatrix} \in \mathbb{R}^{N\cdot n_{\boldsymbol{x}} \times N_\mathbf{x}\cdot n_{\boldsymbol{x}}}, \quad \tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\boldsymbol{x}}_{i_1} \\ \tilde{\boldsymbol{x}}_{i_2} \\ \vdots \\ \tilde{\boldsymbol{x}}_{i_{N_\mathbf{x}}} \end{bmatrix} \in \mathbb{R}^{N_\mathbf{x}\cdot n_{\boldsymbol{x}}},$$

respectively. The remaining states $\tilde{\mathbf{x}}$ and states being condensed out $\bar{\mathbf{x}}$ are separable as

$$\mathbf{E}\tilde{\mathbf{x}} + \mathbf{F}\bar{\mathbf{x}} = \mathbf{x}, \mathbf{E}^T\mathbf{F} = \mathbf{F}^T\mathbf{E} = \mathbf{0} \implies \mathbf{E}\mathbf{E}^T\mathbf{x} = \mathbf{E}\tilde{\mathbf{x}} \tag{4}$$

where

$$\mathbf{F} = \begin{bmatrix} \vdots & & & & & & \\ \boldsymbol{I}_{\bar{i}_1} & & & & & & \\ & \ddots & & & & & \\ & & \boldsymbol{I}_{\bar{i}_1} & & & & \\ & & \mathbf{0} & & & & \\ & & & \boldsymbol{I}_{\bar{i}_2} & & & \\ & & & & \ddots & & \\ & & & & & \boldsymbol{I}_{\bar{i}_2} & \\ & & & & & \mathbf{0} & \\ & & & & & & \ddots \end{bmatrix} \in \mathbb{R}^{N\cdot n_{\boldsymbol{x}} \times (N-N_\mathbf{x})\cdot n_{\boldsymbol{x}}}, \quad \bar{\mathbf{x}} = \begin{bmatrix} \bar{\boldsymbol{x}}_{\bar{i}_1} \\ \bar{\boldsymbol{x}}_{\bar{i}_2} \\ \vdots \\ \bar{\boldsymbol{x}}_{\bar{i}_{N-N_\mathbf{x}}} \end{bmatrix} \in \mathbb{R}^{(N-N_\mathbf{x})\cdot n_{\boldsymbol{x}}},$$

The prediction (2b) can be decomposed on the rows related to the remaining states using $\mathbf{E}\mathbf{E}^T$ and the rest using $\mathbf{F}\mathbf{F}^T$, respectively, as

$$\mathbf{E}\mathbf{E}^T\mathbf{A}\mathbf{x} = \mathbf{E}\mathbf{E}^T\mathbf{B}\mathbf{u} + \mathbf{E}\mathbf{E}^T\mathbf{w}(\boldsymbol{x}_0), \tag{5}$$

$$\mathbf{F}\mathbf{F}^T\mathbf{A}\mathbf{x} = \mathbf{F}\mathbf{F}^T\mathbf{B}\mathbf{u} + \mathbf{F}\mathbf{F}^T\mathbf{w}(\boldsymbol{x}_0). \tag{6}$$

Adding (4) to (6) a partial state prediction can be written down

$$\begin{aligned} (\mathbf{E}\mathbf{E}^T + \mathbf{F}\mathbf{F}^T\mathbf{A})\mathbf{x} &= \mathbf{E}\tilde{\mathbf{x}} + \mathbf{F}\mathbf{F}^T\mathbf{B}\mathbf{u} + \mathbf{F}\mathbf{F}^T\mathbf{w}(\boldsymbol{x}_0), \\ \text{or} \qquad \mathbf{x} &= \mathbf{M}^{-1}\mathbf{E}\tilde{\mathbf{x}} + \mathbf{N}\mathbf{u} + \mathbf{b}, \end{aligned} \tag{7}$$

where

$$\begin{aligned} \mathbf{M} &= \mathbf{E}\mathbf{E}^T + \mathbf{F}\mathbf{F}^T\mathbf{A}, \\ \mathbf{N} &= \mathbf{M}^{-1}\mathbf{F}\mathbf{F}^T\mathbf{B}, \\ \mathbf{b} &= \mathbf{M}^{-1}\mathbf{F}\mathbf{F}^T\mathbf{w}(\boldsymbol{x}_0). \end{aligned}$$

