Structure Exploitation of Practical MPC Formulations for Speeding up First-Order Methods*

D. K. M. Kufoalor[1], S. Richter[2], L. Imsland[1], T. A. Johansen[1]

***Abstract*— This paper presents structure exploitation techniques that lead to faster convergence of first-order methods for practical Model Predictive Control (MPC) formulations. We exploit the special structure of output box constraints as well as input bound and rate constraints. The output constraints are included in the MPC objective as exact penalty functions, in order to avoid feasibility issues due to e.g. plant-model mismatch. Observations from a new derivation of exact penalty functions enable us to formulate exact penalty functions that do not require additional auxiliary variables if first-order solution methods are used. We use the dual fast gradient method to illustrate the effectiveness of our approach. An average speed-up of $\times 8$ and a worst case speed-up of $\times 6$ were obtained, compared with the fastest state-of-the-art first-order method for a subsea separation process. Moreover, hardware-in-the-loop simulations using an ANSI C implementation on a PLC reveal that our first-order solver outperforms the fastest second-order solver deployed for the subsea separation process.**

I. Introduction

A linear Model Predictive Control (MPC) problem is a constrained convex optimization problem, which translates to a convex quadratic programming (QP) problem due to the use of a quadratic objective and linear constraints. Existing methods that compute the QP solution online are mainly first-order or second-order iterative methods (classified according to the highest order of objective function information used).

First-order methods may be preferable in some embedded real-time applications since such methods typically yield simple and efficient code with relatively small memory requirements (see e.g. [1], [2], [3], [4]). Compared with second-order methods, first-order methods perform typically many, but cheap, iterations. The number of iterations can be significantly reduced using different techniques, such as preconditioning, structure exploitation, and warm-starting.

Earlier studies in [1] have revealed that several state-of-the-art first-order methods are unable to perform well on a class of practical MPC problems based on typical industrial design strategies. Such strategies include exact penalty formulations that ensure that the optimization problem does not become infeasible due to large process disturbances or plant-model mismatch. Also, in a practical input-output MPC formulation, some constrained outputs are not required to track any reference and may therefore lack corresponding quadratic weights in the objective. Rate constraints on inputs are also common, and the use of input move blocking or output evaluation points may lead to constraint definitions that vary over the horizon. When these practical design choices are present in the MPC setup, the standard translation into a QP does not facilitate full structure exploitation for first-order methods [5], and hence convergence speed is sacrificed. A standard QP translation involves a positive semi-definite Hessian and a possibly large number of auxiliary variables (see e.g. [1] for an industrial example).

In this paper, we suggest an alternative formulation of exact penalty functions for output box constraints in the context of first-order methods. Exact penalty functions [6, §5.4.5] replace the hard output constraints by a real-valued, typically nonsmooth function that is appended to the objective. While this measure 'softens' the hard output constraints, exact penalty functions are able to retain equivalence in terms of the optimal value and solution. This in contrast to so-called *inexact* penalty functions that can lead to different solutions although the original problem is feasible.

The finally proposed formulation of the MPC problem as a dual optimization problem hinges on the assumption that the projection onto the set of input bound and rate constraints can be computed cheaply. Due to space restrictions, we cannot elaborate on this important aspect here, however, we refer the reader to the extensive technical report [7].

The paper continues with a description of the proposed MPC formulation, followed by derivations that lead to our solution method. We use different benchmark studies to facilitate further discussions before concluding the paper.

II. MPC problem formulation

As a basis for our derivations we consider a setup that is motivated from industrial input-output MPC formulations:

$$\begin{aligned} \min_{\Delta u, y, z} \; & \Big(\frac{1}{2} \Delta u^T R \Delta u + \frac{1}{2} y^T Q y + l^T y + \sum_{j=1}^{n_y} p_j(y_j) + \sum_{j=1}^{n_z} q_j(z_j) \Big) \\ \text{s.t.} \quad & y = \Theta_y \Delta u + y_f \\ & z = \Theta_z \Delta u + z_f \\ & \Delta u_j \in \Delta \mathbb{U}(u_{j,-1}), \quad j \in \{1, 2, \ldots, n_u\}. \end{aligned} \tag{1}$$

The vector Δu is defined as a stack of individual changes in each of the n_u inputs over the control horizon N_u, i.e $\Delta u = (\Delta u_1, \Delta u_2, \ldots, \Delta u_{n_u})$ where $\Delta u_j = (\Delta u_{j,0}, \Delta u_{j,1}, \ldots, \Delta u_{j,N_u-1})$. We assume the input penalty matrix R to be block-diagonal with blocks $R_j = r_j \cdot I, \;\; r_j > 0, \;\; j \in \{1, 2, \ldots, n_u\}$. Input

[1] Center for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), O.S. Bragstads plass 2D N-7491 Trondheim, Norway. {kwame.kufoalor, lars.imsland, tor.arne.johansen}@ntnu.no

[2] Richter Optimization GmbH, Furttalstrasse 29, CH-8046 Zürich, Switzerland. sr@richteroptimization.com

* This work was supported by the Research Council of Norway (NFR) and Statoil through the project 215684 and by NFR through the projects 223254 and 244116/O70.

constraints are hard constraints, but never lead to infeasibility of the MPC problem. In (1), we restrict the input *changes* for each input to be in the convex set

$$\begin{aligned}\Delta\mathbb{U}(u_{-1}) \triangleq \big\{\Delta\upsilon \in \mathbb{R}^{N_u} | \Delta\upsilon_i = \upsilon_i - \upsilon_{i-1}, |\Delta\upsilon_i| \leq \overline{\Delta u}_i, \\ \underline{u} \leq \upsilon_i \leq \overline{u}, i \in \{0,\ldots,N_u-1\}, \upsilon_{-1} = u_{-1}\big\}.\end{aligned} \tag{2}$$

The set $\Delta\mathbb{U}(u_{-1})$ is parameterized by u_{-1}, the input applied at the past sampling instant, and is determined by the static input bounds $\underline{u}, \overline{u}$ and varying rate bounds $\overline{\Delta u}_i$.

