Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: https://www.tandfonline.com/loi/goms20

Proportioning with second-order information for model predictive control

Ondřej Šantin, Marta Jarošová, Vladimír Havlena & Zdeněk Dostál

To cite this article: Ondřej Šantin, Marta Jarošová, Vladimír Havlena & Zdeněk Dostál (2017) Proportioning with second-order information for model predictive control, Optimization Methods and Software, 32:3, 436-454, DOI: 10.1080/10556788.2016.1213840

To link to this article: https://doi.org/10.1080/10556788.2016.1213840

Optimization Methods & Software, 2017
Vol. 32, No. 3, 436–454, http://dx.doi.org/10.1080/10556788.2016.1213840

Taylor & Francis
Taylor & Francis Group

Proportioning with second-order information for model predictive control

Ondřej Šantin[a*], Marta Jarošová[b], Vladimír Havlena[c] and Zdeněk Dostál[b,d]

[a]*Honeywell Automotive Software, V Parku 2326/18, CZ-14800 Prague, Czech Republic;* [b]*IT4Innovations, VŠB-Technical University of Ostrava, Tř 17 listopadu, CZ-70833 Ostrava, Czech Republic;* [c]*Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, CZ-16027 Prague 6, Czech Republic;* [d]*Faculty of Electrical Engineering and Computer Science, VŠB-Technical University of Ostrava, Tř 17 listopadu, CZ-70833 Ostrava, Czech Republic*

(*Received 21 May 2015; accepted 11 July 2016*)

We propose an algorithm for the effective solution of quadratic programming (QP) problems arising from model predictive control (MPC). MPC is a modern multivariable control method which gives the solution for a QP problem at each sample instant. Our algorithm combines the active-set strategy with the proportioning test to decide when to leave the actual active set. For the minimization in the face, we use a direct solver implemented by the Cholesky factors updates. The performance of the algorithm is illustrated by numerical experiments, and the results are compared with the state-of-the-art solvers on benchmarks from MPC.

Keywords: quadratic programming; model predictive control; active-set strategy

AMS Subject Classification: 90C20; 93C95; 49N90

1. Introduction

We shall be concerned with the sequence of problems to find

$$f^*(\boldsymbol{x}) \triangleq \min_{z\in\Omega} f(z,\boldsymbol{x}) = f(z^*(\boldsymbol{x}),\boldsymbol{x}), \tag{1}$$

where $\Omega = \{z : \underline{z} \le z \le \bar{z}\}$, $f(z,\boldsymbol{x}) = \frac{1}{2}z^T\mathbf{H}z + \boldsymbol{h}^T(\boldsymbol{x})z$, $\underline{z}$ and $\bar{z}$ are given column n-vectors, $\mathbf{H}$ is an $n \times n$ symmetric positive-definite (SPD) matrix, and $\boldsymbol{h} = \boldsymbol{h}(\boldsymbol{x})$ is an n-vector, parameterized by a parameter column n_x-vector $\boldsymbol{x} \in \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots\}$. For a fixed parameter $\boldsymbol{x}$, we define

$$q(z) = \tfrac{1}{2}\, z^T\mathbf{H}z + \boldsymbol{h}^T z. \tag{2}$$

We suppose that n is relatively small, i.e. $n \lesssim 200$. Moreover, we assume that the sequence $\{\boldsymbol{x}_k\}$ is not known in advance and $\boldsymbol{x}_k \approx \boldsymbol{x}_{k+1}$, so that we can use $z^*(\boldsymbol{x}_k)$ as a good initial approximation for $z^*(\boldsymbol{x}_{k+1})$. It is shown later that such sequence of parameters is generated in model predictive control (MPC) applications.

*Corresponding author. Email: ondrej.santin@honeywell.com

Problems of type (1) arise in many applications, e.g. in moving horizon estimation [37] or in MPC [17]—our main motivation. MPC is a multi-variable model-based control strategy, which can naturally handle input, state, and output constraints [28]. It is widely used in industrial [36] or automotive applications, see, e.g. [15,23,26].