The state condensing exploits banded structure of $\mathbf{A}$. For the sake of brevity, the structure of the $\mathbf{M}$ matrix follows

$$\mathbf{M} = \begin{bmatrix} \mathbf{A}_{0,p_1} & & \\ & \mathbf{A}_{i_1,p_2} & \\ & & \ddots \end{bmatrix}, \ \mathbf{A}_{i,p} = \begin{bmatrix} \mathbf{I} & & & \\ \mathbf{A}_{i+1} & \mathbf{I} & & \\ & \ddots & \ddots & \\ & & \mathbf{A}_{i+p} & \mathbf{I} \end{bmatrix}.$$

Note that $\mathbf{M}$ remains invertible, moreover, $\mathbf{M}^{-1}$ is also unit lower triangular for any admissible $\mathbf{E}$.

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{A}_{0,p_1}^{-1} & & \\ & \mathbf{A}_{i_1,p_2}^{-1} & \\ & & \ddots \end{bmatrix}, \ \mathbf{A}_{i,p}^{-1} = \begin{bmatrix} \mathbf{I} & & & \\ \mathbf{A}_{i+1} & \mathbf{I} & & \\ \vdots & \ddots & \ddots & \\ \prod_{k=i+1}^{i+p}\mathbf{A}_k & \cdots & \mathbf{A}_{i+p} & \mathbf{I} \end{bmatrix}. \tag{8}$$

Substituting $\mathbf{x}$ in (5) by (7) leads to a new equality constraint

$$\mathbf{E}^T\mathbf{A}\mathbf{M}^{-1}\mathbf{E}\tilde{\mathbf{x}} = \mathbf{E}^T(\mathbf{B} - \mathbf{A}\mathbf{N})\mathbf{u} + \mathbf{E}^T(\mathbf{w}(\boldsymbol{x}_0) - \mathbf{A}\mathbf{b}). \tag{9}$$

[1] Function *cumsum* returns vector of cumulative sum of input argument.

Note that for fully sparse case $\mathbf{E} = \mathbf{I}, \mathbf{F}$ is empty implies $\mathbf{M}, \mathbf{N}, \mathbf{b}$ are empty matrices of particular size, i.e. the transformation is not needed at all. On the other hand, for dense case $\mathbf{E}$ is empty, $\mathbf{F} = \mathbf{I}$ implies $\mathbf{M} = \mathbf{A}, \mathbf{N} = \mathbf{A}^{-1}\mathbf{B}, \mathbf{b} = \mathbf{A}^{-1}\mathbf{w}(\boldsymbol{x}_0)$ and (7) yields an ordinary prediction. Further, only (5) and (7) are going to be used.

State condensing transform the original problem into an *equivalent* one of a smaller dimension. The state-condensing procedure benefits from the fact that $\mathbf{M}$ is block-diagonal. Therefore, it can be computed block by block, and off-diagonal terms remain zero.

VI. Generalized QP Formulation

By interconnectiong both previous methods, namely move blocking (3) and state condensing (4),(7), a systematic transformation can be written down now

$$\mathbf{x} = \boldsymbol{\Upsilon}\tilde{\mathbf{x}} + \boldsymbol{\Gamma}\tilde{\mathbf{u}} + \mathbf{b}, \tag{10a}$$
$$\mathbf{u} = \mathbf{T}\tilde{\mathbf{u}}. \tag{10b}$$

where $\boldsymbol{\Upsilon} = \mathbf{M}^{-1}\mathbf{E}$ and $\boldsymbol{\Gamma} = \mathbf{N}\mathbf{T}$.