Similarly, the outputs are defined as $y = (y_1, y_2, \ldots, y_{n_y})$, where $y_j = (y_{j,1}, y_{j,2}, \ldots, y_{j,N_y})$ and $z = (z_1, z_2, \ldots, z_{n_z})$, with $z_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,N_z})$. When output evaluation points are used, N_y (N_z) becomes the number of output evaluation points. The functions p_j and q_j are convex, nonsmooth exact penalty functions for the outputs, typically used in practical MPC formulations to 'soften' the hard output constraints. The definitions for p_j and q_j are discussed in detail in Section III. The interested reader might notice that there is an additional linear and quadratic term in the outputs y whereas there are no such terms in z. The reason for this is that the outputs y are assumed to be regulated to a given reference, where deviations are penalized quadratically by the positive diagonal matrix Q and the reference is encoded in vector l.

The prediction model is formulated as affine equality constraints that relate input changes to outputs, using the free response y_f (z_f), i.e. the effect of previously applied inputs on the future output, and the forced response $\Theta_y \Delta u$ ($\Theta_z \Delta u$). The model parameters can either be derived from a state space model of the dynamics or obtained from experimental data, e.g. step response tests (see e.g. [8]).

In summary, problem (1) is a multi-parametric convex program with parameters $y_f \in \mathbb{R}^{n_y \cdot N_y}, z_f \in \mathbb{R}^{n_z \cdot N_z}$ and $u_{j,-1} \in \mathbb{R}$, $j \in \{1,2,\ldots,n_u\}$. In case of changing output references, vector l in the linear term becomes a parameter too.

III. Representation of Exact Penalty Functions for First-Order Methods

In this section we will investigate the representation of the exact penalty functions p_j and q_j in (1). The proposed representation comes without the commonly introduced auxiliary variables in standard QP reformulations of (1) and is a consequence of the following theorem.

Theorem 1: Consider the convex program

$$\begin{aligned}f^* \triangleq \min_{x \in \mathbb{X}} \; & f(x) \\ \text{s.t. } & g_j(x) \leq 0, \quad j = 1,2,\ldots,m,\end{aligned} \tag{3}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g_j : \mathbb{R}^n \to \mathbb{R}, j = 1,2,\ldots,m$, are convex real-valued functions and $\mathbb{X}$ is a closed convex subset of $\mathbb{R}^n$. Assume that an optimal solution x^* exists, i.e. $f(x^*) = f^*$, strong duality holds and a Lagrange multiplier vector $\mu^* \in \mathbb{R}^m_+$ for the inequality constraints exists. If the penalty parameter is chosen as $\sigma \geq \|\mu^*\|_\infty$, then it holds that

$$f^* = \min_{x \in \mathbb{X}} \Big(f(x) + \sigma \cdot \sum_{j=1}^{m} \max\{0, g_j(x)\} \Big). \tag{4}$$

Proof: Let $g(x) \triangleq (g_1(x), g_2(x), \ldots, g_m(x))$, for simplicity of notation. By strong duality and existence of a Lagrange multiplier we have

$$\begin{aligned}f^* = \min_{x \in \mathbb{X}} \max_{\mu \geq 0} \big(f(x) + \mu^T g(x) \big) = \max_{\mu \geq 0} \min_{x \in \mathbb{X}} \big(f(x) + \mu^T g(x) \big) = \\ \max_{\mathbf{0} \leq \mu \leq \sigma \mathbf{1}} \min_{x \in \mathbb{X}} \big(f(x) + \mu^T g(x) \big)\end{aligned} \tag{5}$$

Using the max-min inequality, the right hand side in (5) can be upper bounded by

$$\min_{x \in \mathbb{X}} \max_{\mathbf{0} \leq \mu \leq \sigma \mathbf{1}} \big(f(x) + \mu^T g(x) \big), \tag{6}$$

and the min-max expression on the left hand side in (5) can be lower bounded by the same expression in (6). Therefore, (5) can be written as

$$f^* = \min_{x \in \mathbb{X}} \Big(f(x) + \max_{\mathbf{0} \leq \mu \leq \sigma \mathbf{1}} \; \mu^T g(x) \Big),$$

which is equivalent to (4). ∎

Remark 1: To the best of the authors' knowledge, the proof is new in terms of the arguments made and seems to be the most concise one. The result itself, however, has long existed in the literature (see e.g. [6, §5.4.5]).

Whereas Theorem 1 establishes equivalence of the optimal values, the following result gives a sufficient condition for equivalence of the optimal solutions.

Proposition 1: If the penalty parameter is $\sigma > \|\mu^*\|_\infty$, then the solution sets of (3) and (4) coincide.

Proof: Due to space restrictions, we refer the reader to the proof of [7, Proposition 3], which establishes this and related results under an arbitrary l_p-norm. ∎

Theorem 1 is commonly cast in terms of an epigraph reformulation of (4) in the literature, i.e.

$$\begin{aligned}f^* = \min_{x \in \mathbb{X}} \; & f(x) + \sigma \cdot \sum_{j=1}^{m} \varepsilon_j \\ \text{s.t. } & g_j(x) \leq \varepsilon_j, \quad \varepsilon_j \geq 0, \quad j = 1,2,\ldots,m.\end{aligned} \tag{7}$$

The advantage of this representation is that the problem class remains unchanged, e.g. if the original problem (3) is a QP, then (7) will remain a QP and can be passed to a general purpose solver. If first-order methods are used, however, it is convenient to work with the formulation (4) (cf. Section IV).

Remark 2: The penalty parameter σ for a *multi-parametric* program as given by (1) is determined empirically in practice.

IV. Structure Exploiting Lagrange Relaxation

In this section we present the consequences of the exact penalty representation in the previous section when it comes to solving (1) in the dual domain using a first-order method. In particular, we apply Lagrange relaxation for the equality constraints in (1) and show how to efficiently evaluate the gradient of the resulting dual function in case of softened output box constraints.

For this, we introduce the dual multipliers $\lambda \in \mathbb{R}^{n_y \cdot N_y}$ and $\nu \in \mathbb{R}^{n_z \cdot N_z}$ and derive the dual problem

$$\max_{\lambda, \nu} d(\lambda, \nu), \tag{8}$$

where the dual function is given by

$$d(\lambda,\nu) = \underbrace{\min_{\substack{\Delta u_j \in \Delta\mathbb{U}(u_{j,-1}),\\ j\in\{1,2,\dots,n_u\}}} \tfrac{1}{2}\Delta u^T R\Delta u - \lambda^T\Theta_y\Delta u - \nu^T\Theta_z\Delta u}_{\triangleq d_{\Delta u}(\lambda,\nu)}$$
$$+ \underbrace{\min_y \tfrac{1}{2}y^T Qy + l^T y + \sum_{j=1}^{n_y} p_j(y_j) + \lambda^T y}_{\triangleq d_y(\lambda)}$$
$$+ \underbrace{\min_z \Big(\sum_{j=1}^{n_z} q_j(z_j) + \nu^T z\Big)}_{\triangleq d_z(\nu)} - \lambda^T y_f - \nu^T z_f. \tag{9}$$

Next, we investigate each of the terms in the dual function.