MPC solves at each sampling instant a QP problem, where the optimization problem remains the same, but the value of the estimated state $\boldsymbol{x}$ changes. We are often limited by available memory and computation power, e.g. available RAM memory is less than 200 KB and CPU is 200 MHz micro-controller in standard engine control unit (ECU). Moreover, not all resources may be available to the MPC solver, since many other systems are running within the ECU. On the other hand, the number of variables in typical embedded applications of MPC ranges from tens to hundreds. In the last two decades, there has been a rapid development in optimization algorithms enabling sampling times in the order of micro or milliseconds.

It has been shown recently that much of the computational effort for solving (1) can be done off-line. In [2] the authors consider problem (1) as a multi-parametric QP problem and exploit its parametric nature to obtain explicit solutions to the MPC problem. The online phase of the MPC controller is then reduced to the look-up table process and the solution is obtained using an affine map defined by the parameter value. The main drawback of the Explicit MPC is a large increase in memory requirements with an increasing number of constraints. In spite of some improvements, see, e.g. [3,5,21,24], the Explicit MPC is limited to models with small state dimensions and few process inputs. This bottleneck motivates the development of online methods. Most of them reduce the solution of (1) to a series of unconstrained problems, which are solved either by iterative or direct methods. They typically combine the modified Newton method, the active-set method, and the gradient method.

The methods include the fast gradient methods (FGM) [30], which attracted the attention of the MPC community by the cheap iterations and the rate of convergence, which depends on the square root of the condition number [39] of $\mathbf{H}$. Moreover, it was shown that the upper bound on the number of iterations can be determined off-line. The tight upper bound (2-4 times larger than the real iterations observed) has been reported for the problems with simple bounds in [41]. The estimate of the upper bound has also been developed in [19,34,40].

The interior-point methods (IPM) typically require a relatively small number of computationally expensive Newton iterations for the auxiliary penalized problems, see, e.g. [7,31,48]. It was shown in [38] that the equality constrained problems in primal–dual IPM iterations can be solved at the cost proportional to the length of the prediction horizon when the Riccati recursion is used for the sparse formulation of MPC. The same computational complexity was achieved in [48] with the block-wise Cholesky factorization. Moreover, in the paper the warm start with the fixed barrier parameter and fixed number of iterations was used. This reduced the computational cost but resulted in a suboptimal solution, which can even violate the system dynamics equation. The FORCES package [10] produces an automatically generated code in which the Newton-step computation [48] is enhanced by two substitutions using the triangular factorization with possible speed-up for special instances. Although the automatically generated problem-tailored code may result in superior solution time, the code size might increase rapidly with the problem size, thus preventing an embedded application.

In the active-set methods (ASM), the optimal active set is estimated by the working set [31]. The well-known drawback of ASM is that if the initial working set is far from optimal, then the algorithm might require many iterations to reach the solution. This has been partly removed in the open source package qpOASES [14], which implements the active-set strategy combined with the piece-wise affine structure of the explicit MPC solution.

A combination of the Newton method and the gradient projection (GP) algorithm [31, Chapter 16.7] was applied to a dual sparse MPC formulation [1], and the linear complexity of each iteration with respect to the prediction horizon using the Riccati recursion was established. The algorithm [45], based on [4], combines the GP algorithm with projection of the Newton step. The

Newton step is defined by the current active set, and computed by the Cholesky factorization and the null-space method [31, Chapter 16.2]. The null-space method was replaced by the conjugate gradient method in [44] for large-scale MPC. A favourable distribution of the eigenvalues of **H** arising in the modified MPC problem was exploited in [43], to speed-up the solution of auxiliary linear problems by the conjugate gradients combined with the gradient/Newton projection algorithm. The combined Gradient/Newton Projection (CGNP) of Otta *et al.* [32], applied to the sparse approximation of the MPC problem, explores first the Newton direction up to the first bound and then proceeds by using either the projection of the Newton direction by projected line search (PLS) [31, Chapter 16.7] or the GP with the fixed step-size.