The problem (2) can be then transformed using (10) into a generalized QP (11) of the similar structure. The transformation (10) is applied on (2) such that (10a) and (10b) are substitute in (2). Then the resulting generalized QP problem is

$$\begin{aligned} \min_{\tilde{\mathbf{u}},\tilde{\mathbf{x}}} \quad & \frac{1}{2}\begin{bmatrix}\tilde{\mathbf{u}}\\ \tilde{\mathbf{x}}\end{bmatrix}^T \begin{bmatrix}\tilde{\mathbf{R}} & \tilde{\mathbf{S}}^T\\ \tilde{\mathbf{S}} & \tilde{\mathbf{Q}}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{u}}\\ \tilde{\mathbf{x}}\end{bmatrix} + \begin{bmatrix}\tilde{\mathbf{u}}\\ \tilde{\mathbf{x}}\end{bmatrix}^T \begin{bmatrix}\tilde{\mathbf{r}}(\boldsymbol{x}_0)\\ \tilde{\mathbf{q}}(\boldsymbol{x}_0)\end{bmatrix} + \tilde{c}(\boldsymbol{x}_0)\\ \text{s.t.} \quad & \tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{B}}\tilde{\mathbf{u}} + \tilde{\mathbf{w}}(\boldsymbol{x}_0),\\ & \underline{\tilde{\mathbf{u}}} \le \tilde{\mathbf{u}} \le \overline{\tilde{\mathbf{u}}}, \end{aligned} \tag{11}$$

where

$$\begin{aligned} \tilde{\mathbf{R}} &= \mathbf{T}^T\mathbf{R}\mathbf{T} + \boldsymbol{\Gamma}^T(\mathbf{Q}\boldsymbol{\Gamma} + \mathbf{S}\mathbf{T}) + (\mathbf{S}\mathbf{T})^T\boldsymbol{\Gamma},\\ \tilde{\mathbf{S}} &= \boldsymbol{\Upsilon}^T(\mathbf{Q}\boldsymbol{\Gamma} + \mathbf{S}\mathbf{T}),\\ \tilde{\mathbf{Q}} &= \boldsymbol{\Upsilon}^T\mathbf{Q}\boldsymbol{\Upsilon}, \qquad & \tilde{\mathbf{r}}(\boldsymbol{x}_0) &= \mathbf{T}^T\mathbf{r}(\boldsymbol{x}_0) + (\mathbf{S}\mathbf{T})^T\mathbf{b} + \boldsymbol{\Gamma}^T(\mathbf{q} + \mathbf{Q}\mathbf{b}),\\ \tilde{\mathbf{A}} &= \mathbf{E}^T\mathbf{A}\boldsymbol{\Upsilon}, & \tilde{\mathbf{q}}(\boldsymbol{x}_0) &= \boldsymbol{\Upsilon}^T(\mathbf{q} + \mathbf{Q}\mathbf{b}),\\ \tilde{\mathbf{B}} &= \mathbf{E}^T(\mathbf{B}\mathbf{T} - \mathbf{A}\boldsymbol{\Gamma}), & \tilde{\mathbf{w}}(\boldsymbol{x}_0) &= \mathbf{E}^T(\mathbf{w}(\boldsymbol{x}_0) - \mathbf{A}\mathbf{b}),\\ \overline{\tilde{\mathbf{u}}} &= \mathbf{T}^+\overline{\mathbf{u}}, & \tilde{c}(\boldsymbol{x}_0) &= c(\boldsymbol{x}_0) + 0.5\mathbf{b}^T(\mathbf{q} + \mathbf{Q}\mathbf{b}),\\ \underline{\tilde{\mathbf{u}}} &= \mathbf{T}^+\underline{\mathbf{u}}, & \tilde{\mathbf{u}}_0 &= \mathbf{T}^+\mathbf{u}_0, \end{aligned} \tag{12}$$