Term $d_{\Delta u}(\lambda,\nu)$: Due to strong convexity of the objective function ($R \succ 0$), the minimizer $\Delta u^*(\lambda,\nu)$ is unique and hence, by Danskin's Theorem, this term is differentiable:

$$\nabla_\lambda d_{\Delta u}(\lambda,\nu) = -\Theta_y\Delta u^*(\lambda,\nu),$$
$$\nabla_\nu d_{\Delta u}(\lambda,\nu) = -\Theta_z\Delta u^*(\lambda,\nu),$$

where $\Delta u^*(\lambda,\nu) = \big(\Delta u_1^*(\lambda,\nu),\dots,\Delta u_{n_u}^*(\lambda,\nu)\big)$ and, since R is block-diagonal, the minimization can be separated over the n_u inputs, i.e. for all $j\in\{1,2,\dots,n_u\}$

$$\Delta u_j^*(\lambda,\nu) = \arg\min_{\Delta u_j\in\Delta\mathbb{U}(u_{j,-1})} \frac{r_j}{2}\Delta u_j^T\Delta u_j - (\Theta_y^T\lambda+\Theta_z^T\nu)_j^T\Delta u_j$$
$$= \pi_{\Delta\mathbb{U}(u_{j,-1})}\Big(\frac{1}{r_j}\big(\Theta_y^T\lambda+\Theta_z^T\nu\big)_j\Big), \tag{10}$$

where $\big(\Theta_y^T\lambda+\Theta_z^T\nu\big)_j$ denotes the jth block of components of vector $\Theta_y^T\lambda+\Theta_z^T\nu$ that corresponds to the input change Δu_j. Our simulation studies suggest that computing the projection $\pi_{\Delta\mathbb{U}(u_{j,-1})}(\cdot)$ in (10) *approximately* by means of a proximal gradient method produces a high quality solution adequate for MPC. We refer the reader to [7] for a detailed analysis of different projection methods for set $\Delta\mathbb{U}(u_{j,-1})$.

Term $d_y(\lambda)$: By similar reasoning as before, this term is differentiable and the gradient is $\nabla_\lambda d_y(\lambda) = y^*(\lambda)$, where $y^*(\lambda) = \Big(y_1^*(\lambda_1),\dots,y_{n_y}^*(\lambda_{n_y})\Big)$, if the dual multiplier is partitioned according to the outputs as $\lambda = (\lambda_1,\dots,\lambda_{n_y})$ with $\lambda_j\in\mathbb{R}^{N_y}$. Separating the minimization over the outputs yields

$$y_j^*(\lambda_j) = \arg\min_{y_j}\Big(\frac{1}{2}y_j^T Q_j y_j + l_j^T y_j + p_j(y_j) + \lambda_j^T y_j\Big), \tag{11}$$

for all $j\in\{1,2,\dots,n_y\}$, where Q_j is the jth diagonal submatrix of Q, l_j denotes the jth block of components of vector l, and $p_j(y_j)$ encodes an exact penalty function for lower, upper or joint lower/upper bound constraints on the output y_j. We summarize the computation of $y_j^*(\lambda_j)$ next.

Proposition 2: Consider the minimization problem in (11), using the exact penalty form in (4). The solution to (11) for each of the following types of bound constraints is:

- *Lower bound* ($\underline{y}_j \le y_j$):

$$y_j^*(\lambda_j) = -Q_j^{-1}\big(l_j+\lambda_j-\pi_{[0,\gamma]^{N_y}}(Q_j\underline{y}_j+l_j+\lambda_j)\big)$$

- *Upper bound* ($y_j \le \bar{y}_j$):

$$y_j^*(\lambda_j) = -Q_j^{-1}\big(l_j+\lambda_j+\pi_{[0,\gamma]^{N_y}}\big(-(Q_j\bar{y}_j+l_j+\lambda_j)\big)\big)$$

- *Joint lower/upper bounds* ($\underline{y}_j \le y_j \le \bar{y}_j$):

$$y_j^*(\lambda_j) = -Q_j^{-1}\big(l_j+\lambda_j-v^*(\lambda_j)\big),$$

where for every component $i\in\{1,2,\dots,N_y\}$

$$v_i^*(\lambda_j) = \begin{cases} \pi_{[-\gamma_u,0]}\big((Q_j\bar{y}_j+l_j+\lambda_j)_i\big) & \text{if } \zeta_i<0,\\ \pi_{[0,\gamma_l]}\big((Q_j\underline{y}_j+l_j+\lambda_j)_i\big) & \text{if } \zeta_i>(\bar{y}_j-\underline{y}_j)_i,\\ 0 & \text{otherwise,}\end{cases}$$

with $\zeta_i = \big(Q_j^{-1}(l_j+\lambda_j)+\bar{y}_j\big)_i$.

The scalars $\gamma>0$ and $\gamma_l,\gamma_u>0$ denote the penalty parameters, and $\pi_{[0,\gamma]}(\cdot)$ represents the projection on the set $[0,\gamma]$.

Proof. See Appendix VIII-A.

Term $d_z(\nu)$: This term is nonsmooth since the minimum is not guaranteed to be unique (cf. Danskin's Theorem). However, we can separate the minimization over the n_z outputs, i.e. for $j\in\{1,2,\dots,n_z\}$ we may consider the functions

$$d_{z_j}(\nu_j) = \min_{z_j}\ q_j(z_j)+\nu_j^T z_j, \tag{12}$$

where ν_j denotes the dual multiplier vector that corresponds to the jth output. Note that $d_z(\nu) = \sum_{j=1}^{n_z} d_{z_j}(\nu_j)$. The next proposition reveals the structure of the functions d_{z_j} in case of lower, upper and joint lower/upper bound constraints.