The same approach applied to the dense MPC formulation was patented [35] in the domain of automated process control. This was later developed as part of a commercially available software product for the diesel engine's air-path control using MPC, see [23]. The authors report the use of the software tool in several embedded applications using either directly ECU or rapid prototyping system: the air-path control of dual loop re-circulation diesel engine [26], the temperature control of the diesel oxidation catalyst [27], and the thermal management of combustion engine [25]. The CGNP method [32] has been extended to nonlinear MPC [46] using the PLS with the GP. The resulting method has been applied to the cruise control problem running on the standard production powertrain control module.

The effective precision control of the solution of auxiliary linear problems is exploited by the modified proportioning with reduced gradient projections (MPRGP) algorithm [13]. This algorithm explores the face defined by the current active set by the conjugate gradients until the component of the gradient which corresponds to the active set dominates the violation of the Karush–Kuhn–Tucker (KKT) conditions, or an infeasible iteration is generated. In the first case, the active set is expanded by the GP with a fixed step length, otherwise it is reduced by the so-called proportioning step. The algorithm has been proved to enjoy a global R-linear rate of convergence and was successfully applied to the solution of large problems discretized by billions of variables.

In this paper, we propose a new algorithm for an effective solution of QP problems arising from MPC, which tries to exploit the advantages of the algorithms MPRGP and CGNP. The algorithm exploits the control of MPRGP to 'look ahead' in order to avoid unnecessary expansion of the active set in combination with full exploitation of the second-order information to solve the face problem exactly by the direct solver.

The rest of the paper is organized as follows. In Section 2, the MPC problem is introduced. Section 3 gives an overview of MPC problem formulation. Section 4 presents the basic ingredients used later in Section 5, where the proposed algorithm is described. Convergence analysis of the proposed algorithm is given in Section 6. Effectiveness of the proposed method is confirmed in Section 7. The last section presents the conclusions.

2. Model predictive control

Let us consider a linear regulator MPC problem

$$f^*_{\mathrm{MPC}}(\boldsymbol{x}) \triangleq \min_{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{N-1}} \quad \frac{1}{2}\boldsymbol{x}_N^T\mathbf{P}\boldsymbol{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}(\boldsymbol{x}_k^T\mathbf{Q}\boldsymbol{x}_k + \boldsymbol{u}_k^T\mathbf{R}\boldsymbol{u}_k), \tag{3a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \mathbf{A}\boldsymbol{x}_k + \mathbf{B}\boldsymbol{u}_k, \quad k = 0,\ldots,N-1,$$

$$\boldsymbol{u}_k \in \tilde{\Omega}, \quad k = 0,\ldots,N-1,$$

$$\boldsymbol{x}_0 = \boldsymbol{x}, \tag{3b}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is a current state estimate, $N \in \mathbb{N}$ is the finite prediction horizon, and $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ denote the state and control input at time-step k, respectively. Matrix $\mathbf{Q} \in \mathbb{R}^{n_x \times n_x}$ is symmetric positive semi-definite (SPS), matrices $\mathbf{R} \in \mathbb{R}^{n_u \times n_u}$ and $\mathbf{P} \in \mathbb{R}^{n_x \times n_x}$ are SPD.

The control input is assumed to belong to the convex compact set $\tilde{\Omega} = \{\boldsymbol{u} : \underline{\boldsymbol{u}} \leq \boldsymbol{u} \leq \bar{\boldsymbol{u}}\}$ which contains the origin in its interior.

Remark 1 As there is a new $\boldsymbol{x}$ at each sampling instant, the problem (3) has to be recomputed. Hence, the so-called receding horizon control concept is established, i.e. the plan of control inputs $\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}$ is recomputed at each sampling instant with estimated/measured system state $\boldsymbol{x}$ as the parameter and only the first control move $\boldsymbol{u}_0$ is actually applied to the system, cf. [29]. The need for re-computation requires a fast solver for problem (3), in order to have a solution ready by the next sampling time.