and where $\mathbf{T}^+$ denotes left-inverse of $\mathbf{T}$. When the data for (2) are available a generalized QP formulation QP (11) is given by (12) parametrized by $\mathbf{T}$ and $\mathbf{E}$. Matrices in (12) can be build efficiently with respect the sparsity structure and Floating Point Operations (FLOPs) count can be obtained easily. Remember, any admissible choice of blocking matrix (except $\mathbf{T} = \mathbf{I}$) causes (11) *approximates* (2) by problem with less degree of freedom. On the other hand, any admissible $\mathbf{E}$ does not affect the minimization result.

Problem (11) can be solved efficiently by an active-set or an interior-point method. Note that both $\mathrm{LDL}^{\mathrm{T}}$ and Cholesky decomposition typically used to find a Newton step within any of these methods preserves the sparsity pattern of this problem.

VII. Numerical Examples

For the sake of brevity, transformation matrices for a short prediction horizon problem setup are shown first. It is followed by a common benchmark – the oscillating masses controlled by move-blocked MPC with a given blocking strategy. Finally, for a given system and prediction horizon, the whole transformation space is sampled to demonstrate the behavior of the proposed problem setups in more detail.

A. Illustrative Example

Assume a random LTV system with ten states ($n_{\boldsymbol{x}} = 10$) and five inputs ($n_{\boldsymbol{u}} = 5$) for which sequence of move blocks was given to be $\boldsymbol{m}_{\mathrm{MB}} = [1, 2, 3]$, consequently, $\boldsymbol{j}_{\mathrm{MB}} = [1, 3, 6]$. Prediction horizon is $N = \mathrm{sum}(\boldsymbol{m}_{\mathrm{MB}}) = 6$ and control horizon is $N_{\mathbf{u}} = \mathrm{length}(\boldsymbol{m}_{\mathrm{MB}}) = 3$. Let's choose $\boldsymbol{p}_{\mathrm{PC}} = [1, 2, 3]$ to be similar to the $\boldsymbol{m}_{\mathrm{MB}}$ in the first $N_{\mathbf{u}} - 1$ entries, consequently, $\boldsymbol{j}_{\mathrm{PC}} = [1, 3, 6]$. In this case, the transformation matrices in (3) and (4) are given by

$$\mathbf{T} = \begin{bmatrix} \boldsymbol{I} & & \\ & \boldsymbol{I} & \\ & \boldsymbol{I} & \\ & & \boldsymbol{I} \\ & & \boldsymbol{I} \\ & & \boldsymbol{I} \end{bmatrix}, \quad \tilde{\mathbf{u}} = \begin{bmatrix}\tilde{\boldsymbol{u}}_0\\ \tilde{\boldsymbol{u}}_1\\ \tilde{\boldsymbol{u}}_3\end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \boldsymbol{I} & & \\ & \boldsymbol{0} & \\ & \boldsymbol{I} & \\ & & \boldsymbol{0} \\ & & \boldsymbol{0} \\ & & \boldsymbol{I} \end{bmatrix}, \quad \tilde{\mathbf{x}} = \begin{bmatrix}\tilde{\boldsymbol{x}}_1\\ \tilde{\boldsymbol{x}}_3\\ \tilde{\boldsymbol{x}}_6\end{bmatrix},$$

$$\mathbf{F} = \begin{bmatrix} \boldsymbol{0} & & \\ \boldsymbol{I} & & \\ \boldsymbol{0} & & \\ & \boldsymbol{I} & \\ & & \boldsymbol{I} \\ & & \boldsymbol{0} \end{bmatrix}, \quad \bar{\mathbf{x}} = \begin{bmatrix}\bar{\boldsymbol{x}}_2\\ \bar{\boldsymbol{x}}_4\\ \bar{\boldsymbol{x}}_5\end{bmatrix},$$