Proposition 3: Consider the definition of the concave function d_{z_j} in (12), using the exact penalty form in (4). For each of the following types of bound constraints, the function d_{z_j} can be written as:

- *Lower bound* ($\underline{z}_j \le z_j$):

$$d_{z_j}(\nu_j) = \begin{cases} \nu_j^T\underline{z}_j & \text{if } \nu_j\in[0,\gamma]^{N_z},\\ -\infty & \text{otherwise.}\end{cases}$$

- *Upper bound* ($z_j \le \bar{z}_j$):

$$d_{z_j}(\nu_j) = \begin{cases} \nu_j^T\bar{z}_j & \text{if } \nu_j\in[-\gamma,0]^{N_z},\\ -\infty & \text{otherwise.}\end{cases}$$

- *Joint lower/upper bounds* ($\underline{z}_j \le z_j \le \bar{z}_j$):

$$d_{z_j}(\nu_j) = \begin{cases} \nu_j^T\bar{z}_j - \max\{0,\nu_j\}^T(\bar{z}_j-\underline{z}_j) & \text{if } \nu_j\in[-\gamma_u,\gamma_l]^{N_z},\\ -\infty & \text{otherwise.}\end{cases}$$

Proof. See Appendix VIII-B.

V. Practical MPC example

We will exemplify the theoretical results obtained so far on a real-world instance of (1) that stems from a compact subsea separator MPC problem in [1]. The compact separator separates a multiphase input flow of liquid (oil/water) and gas using compact co-axial cyclones at two stages [9]. As a first-order solution method in the dual domain we will apply Nesterov's fast gradient method in the variant given in [10].

According to formulation (1), the compact separator MPC problem setup has $n_u = 3$ inputs subject to bound and rate constraints (each with $N_u = 6$) and $n_y = 2$ and $n_z = 2$ outputs (each with $N_y = N_z = 10$). The outputs y have joint lower/upper bounds, i.e.

$$\underline{y}_1 \le y_1 \le \overline{y}_1, \text{ and } \underline{y}_2 \le y_2 \le \overline{y}_2,$$

which are taken care of in the form of exact penalty terms in the objective, weighted with penalty weights $\underline{\rho}_1, \underline{\rho}_2 > 0$ for the lower bound and $\overline{\rho}_1, \overline{\rho}_2 > 0$ for the upper bound.

Based on (4), the exact penalty terms in (1) become

$$p_j(y_j) = \underline{\rho}_j \cdot \sum_{k=1}^{N_y} \max\{0, \underline{y}_{j,k} - y_{j,k}\} + \overline{\rho}_j \cdot \sum_{k=1}^{N_y} \max\{0, y_{j,k} - \overline{y}_{j,k}\},$$

for $j \in \{1,2\}$. The outputs z are subject to one-sided lower and upper bounds, i.e. $\underline{z}_1 \le z_1$, and $z_2 \le \overline{z}_2$, which lead to the exact penalty terms

$$q_1(z_1) = \underline{\eta}_1 \cdot \sum_{k=1}^{N_z} \max\{0, \underline{z}_{1,k} - z_{1,k}\},$$
$$q_2(z_2) = \overline{\eta}_2 \cdot \sum_{k=1}^{N_z} \max\{0, z_{2,k} - \overline{z}_{2,k}\},$$

with penalty weights $\underline{\eta}_1, \overline{\eta}_2 > 0$.

Following Proposition 3, the dual problem for this setup reads

$$\begin{aligned} &\max \tilde{d}(\lambda, \nu) \\ &\text{s.t. } \lambda \in \mathbb{R}^{n_y \cdot N_y} \\ &\quad \nu = (\nu_1, \nu_2) \\ &\quad \nu_1 \in [0, \underline{\eta}_1]^{N_z} \\ &\quad \nu_2 \in [-\overline{\eta}_2, 0]^{N_z}, \end{aligned} \tag{13}$$

with the concave objective function being defined as

$$\begin{aligned} \tilde{d}(\lambda, \nu) \triangleq\ & d_{\Delta u}(\lambda, \nu) + d_y(\lambda) \\ & + \nu_1^T (\underline{z}_1 - z_{f,1}) + \nu_2^T (\overline{z}_2 - z_{f,2}) - \lambda^T y_f. \end{aligned}$$

Observe that the dual objective $\tilde{d}$ is differentiable and the projection operator of the feasible set is easy to evaluate.

In order to solve (13) by means of a gradient method, we require the objective to be L-smooth, i.e. its gradient to be Lipschitz continuous. This requirement makes it possible to globally lower-bound the dual objective by a quadratic function. It is common to use a quadratic with a diagonal Hessian $-L \cdot I$. However, one can improve convergence substantially by allowing for a full Hessian matrix, which corresponds to a preconditioning of the dual problem. In fact, the best possible Hessian can be chosen according to [10, Theorem 10], so that we obtain for every pair $(\bar{\lambda}, \bar{\nu}) \in \mathbb{R}^{n_y \cdot N_y} \times \mathbb{R}^{n_z \cdot N_z}$ and for all $(\lambda, \nu) \in \mathbb{R}^{n_y \cdot N_y} \times \mathbb{R}^{n_z \cdot N_z}$

$$\tilde{d}(\lambda, \nu) \ge s_d(\lambda, \nu; \bar{\lambda}, \bar{\nu}),$$

where

$$\begin{aligned} s_d(\lambda, \nu; \bar{\lambda}, \bar{\nu}) \triangleq\ & \tilde{d}(\bar{\lambda}, \bar{\nu}) + \nabla_\lambda \tilde{d}(\bar{\lambda}, \bar{\nu})^T (\lambda - \bar{\lambda}) \\ & + \nabla_\nu \tilde{d}(\bar{\lambda}, \bar{\nu})^T (\nu - \bar{\nu}) - \frac{1}{2} \left\| \begin{bmatrix} \lambda - \bar{\lambda} \\ \nu - \bar{\nu} \end{bmatrix} \right\|_H^2, \end{aligned}$$

with $H = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$, and the respective block matrices being

$$A = \Theta_y R^{-1} \Theta_y^T + Q^{-1}, \ B = \Theta_y R^{-1} \Theta_z^T, \ C = \Theta_z R^{-1} \Theta_z^T.$$