Remark 2 The practical need for constraints on system states and outputs can be translated to

$$\mathbf{G}_c \boldsymbol{u}_k \leq \mathbf{S}_c \boldsymbol{x} + \boldsymbol{w}_c, \quad k = 0, \ldots, N-1.$$

These constraints are treated as *soft* constraints, i.e. those which can be violated, but any violation is penalized in the objective function and simple constrained slack variables are added as decision variables, see, e.g. [22,29].

3. Problem formulation

There are several possible approaches to recast the MPC problem (3) into an optimization problem. In *sparse formulation*, the future states $\boldsymbol{x}_k$ are added as the decision variables and the dynamic constraints (3b) are incorporated into the problem as equality constraints. The resulting optimization problem then has $N(n_x + n_u)$ decision variables, but the problem matrices are sparse. This approach is usually used in combination with IPM, as it can lead to significant speed-ups, mainly for larger prediction horizons. The problem structure can be exploited to have linear complexity at each iteration with respect to N, see, e.g., [48].

To eliminate the equality constraints and preserve the linear complexity of each iteration, the dual optimization of the sparse problem was formulated and solved by the GP algorithm in [1]. The main drawback of the dual formulation is the double number of the decision variables as compared to the primal one.

In *dense formulation*, the state variables are eliminated using predictions. Even if the sparse structure is lost, the number of decision variables is reduced to $N \cdot n_u$, hence this formulation is favourable for problems with a relatively short prediction horizon ($N < 50$ samples) or for problems with a large number of states. Moreover, dense formulation enables to use the move blocking strategies [8] to further reduce the number of optimization variables.

The state prediction can be expressed as a function of the current state $\boldsymbol{x}$ and the control inputs $\mathbf{U}$ [28] by the recursive use of (3b) as

$$\mathsf{X} = \mathsf{A}\boldsymbol{x} + \mathsf{B}\mathsf{U},$$

where

$$\mathsf{X} = [\boldsymbol{x}_1^{\mathrm{T}}\, \boldsymbol{x}_2^{\mathrm{T}} \ldots \boldsymbol{x}_N^{\mathrm{T}}]^{\mathrm{T}}, \quad \mathsf{U} = [\boldsymbol{u}_0^{\mathrm{T}}\, \boldsymbol{u}_1^{\mathrm{T}} \ldots \boldsymbol{u}_{N-1}^{\mathrm{T}}]^{\mathrm{T}}$$

and

$$\mathsf{A} = \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^{N-1} \\ \mathbf{A}^N \end{bmatrix}, \quad \mathsf{B} = \begin{bmatrix} \mathbf{B} & 0 & & & \\ \mathbf{AB} & \mathbf{B} & \ddots & & \\ \vdots & & \ddots & & \\ \mathbf{A}^{N-2}\boldsymbol{B} & & & \mathbf{B} & 0 \\ \mathbf{A}^{N-1}\boldsymbol{B} & \mathbf{A}^{N-2}\mathbf{B} & \dots & \mathbf{AB} & \mathbf{B} \end{bmatrix}.$$

Problem (3) can be rewritten in the form (1) denoting $\boldsymbol{z} = \mathsf{U} \in \mathbb{R}^n$, $n = N \cdot n_u$,

$$\mathbf{H} = \mathsf{B}^{\mathrm{T}}\mathsf{Q}\mathsf{B} + \mathsf{R}, \quad \boldsymbol{h}(\boldsymbol{x}) = (\mathsf{B}^{\mathrm{T}}\mathsf{Q}\mathsf{A})\boldsymbol{x}, \quad \underline{\boldsymbol{z}} = \mathbf{1}_N \otimes \underline{\boldsymbol{u}},$$
$$\mathsf{Q} = \operatorname{diag}(\mathbf{I}_N \otimes \mathbf{Q}, \mathbf{P}), \quad \mathsf{R} = \mathbf{I}_N \otimes \mathbf{R}, \quad \bar{\boldsymbol{z}} = \mathbf{1}_N \otimes \bar{\boldsymbol{u}},$$

where $\otimes$ refers to the Kronecker product, $\mathbf{I}_N$ and $\mathbf{1}_N$ are the identity matrix and the column vector of ones of the dimension N, respectively. The constant term was omitted since it does not influence the minimizer.