Consequently, condensing matrices in (7) and (10) are

$$\mathbf{M} = \begin{bmatrix} \boldsymbol{I} & & & & & \\ -\boldsymbol{A}_1 & \boldsymbol{I} & & & & \\ & & \boldsymbol{I} & & & \\ & & -\boldsymbol{A}_3 & \boldsymbol{I} & & \\ & & & -\boldsymbol{A}_4 & \boldsymbol{I} & \\ & & & & & \boldsymbol{I} \end{bmatrix}, \quad \mathbf{N} = \begin{bmatrix} \boldsymbol{0} & & & & & \\ & \boldsymbol{B}_1 & & & & \\ & & \boldsymbol{0} & & & \\ & & & \boldsymbol{B}_3 & & \\ & & & \boldsymbol{A}_4\boldsymbol{B}_3 & \boldsymbol{B}_4 & \\ & & & & & \boldsymbol{0} \end{bmatrix},$$

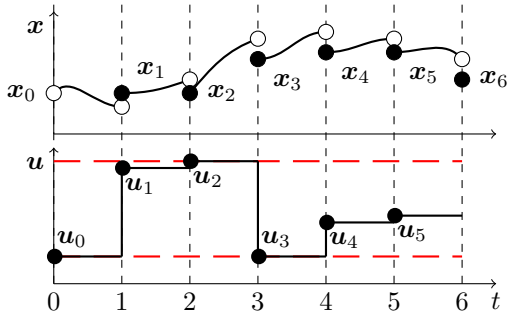
$$\mathbf{M}^{-1} = \begin{bmatrix} \boldsymbol{I} & & & & & \\ \boldsymbol{A}_1 & \boldsymbol{I} & & & & \\ & & \boldsymbol{I} & & & \\ & & \boldsymbol{A}_3 & \boldsymbol{I} & & \\ & & \boldsymbol{A}_4\boldsymbol{A}_3 & \boldsymbol{A}_4 & \boldsymbol{I} & \\ & & & & & \boldsymbol{I} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{w}_2 \\ \boldsymbol{0} \\ \boldsymbol{w}_4 \\ \boldsymbol{A}_4\boldsymbol{w}_3 + \boldsymbol{w}_4 \\ \boldsymbol{0} \end{bmatrix}.$$

The sparsity structure of the resulting QP problem can be demonstrated at the Karush–Kuhn–Tucker (KKT) matrix of QP problem (11) with no inequality constraints being activated. The associated KKT matrix is

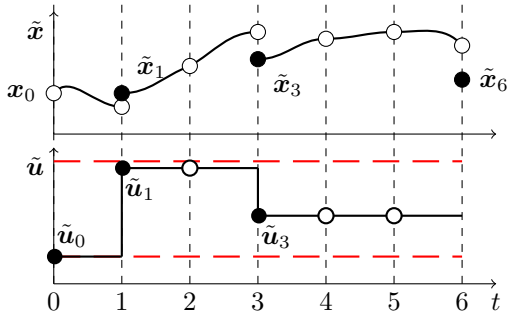
$$\tilde{\mathbf{H}} = \begin{bmatrix} \tilde{\mathbf{R}} & \tilde{\mathbf{S}}^T & \tilde{\mathbf{B}}^T \\ \tilde{\mathbf{S}} & \tilde{\mathbf{Q}} & \tilde{\mathbf{A}}^T \\ \tilde{\mathbf{B}} & \tilde{\mathbf{A}} & \mathbf{0} \end{bmatrix}. \tag{13}$$

The example is sketched in Fig. 1 and sparsity pattern of the associated KKT matrix (13) in Fig. 2.

This simple example illustrates (see Fig. 2) that the structure pattern of the problem is invariant for the proposed transformation. Also notice, the transformed problem has a smaller dimension, and less than half non-zero elements; therefore, it is expected the transformed problem requires less computational effort to a solution.