In the fast gradient method the main effort in every iteration is to maximize the concave quadratic surrogate function s_d, defined around the current pair of 'iterates' $(\bar{\lambda}, \bar{\nu})$, over the dual feasible set, i.e. to solve

$$\begin{aligned} &\max_{\lambda, \nu} s_d(\lambda, \nu; \bar{\lambda}, \bar{\nu}) \\ &\text{s.t. } \lambda \in \mathbb{R}^{n_y \cdot N_y} \\ &\quad \nu = (\nu_1, \nu_2) \\ &\quad \nu_1 \in [0, \underline{\eta}_1]^{N_z} \\ &\quad \nu_2 \in [-\overline{\eta}_2, 0]^{N_z}. \end{aligned} \tag{14}$$

A closer look at the matrix H reveals that problem (14) cannot be solved in a convenient way (H is positive semi-definite and also dense). Therefore, we propose the following remedy: First, we maximize the original surrogate function s_d over λ, which can be done in closed form since the block matrix A is positive definite, i.e.

$$\hat{s}_d(\nu; \bar{\lambda}, \bar{\nu}) \triangleq \max_\lambda s_d(\lambda, \nu; \bar{\lambda}, \bar{\nu}).$$

This maximization evaluates to

$$\begin{aligned} \hat{s}_d(\nu; \bar{\lambda}, \bar{\nu}) = \tilde{d}(\bar{\lambda}, \bar{\nu}) &+ \frac{1}{2} \left\| \nabla_\lambda \tilde{d}(\bar{\lambda}, \bar{\nu}) \right\|_{A^{-1}}^2 \\ &- \frac{1}{2} \|\nu - \bar{\nu}\|_S^2 + w^T (\nu - \bar{\nu}), \end{aligned} \tag{15}$$

where the Schur complement of A is $S = C - B^T A^{-1} B$ and the vector w is

$$w = \nabla_\nu \tilde{d}(\bar{\lambda}, \bar{\nu}) - B^T A^{-1} \nabla_\lambda \tilde{d}(\bar{\lambda}, \bar{\nu}).$$

Considering the resulting quadratic function $\hat{s}_d$, we then perform a single gradient ascent step at $\nu = \bar{\nu}$ with line search, i.e. we solve

$$\begin{aligned} &\max_{\alpha, \nu} -\frac{1}{2} \|\nu - \bar{\nu}\|_S^2 + w^T (\nu - \bar{\nu}) \\ &\text{s.t. } \alpha \in [0, \bar{\alpha}], \ \nu = \bar{\nu} + \alpha w, \end{aligned} \tag{16}$$

which is more conveniently written as the one-dimensional problem

$$\max_{\alpha \in [0, \bar{\alpha}]} -\frac{1}{2} c_q \alpha^2 + c_l \alpha, \tag{17}$$

where $c_q = \|w\|_S^2$ and $c_l = \|w\|_2^2$. We introduce an upper bound $\bar{\alpha}$ on the step size to capture the case $c_q = 0$. A reasonable choice is given by

$$\bar{\alpha} \gg \frac{1}{\lambda_{\max}(S)},$$

which follows from the standard constant step size rule in gradient methods (see e.g. [10]).

The solution to (17) can be compactly written as

$$\alpha^* = \begin{cases} 0 & \text{if } c_q = 0 \text{ and } c_l = 0 \\ \bar{\alpha} & \text{if } c_q = 0 \text{ and } c_l > 0 \\ \min\{c_q^{-1} c_l, \bar{\alpha}\} & \text{otherwise.} \end{cases} \tag{18}$$

Algorithm 1 *Customized* dual fast gradient method for problem (13)

Require: $\lambda_0 \in \mathbb{R}^{n_y \cdot N_y}$, $\nu_0 \in \mathbb{R}^{n_z \cdot N_z}$

1: $\lambda_{-1} = \lambda_0, \nu_{-1} = \nu_0$

2: $\theta_0 = \theta_{-1} = 1$

3: **loop**

4: $\begin{bmatrix} \tau_i \\ \psi_i \end{bmatrix} = \begin{bmatrix} \lambda_i \\ \nu_i \end{bmatrix} + \theta_i(\theta_{i-1}^{-1} - 1) \begin{bmatrix} \lambda_i - \lambda_{i-1} \\ \nu_i - \nu_{i-1} \end{bmatrix}$ {auxiliary iterates}

5: compute the dual gradient at (τ_i, ψ_i):

$\nabla_\lambda \tilde{d}(\tau_i, \psi_i) = y^*(\tau_i) - \Theta_y \Delta u^*(\tau_i, \psi_i) - y_f$

$\nabla_\nu \tilde{d}(\tau_i, \psi_i) = \begin{bmatrix} z_1 \\ \bar{z}_2 \end{bmatrix} - \Theta_z \Delta u^*(\tau_i, \psi_i) - z_f,$

where $y^*(\tau_i) = \big(y_1^*(\tau_{i,1}), y_2^*(\tau_{i,2})\big)$, {using Proposition 2}

$\Delta u^*(\tau_i, \psi_i) = \big(\Delta u_j^*(\tau_i, \psi_i)\big)_{j=1}^{j=n_u}$, {cf. (10)}

6: $w = \nabla_\nu \tilde{d}(\tau_i, \psi_i) - B^T A^{-1} \nabla_\lambda \tilde{d}(\tau_i, \psi_i)$

7: $\nu_{i+1} = \pi_{[0,\underline{\eta}_1]^{N_z} \times [-\overline{\eta}_2, 0]^{N_z}}(\psi_i + \alpha^* w_i)$ {α from (18)}

8: $\lambda_{i+1} = \tau_i + A^{-1}\big(\nabla_\lambda \tilde{d}(\tau_i, \psi_i) - B(\nu_{i+1} - \psi_i)\big)$

9: $\theta_{i+1} = \frac{\theta_i}{2}\left(\sqrt{\theta_i^2 + 4} - \theta_i\right)$

10: **end loop**

11: **return** $\Delta u^*(\tau_i, \psi_i)$ {when stopping criterion is met}

Lastly, we determine a feasible, approximate pair of maximizers $(\hat{\lambda}, \hat{\nu})$ of the quadratic lower bound $s_d(\lambda, \nu; \bar{\lambda}, \bar{\nu})$ from

$$\hat{\nu} = \pi_{[0,\underline{\eta}_1]^{N_z} \times [-\overline{\eta}_2, 0]^{N_z}}(\bar{\nu} + \alpha^* w),$$
$$\hat{\lambda} = \bar{\lambda} + A^{-1}\big(\nabla_\lambda \tilde{d}(\bar{\lambda}, \bar{\nu}) - B(\hat{\nu} - \bar{\nu})\big).$$

Putting together the structure exploitation results outlined above and choosing the fast gradient method leads to our proposed solution method summarized in Algorithm 1. Note that Algorithm 1 can be implemented in a code generation framework to automate the customization process for different problem sizes and data.