4. Basic ingredients

Let us first review the notations and present the basic ingredients of the proposed algorithm.

For any SPD matrix $\mathbf{M}$, the Euclidean norm and the $\mathbf{M}$-energy norm of $\boldsymbol{y}$ will be denoted by $\|\boldsymbol{y}\|$ and $\|\boldsymbol{y}\|_{\mathbf{M}}$, respectively. Hence $\|\boldsymbol{y}\|^2 = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}$ and $\|\boldsymbol{y}\|_{\mathbf{M}}^2 = \boldsymbol{y}^{\mathrm{T}}\mathbf{M}\boldsymbol{y}$. Analogous notation will be used for the induced norm, hence the spectral condition number $\kappa(\mathbf{M})$ of matrix $\mathbf{M}$ is defined by $\kappa(\mathbf{M}) = \|\mathbf{M}\|\|\mathbf{M}^{-1}\|$.

The projection $\mathcal{P}_\Omega$ to $\Omega = \{\boldsymbol{z} : \underline{\boldsymbol{z}} \le \boldsymbol{z} \le \bar{\boldsymbol{z}}\}$ is defined for any n-vector $\boldsymbol{z}$ by

$$\mathcal{P}_\Omega(\boldsymbol{z}) = \boldsymbol{z}^+,$$

where the entries of $\boldsymbol{z}^+$ are defined as $z_i^+ = \max\{\underline{z}_i, \min\{\bar{z}_i, z_i\}\}$.

For any matrix $\mathbf{N}$, $\mathbf{N}_i$ denotes its ith row. Given the matrix $\mathbf{N} \in \mathbb{R}^{m\times n}$ and any set $\mathcal{R} \subseteq \{1, \dots, m\}$, $\mathcal{C} \subseteq \{1, \dots, n\}$, $\mathbf{N}_{\mathcal{R},\mathcal{C}}$ denotes the submatrix of $\mathbf{N}$, comprising the rows indexed by $\mathcal{R}$, and columns indexed by $\mathcal{C}$.

4.1 *Quantitative refinement of the KKT conditions*

For an arbitrary n-vector $\boldsymbol{z}$, let us define the gradient $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{z})$ of q in (2) by

$$\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{z}) = \mathbf{H}\boldsymbol{z} + \boldsymbol{h}.$$

The optimal solution $\boldsymbol{z}^*$ of (1) is fully determined by the KKT optimality conditions[1] [7] which require that for $i = 1, \dots, n$,

$$z_i^* = \begin{cases} \bar{z}_i & \text{implies } g_i^* \le 0, \\ \underline{z}_i & \text{implies } g_i^* \ge 0, \\ \underline{z}_i \le z_i^* \le \bar{z}_i & \text{implies } g_i^* = 0. \end{cases} \tag{4}$$

Denoting $\mathcal{I} = \{1, 2, \dots, n\}$, let us define the upper, lower, and active set of $\boldsymbol{z}$ as $\mathcal{U} = \{i \in \mathcal{I} : z_i = \bar{z}_i\}$, $\mathcal{L} = \{i \in \mathcal{I} : z_i = \underline{z}_i\}$, and $\mathcal{A} = \mathcal{U} \cup \mathcal{L}$, respectively. The complement of the active set is called the *free set* $\mathcal{F} = \{i \in \mathcal{I} : i \notin \mathcal{A}\}$.