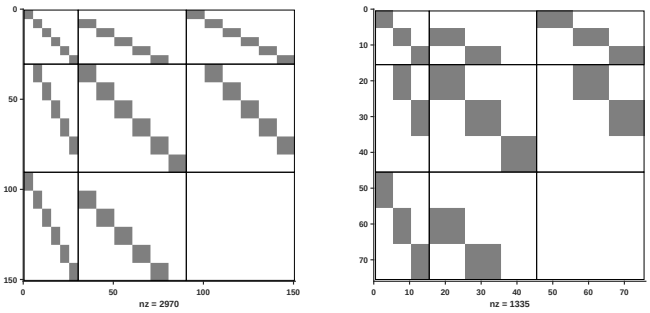


(a) Original



(b) Transformed

Fig. 1: Sketch of the optimization problem. Comparison of the original and transformed problem. ● denotes vectors included in and ○ vectors excluded from the optimization.



(a) Original (b) Transformed

Fig. 2: Sparsity patterns of the KKT matrix $\tilde{\mathsf{H}}$ associated with a short prediction horizon example. All inequality constraints are assumed to be inactive.

B. Oscillating Masses

This benchmark is inspired by [20]. The proposed system consists of a sequence of six masses connected to each other by spring dampers. The first and the last masses are connected to the walls. The weight of each mass is 1 kg and the spring constant is 1 N/m without damping. The system state $\boldsymbol{x} \in \mathbb{R}^{12}$ represents the displacement and velocity of an individual mass. There are four control inputs, i.e., $\boldsymbol{u} \in \mathbb{R}^4$, which exert tensions between different masses. We assume control limits $-0.5 \leq \boldsymbol{u} \leq 0.5$, and the presence of random bounded external disturbance $\boldsymbol{v} \in \mathbb{R}^6$ with a uniform distribution on $[-0.5, 0.5]$, which acts additionally on the displacement state of each mass. The control objective is to stabilize each mass in its origin, i.e., to solve (1) with $\boldsymbol{Q} = \boldsymbol{I}$, $\boldsymbol{R} = \boldsymbol{I}$, $\boldsymbol{S} = \boldsymbol{0}$, $\boldsymbol{q} = \boldsymbol{0}$, $\boldsymbol{r} = \boldsymbol{0}$, and with sampling time $T_s = 0.5$ s.

Further, we focus on the computational cost of problem (11) build (preparation phase) and its solution (feedback phase). This is typically studied in nonlinear MPC, where (11) has to be built and solved every sampling period [3]. The more QP problem is condensed and/or blocked, the more expansive the preparation is. On the other hand, the preparation may sufficiently decrease the solution time in particular. The preparation and feedback phase for some specific cases are examined numerically, and FLOPs are measured.

The feedback phase denotes the cost of problem (11) solution, in this paper, it is counted for *NPPsparse* solver [13]. This active-set-like method converges typically in several iterations. The method benefits from the use of warm/hot-start while the number of iterations is insensitive to the problem conditioning.

In this exemplary case, the move-blocking strategy was chosen to be

$$\boldsymbol{m}_{\mathrm{MB}} = [10, \ldots, 10] \in \mathbb{N}^{24}.$$

Once the blocking strategy is fixed, the control performance is immutable, i.e., all examined QPs are equivalent. For this setup, various levels of condensing are tested. Condensing vector is chosen such that any state vector is closing move-blocking series, more precisely

$$\boldsymbol{p}_{\mathrm{PC}} = \begin{cases} \text{empty vector}, \ i = 0, \\ [240/i, \ldots, 240/i] \in \mathbb{N}^i, \\ \qquad i \in \{1, 2, 3, 4, 5, 6, 8, 12, 15, 16, 20, 24\}, \end{cases}$$