A. Computational Results

This section presents computational results of a Matlab implementation of Algorithm 1 using the same benchmark data as used earlier in [1]. Different to the study in [1], we use a stopping criterion that is solely based on the input changes, i.e. we stop at iteration i whenever

$$\frac{\|\Delta u^*(\tau_i, \psi_i) - \Delta u^*\|_2}{\|\Delta u^*\|_2} \leq 0.02, \tag{19}$$

where the reference solution Δu^* is found with qpOASES [11] (option set *reliable*). The stopping criterion (19) is a better choice for a solid comparison of different methods in an MPC context since the entire decision vector, as used in [1], contains variables of different magnitudes. The use of (19) leads to considerably larger iteration counts for all methods tested compared to the previous study in [1] – despite the fact that we only require a moderate relative accuracy of 2%, whereas 0.1% was required in [1].

Table I contains the results for both Algorithm 1 and the previously benchmarked methods in [1] (without restarts), where all methods are terminated after 100'000 iterations. All methods are coldstarted, e.g. $\lambda_0 = 0, \nu_0 = 0$ in Algorithm 1, and Table I states all the main parameters of the methods in the notation used in the original publications. We use $n_{\max} = 2$ (!) warm-started iterations of the classic proximal gradient method to evaluate $\Delta u^*(\tau_i, \psi_i)$ approximately in Algorithm 1 (line 5). Two iterations for the projection operation are sufficient since performing more iterations only implies minor improvements in the iteration counts (see [7] for alternative implementations for the projection operation).

TABLE I: Performance of different first-order methods for the compact separator MPC problem. Methods with superscript $^+$ have a significantly higher per iteration cost than Algorithm 1. Methods with subscript $_\delta$ require a positive definite Hessian, i.e. they solve the perturbed problem in [1].

Method	Parameters	min / average / max iterations
Algorithm 1	$n_{\max} = 2$	2/72/188
PD method in [1]	$\eta = 10$	252/629/1176
ADMM$^+$ [12]	$\rho = 25,\ \alpha = 1.62$	157/218/286
Dual FGM$_\delta$ [13]	$L = \|AH_\delta^{-1}A^T\|$	3587/45469/100'000
Dual FGM$_\delta$ [14]	L_λ, L_μ from SDP	2/40840/100'000
GPAD$_\delta$ [2]	$L_\Psi = \|A_i H_\delta^{-1} A_i^T\|$	all 100'000
GPAD$_\delta^+$ [14]	$L_\mu = A_i H_\delta^{-1} A_i^T + 10^{-6} I$	1/17382/46596

Compared with the primal-dual (PD) method proposed in [1], Algorithm 1 shows a reduction of the iteration count by about $\times 8$ on average and about $\times 6$ for the worst case (where the number of arithmetic operations per iteration is comparable). Also, the minimum iteration count is significantly low for Algorithm 1.

VI. Hardware-in-the-loop simulation study

In this section, we present a hardware-in-the-loop benchmark study featuring ANSI C implementations of Algorithm 1, the FiOrdOs generated primal-dual first-order method from [1] (i.e. the PD method in Table I), the HPMPC interior-point solver presented in [15] (using tailored linear algebra), and the qpOASES active-set solver reported in [16] for the subsea separation MPC problem. The FiOrdOs, qpOASES and HPMPC solvers are used in the current study since they were the fastest state-of-the-art solvers in their respective categories implemented for the subsea separation process (see [1], [15], [16]).

A. PLC implementation

We deployed PLC programs that embed the C implementation of the solvers on the same ABB AC500 PLC used in [1], [15] and [16]. The PLC has an MPC603e microprocessor that runs at 400 MHz. The MPC603e has a dedicated hardware FPU and is equipped with 4MB RAM and 4MB integrated memory for user program and data respectively. The C code part of the PLC application is compiled using GNU `gcc 4.7.0` with the same settings for all solvers. Specifically, `-mcpu=603e` and `-O1`, which is the maximum optimization level that could be used on the PLC.

The C code of Algorithm 1 is a simple custom implementation, which relies on offline static data preparations according to the structure exploitation results in Section IV. In particular, we pre-compute and store the matrices $\Theta_y, \Theta_z, B, A^{-1}$

TABLE II: *Cold-start* HIL test results for 600 time steps of the subsea compact separation process.

QP Solver (cold-start)	**Time** (ms) average/max	**Iterations** average/max	**Mean Square Error** $P_1/P_2/Liq_{out}/Gas_{out}$
Algorithm 1	1.5/2.1	4/4	0.01/0.002/3.91/0.26
FiOrdOs [1]	111.9/113.6	785/785	0.01/0.002/4.07/0.22
qpOASES [16]	10.4/15.9	19/26	0.01/0.002/4.15/0.23
HPMPC [15]	15.4/18.2	9/11	0.01/0.002/4.08/0.26

TABLE III: *Warm-start* HIL test results for 600 time steps of the subsea compact separation process.

QP Solver (warm-start)	**Time** (ms) average/max	**Iterations** average/max	**Mean Square Error** $P_1/P_2/Liq_{out}/Gas_{out}$
Algorithm 1	1.1/1.4	2/2	0.01/0.002/3.88/0.28
FiOrdOs [1]	14.6/16.4	100/100	0.01/0.002/3.87/0.28
qpOASES [16]	1.4/9.4	1/19	0.01/0.002/3.97/0.28

and B^TA^{-1}. Note that our implementation currently uses the `sqrt` and `pow` functions from the C standard math library. Compared to earlier studies in [1] and [16], we use the efficient recursive implementation of free-response prediction (y_f, z_f in (1)), as reported in [17], for all the embedded solver implementations tested in this paper.