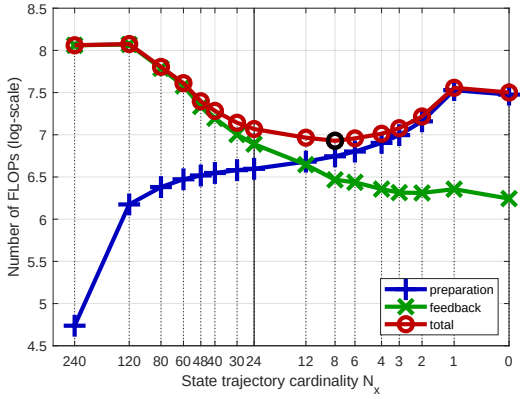
and

$$\boldsymbol{p}_{\mathrm{PC}} = [10, \ldots, 10] \in \mathbb{N}^{240/i}, i \in \{30, 40, 48, 60, 80, 120, 240\}$$

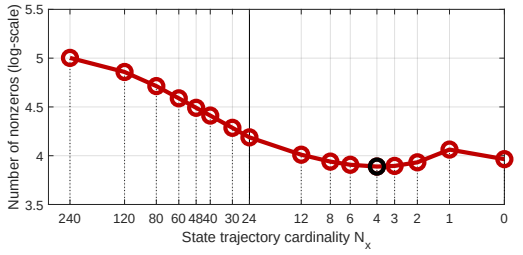
where in the later some of the block windows are additionally splitted in half to obtain the proper length i of vector $\boldsymbol{p}_{\mathrm{PC}}$. Resulting computational and memory requirements for this setup are shown in Fig. 3.

In Fig. 3, on the left ($N_{\mathsf{x}} = 240$) is the sparse QP and on the right ($N_{\mathsf{x}} = 0$) is the dense problem formulation. The first observation is that minimal computational or memory burden is in between these two. In other words, it is beneficial to condense the original sparse QP partially. Specifically, computational requirements of the optimally condensed ($\approx 10^7$FLOPs) compared to the dense QP ($\approx 10^{7.5}$FLOPs) is more than three times lower. One can save 20% of memory space in case of optimally condensed ($\approx 10^{3.9}$Number of non-zero elements (NNZs)) compared to the dense QP ($\approx 10^4$NNZs). Another observation is that preparation cost when no condensing is required is significantly smaller compared to any other level of condensing.

In general, the computational cost of the preparation phase grows with the level of condensing. On the other hand, the computational cost of the feedback phase is not monotone



(a) Computational requirements – number of FLOPs of NPPsparse [13].



(b) Memory requirements – number of nonzero elements in (13).

Fig. 3: Computational and memory requirements for equivalent QP representing different level of condensing of move-blocked MPC for oscillating masses. ○ denote optimal requirements.

and changes depending on multiple factors ($N_{\mathbf{x}}, N_{\mathbf{u}}, n_{\boldsymbol{x}}, n_{\boldsymbol{u}}$) and importantly, on a particular algorithm implementation.

The presented numerical experience illustrates that using generalized QP formulation (11), for a given move-blocked MPC, an optimization problem that requires minimal FLOPs can be found.

The proposed approach has been implemented and tested in MATLAB environment, which allows code generation to an embedded platform. There is no need for two separate pieces of code (for dense and sparse QP formulation), which labor-saving of code maintenance.

VIII. Conclusion

In this paper, we combine move-blocking and state-condensing procedures in generalized QP formulation of a move-blocked MPC problem. The combination of move-blocking and state-condensing methods allows reducing the number of input variables as well as a number of state variables. This approach allows for a MPC with a given move-blocking strategy to find such a QP formulation for which a total computational burden or memory footprint of the MPC regulator is minimal. It has been illustrated how the proposed transformation affects memory footprint and computational burden by numerical examples. We analyzed on the example, the computational burden could be significantly decreased ($\approx 3\times$) or memory saved ($\approx 20\%$). The proposed approach requires specialized QP solver used together with an optimized library for sparse linear algebra.

References

[1] Daniel Axehill. Controlling the level of sparsity in mpc. *Systems & Control Letters*, 76:1–7, 2015.

[2] Yutao Chen, Gianluca Frison, Niels van Duijkeren, Mattia Bruschetta, Alessandro Beghi, and Moritz Diehl. Efficient partial condensing algorithms for nonlinear model predictive control with partial sensitivity update. *IFAC-PapersOnLine*, 51(20):406 – 411, 2018. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[3] Moritz Diehl, Hans Georg Bock, and Johannes P. Schlder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. 43(5):1714–1736, 2005. Exported from https://app.dimensions.ai on 2018/12/13.

[4] Dimitar Dimitrov, Alexander Sherikov, and Pierre-Brice Wieber. A sparse model predictive control formulation for walking motion generation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, September, 2011*, pages 2292–2299, San Francisco, Etats-Unis, sep 2011. IEEE.

[5] M. Faroni, M. Beschi, M. Berenguel, and A. Visioli. Fast mpc with staircase parametrization of the inputs: Continuous input blocking. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2017.

[6] Janick V. Frasch, Milan Vukov, Hans Joachim Ferreau, and Moritz Diehl. A new quadratic programming strategy for efficient sparsity exploitation in sqp-based nonlinear mpc and mhe. *IFAC Proceedings Volumes*, 47(3):2945 – 2950, 2014. 19th IFAC World Congress.

[7] Gianluca Frison. Algorithms and methods for high-performance model predictive control. 2016.

[8] Ravi Gondhalekar and Jun ichi Imura. Least-restrictive move-blocking model predictive control. *Automatica*, 46(7):1234 – 1240, 2010.

[9] Juan L Jerez, Eric C Kerrigan, and George A Constantinides. A condensed and sparse qp formulation for predictive control. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 5217–5222. IEEE, 2011.

[10] Eric C. Kerrigan and Jan M. Maciejowski. Soft constraints and exact penalty functions in model predictive control. In *Proc. UKACC International Conference Control*, 2000.

[11] D. Kouzoupis, R. Quirynen, J.V. Frasch, and M. Diehl. Block condensing for fast nonlinear mpc with the dual newton strategy. *IFAC-PapersOnLine*, 48(23):26 – 31, 2015. 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015.

[12] D Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica J. IFAC*, 36(6):789–814, 2000.

[13] Pavel Otta, Jiří Burant, Ondřej Šantin, and Vladimír Havlena. Newton projection with proportioning using iterative linear algebra for model predictive control with long prediction horizon. *Optimization Methods and Software*, 34(5):1075–1098, 2019.

[14] Cagienard Raphael, Grieder Pascal, Kerrigan Eric C., and Manfred Morari. Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6):563 – 570, 2007.

[15] Stefan Richter, Sebastien Mariethoz, Manfred S. Richter Morari, S. Marithoz, and M. Morari. High-speed online mpc based on a fast gradient method applied to power converter control. In *Proceedings of the 2010 American Control Conference*, pages 4737–4743, June 2010.

[16] T. Schwickart, H. Voos, M. Darouach, and S. Bezzaoucha. A flexible move blocking strategy to speed up model-predictive control while retaining a high tracking performance. In *2016 European Control Conference (ECC)*, pages 764–769, June 2016.

[17] Rohan C. Shekhar and Jan M. Maciejowski. Robust variable horizon mpc with move blocking. *Systems & Control Letters*, 61(4):587 – 594, 2012.

[18] Rohan C. Shekhar and Chris Manzie. Optimal move blocking strategies for model predictive control. *Automatica*, 61:27 – 34, 2015.

[19] F. Ullmann. FiOrdOs: A Matlab toolbox for C-code generation for first order methods. Master thesis, Swiss Federal Institute of Technology Zurich, ETH, Zurich, 2011.

[20] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Trans. Control Syst. Technol.*, 18(2):267–278, 2010.