B. Test setup and objectives

The test setup consists of the embedded MPC running on the PLC in closed-loop with a process simulator developed for the subsea compact separator process. Communication between the PLC and the simulator is achieved using Ethernet and an OPC server. The main objective of the compact separator MPC is to control the quality (i.e. gas volume fraction) of separated liquid and gas in their respective outlets Liq_{out} and Gas_{out}, while regulating the pressure (respectively, P_1 and P_2) at the two stages of the separation process. We use the same hydrodynamic slugging (disturbance) flow sequence in [1] in our tests, and a sampling frequency of at least 1 Hz is desired. Since control performance is essential in industrial applications, the closed-loop control performance for each embedded solver is used as the basis for result comparison.

C. Results

The hardware-in-the-loop (HIL) test results are shown in Tables II and III, where double precision floating point computations are used for all solvers. The Mean Square Error values are used as the control performance measure, and it should be noted that similar control performance is achieved for all embedded solvers.

The cold-start results in Table II show that the embedded implementation of Algorithm 1 is more than 50 times faster than the FiOrdOs implementation, more than 8 times faster than the HPMPC solver, and about 7 times faster than qpOASES. Since the HPMPC interior-point solver does not support warm starting, it is omitted from the warm-start tests. The average computational speed of Algorithm 1 for the warm-start case is comparable to that of qpOASES. However, Algorithm 1 is about 7 times faster than qpOASES in the worst case and at least 11 times faster than the FiOrdOs implementation. It is clear from the results that the proposed structure exploitation techniques lead to fast convergence for the simple dual first-order method in Algorithm 1.

TABLE IV: Memory usage on the ABB AC500 PLC.

Memory (MB)	**Algorithm 1**	**FiOrdOs**	**qpOASES**	**HPMPC**
Data size	0.04	0.04	0.16	0.10
C code size	0.05	0.08	0.16	0.20
PLC program size	0.18	0.22	0.45	0.45

Moreover, the memory footprint of Algorithm 1 reflects the memory requirement of first-order methods, as shown in Table IV (cf. columns 2 and 3). The code size of our implementation of Algorithm 1 is the smallest, and it occupies 60% less memory compared with the second-order solvers. It is therefore obvious from the results that Algorithm 1 provides the best solver implementation for the compact separator.

Note that, compared with earlier results in [1] and [16], much faster computations and smaller memory usage are obtained for all implementations due to the recursive implementation used for the free-response predictions [17].

VII. Conclusions

This paper presents techniques to exploit structure in an appropriate exact penalty formulation for output box constraints in the context of first-order methods. Together with a new, iterative projection method for input bound and rate constraints, these techniques are shown to lead to a first-order method that outperforms all of the state-of-the-art first- and second-order solution methods with respect to computational speed and memory usage when applied to a challenging industrial MPC problem.

VIII. Appendix

A. Proof of Proposition 2

We apply the saddle point theorem in [18, Proposition 5.5.7], and in the case of the lower bound on y_j, the minimization problem in (11) can be written as

$$\begin{aligned}&\min_{y_j}\left(\frac{1}{2}y_j^TQ_jy_j+(l_j+\lambda_j)^Ty_j+\max_{\mu\in[0,\gamma]^{N_y}}\mu^T(\underline{y}_j-y_j)\right)\\&=\max_{\mu\in[0,\gamma]^{N_y}}\left(\mu^T\underline{y}_j+\min_{y_j}\left(\frac{1}{2}y_j^TQ_jy_j+(l_j+\lambda_j-\mu)^Ty_j\right)\right)\end{aligned}\tag{20}$$

where the feasible set of the maximization problem is the box $[0,\gamma]^{N_y}$. The inner minimization problem solves to $y_j^*=-Q_j^{-1}(l_j+\lambda_j-\mu)$ with the minimum value given as $-\frac{1}{2}\|l_j+\lambda_j-\mu\|^2_{Q_j^{-1}}$. We then rewrite the maximization as a minimization and solve for the unique minimizer $\mu^*(\lambda_j)$,

$$\mu^*(\lambda_j)=\arg\min_{\mu\in[0,\gamma]^{N_y}}\left(\frac{1}{2}\|l_j+\lambda_j-\mu\|^2_{Q_j^{-1}}-\mu^T\underline{y}_j\right).$$

The result is obtained by noting that the computation of the minimizer can be rephrased as a projection on the box $[0,\gamma]^{N_y}$ since Q_j^{-1} is diagonal. The result for the upper bound is proved equivalently.

For the case of joint lower/upper bounds, we obtain by similar reasoning the maximization problem (cf. (20))

$$\max_{\substack{\mu_l \in [0,\gamma_l]^{N_y},\\ \mu_u \in [-\gamma_u,0]^{N_y}}} -\frac{1}{2}\|l_j+\lambda_j+\mu_u-\mu_l\|^2_{Q_j^{-1}}+\mu_l^T\underline{y}_j-\mu_u^T\bar{y}_j \quad (21)$$

with the inner minimizer $y_j^* = -Q_j^{-1}(l_j+\lambda_j+\mu_u-\mu_l)$. Note that the form of the inner minimizer allows us to compute $y_j^*(\lambda_j)$ by computing the difference $\mu_u^*-\mu_l^*$ of the optimizers of (21) only. Therefore, we introduce the definition $-\nu = \mu_u-\mu_l$, rewrite (21) accordingly, and solve for the unique minimizer $\nu^*(\lambda_j)$,

$$\begin{aligned}\nu^*(\lambda_j) = \arg\min_{\nu\in[-\gamma_u,\gamma_l]^{N_y}} \Big(&\frac{1}{2}\|l_j+\lambda_j-\nu\|^2_{Q_j^{-1}}-\nu^T\bar{y}_j\\ &+\min_{\mu_l\in\mathbb{M}}\mu_l^T(\bar{y}_j-\underline{y}_j)\Big),\end{aligned} \quad (22)$$

where for $i\in\{1,2,\ldots,N_y\}$, the feasible set

$$\mathbb{M} = \left\{\mu\in\mathbb{R}^{N_y} \mid \max\{0,\nu_i\}\le\mu_i\le\min\{\gamma_l,\nu_i+\gamma_u\}\right\}.$$

Since $\bar{y}_j \ge \underline{y}_j$ (component-wise), the inner minimization problem in (22) has the solution $\max\{0,\nu\}^T(\bar{y}_j-\underline{y}_j)$, where the max acts component-wise and is vector-valued. Inserting this expression into (22) and working out the three cases reveals the result in Proposition 2.

B. Proof of Proposition 3

In case of the lower bound on z_j, specifying the penalty function q_j leads to

$$d_{z_j}(\nu_j) = \min_{z_j}\left(\nu_j^T z_j + \max_{\mu\in[0,\gamma]^{N_z}}\mu^T(\underline{z}_j-z_j)\right). \quad (23)$$

Using $\Delta z_j \triangleq \underline{z}_j - z_j$, (23) can be rewritten as

$$d_{z_j}(\nu_j) = \nu_j^T\underline{z}_j - \max_{\Delta z_j}\left(\nu_j^T\Delta z_j - \sigma_{[0,\gamma]^{N_z}}(\Delta z_j)\right), \quad (24)$$

The optimal value of the max in (24) is the conjugate of the support function $\sigma_{[0,\gamma]^{N_z}}(\Delta z_j)$. Since the convex set $[0,\gamma]^{N_z}$ is closed, we obtain $\max_{\Delta z_j}\left(\nu_j^T\Delta z_j-\sigma_{[0,\gamma]^{N_z}}(\Delta z_j)\right) = \iota_{[0,\gamma]^{N_z}}(\nu_j)$, where $\iota_{[0,\gamma]^{N_z}}(\nu_j)$ is the indicator function of the box $[0,\gamma]^{N_z}$. This proves the result for the lower bound. The proof for the upper bound follows similar reasoning.

In the case of the joint lower/upper bounds, specifying the penalty function yields

$$d_{z_j}(\nu_j) = \min_{z_j}\left(\nu_j^T z_j + \max_{\substack{\mu_l\in[0,\gamma_l]^{N_z},\\ \mu_u\in[-\gamma_u,0]^{N_z}}}\mu_l^T(\underline{z}_j-z_j)+\mu_u^T(z_j-\bar{z}_j)\right).$$

Using [19, Corollary 37.3.2], we can swap the min and the max and rewrite d_{z_j} as

$$d_{z_j}(\nu_j) = \max_{\substack{\mu_l\in[0,\gamma_l]^{N_z},\\ \mu_u\in[-\gamma_u,0]^{N_z}}}\left(\mu_l^T\underline{z}_j-\mu_u^T\bar{z}_j+\min_{z_j}(\mu_u-\mu_l+\nu_j)^T z_j\right).$$

The inner minimization problem evaluates to

$$\min_{z_j}(\mu_u-\mu_l+\nu_j)^T z_j = \begin{cases}0 & \text{if } \mu_l-\mu_u=\nu_j,\\ -\infty & \text{otherwise,}\end{cases}$$

which implies the domain $[-\gamma_u,\gamma_l]^{N_z}$ of the function d_{z_j}. Therefore, for any ν_j in the domain, we obtain

$$d_{z_j}(\nu_j) = \max_{\substack{\mu_l\in[0,\gamma_l]^{N_z},\\ \mu_u\in[-\gamma_u,0]^{N_z},\\ \mu_l-\mu_u=\nu_j}}\mu_l^T\underline{z}_j-\mu_u^T\bar{z}_j = \nu_j^T\bar{z}_j-\max\{0,\nu_j\}^T(\bar{z}_j-\underline{z}_j),$$

which completes the proof.

References

[1] D. K. M. Kufoalor, S. Richter, L. Imsland, T. A. Johansen, M. Morari, and G. O. Eikrem, "Embedded Model Predictive Control on a PLC Using a Primal-Dual First-Order Method for a Subsea Separation Process," in *MED 2014*, Palermo, Italy, June 2014, pp. 368–373.

[2] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 18–33, 2014.

[3] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded Online Optimization for Model Predictive Control at Megahertz Rates," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3238–3251, Dec. 2014.

[4] S. Richter, "Computational complexity certification of gradient methods for real-time model predictive control," Ph.D. dissertation, ETH Zürich. Diss. Nr. 20718, 2012.

[5] D. K. M. Kufoalor, "High-performance Industrial Embedded Model Predictive Control," Ph.D. dissertation, Norwegian University of Science and Technology. Diss. Nr. 2016:157, 2016.

[6] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[7] S. Richter, "Structure Exploitation of Practical MPC Formulations for Fast and Efficient First-Order Methods," Richter Optimization GmbH, Zurich, Switzerland, Tech. Rep., 2016, available at http://www.richteroptimization.com/public/techreport_structure_exploit.pdf.

[8] J. M. Maciejowski, *Predictive Control With Constraints*. Pearson and Prentice Hall, 2002.

[9] J. Høydal, O. Kristiansen, G. O. Eikrem, and K. Fjalestad, "Method and system for fluid separation with an integrated control system," Patent WO2 013 091 719 A1, 06 27, 2013. [Online]. Available: http://www.google.com/patents/WO2013091719A1?cl=en&hl=no

[10] P. Giselsson, "Improved Fast Dual Gradient Methods for Embedded Model Predictive Control," in *IFAC World Congress*, vol. 19, 2014, pp. 2303–2309.

[11] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[12] B. O'Donoghue, G. Stathopoulos, and S. Boyd, "A Splitting Method for Optimal Control," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2432–2442, Nov. 2013.

[13] S. Richter, C. N. Jones, and M. Morari, "Certification Aspects of the Fast Gradient Method for Solving the Dual of Parametric Convex Programs," *Mathematical Methods of Operations Research*, vol. 77, no. 3, pp. 305–321, Jan. 2013.

[14] P. Giselsson and S. Boyd, "Metric selection in fast dual forward-backward splitting," *Automatica*, vol. 62, pp. 1–10, Dec. 2015.

[15] D. Kufoalor, G. Frison, L. Imsland, T. Johansen, and J. Jørgensen, "Block factorization of step response model predictive control problems," *Journal of Process Control*, vol. 53, pp. 1 – 14, 2017.

[16] D. K. M. Kufoalor, B. J. T. Binder, H. J. Ferreau, L. Imsland, T. A. Johansen, and M. Diehl, "Automatic deployment of industrial embedded model predictive control using qpOASES," in *2015 European Control Conference (ECC)*, July 2015, pp. 2601–2608.

[17] D. K. M. Kufoalor, L. Imsland, and T. A. Johansen, "Efficient implementation of step response models for embedded Model Predictive Control," *Computers & Chemical Engineering*, vol. 90, pp. 121–135, 2016.

[18] D. P. Bertsekas, *Convex Optimization Theory*, 1st ed. Athena Scientific, 2009.

[19] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 1997.