To give an alternative reference to the KKT conditions (4), let us define the *free gradient* $\boldsymbol{\varphi}$ and the *chopped gradient* $\boldsymbol{\beta}$ by

$$\begin{aligned} \varphi_i(z) &= g_i \quad \text{for } i \in \mathcal{F}, \quad \varphi_i(z) = 0 \quad \text{for } i \in \mathcal{A}, \\ \beta_i(z) &= 0 \quad \text{for } i \in \mathcal{F}, \quad \beta_i(z) = g_i^{\#}(z) \quad \text{for } i \in \mathcal{A}, \end{aligned} \tag{5}$$

where

$$g_i^{\#} = \begin{cases} \max\{g_i, 0\} & \text{if } i \in \mathcal{U}, \\ \min\{g_i, 0\} & \text{if } i \in \mathcal{L}. \end{cases}$$

Thus the KKT conditions are satisfied if and only if the *projected gradient* $\boldsymbol{v} = \boldsymbol{\varphi} + \boldsymbol{\beta}$ is equal to zero. It can be checked easily that

$$\|z - z^*\|_{\mathbf{H}}^2 \le 2(q(z) - q^*(z^*)) \le \lambda_{\min}^{-1}(\mathbf{H})\|\boldsymbol{v}\|^2,$$

where $\lambda_{\min}(\mathbf{H})$ denotes the smallest eigenvalue of $\mathbf{H}$ (see, e.g. [12, Section 5.2]).

4.2 *Face problem solution*

We shall use the active-set strategy which reduces the solution of (1) to the solution of a sequence of equality constrained problems. Thus in each iteration, we need to solve the auxiliary minimization problem

$$\tilde{z} = \arg\min_{z \in \Phi} q(z), \tag{6}$$

where

$$\Phi = \{z : z_i = \bar{z}_i \text{ for } i \in \mathcal{U} \text{ or } z_i = \underline{z}_i \text{ for } i \in \mathcal{L}\}$$

is the *face* defined by the lower and upper active sets $\mathcal{L}, \mathcal{U}$ of the indices which are predicted to be active in the solution. Since $\tilde{z}_{\mathcal{U}} = \bar{z}_{\mathcal{U}}$ and $\tilde{z}_{\mathcal{L}} = \underline{z}_{\mathcal{L}}$, the equality constrained problem (6) is equivalent to the unconstrained minimization problem defined only for the variables $z_{\mathcal{I}\setminus\mathcal{A}}$.

Problem (6) can be solved by the Krylov space methods, which are advantageous in large-scale optimization. Since a low precision of the solution of auxiliary problems is often sufficient, the performance of the Krylov space methods can often be improved by an efficient adaptive precision control [12]. On the other hand, if the size of the problem is relatively small, as we consider in this paper, we can find the exact solution more efficiently by a direct solver. Since only simple constraints are present in (1), assuming $z \in \Phi$, the minimizer of the face problem (6) can be written as

$$\tilde{z} = z - \boldsymbol{p},$$

where

$$\boldsymbol{p} = \begin{cases} \boldsymbol{p}_{\mathcal{I}\setminus\mathcal{A}} & = \mathbf{G}^{-1}\boldsymbol{r}, \\ \boldsymbol{p}_{\mathcal{A}} & = \mathbf{0}, \end{cases} \tag{7}$$

and the reduced Hessian and the reduced gradient are defined as

$$\mathbf{G} = \mathbf{H}_{\mathcal{I}\setminus\mathcal{A},\mathcal{I}\setminus\mathcal{A}}, \quad \boldsymbol{r} = \boldsymbol{g}(z)_{\mathcal{I}\setminus\mathcal{A}}. \tag{8}$$

The proposed method moves towards the minimizer of (6). If a new constraint is activated along the way, the active set is expanded via projected line search (PLS) of [31] as described in Section 4.6.

4.3 *Cholesky factors updates and proportioning*

Problem (7) can be solved, e.g. by the Cholesky factorization in $\mathcal{O}((n-m)^3)$ flops, followed by two backward substitutions in $2(n-m)^2$ flops, where $m = |\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$. The cost of factorization can be reduced to $\mathcal{O}(n^2)$ by the application of the factor update scheme [18]. Hence the complexity of each iteration is relatively low, but it is still important to keep the number of iterations small.

The algorithm that we propose here tries to limit the number of updates of the active sets $\mathcal{A}^k$ by using the proportionality test. Given $\Gamma > 0$, we call the iteration z^k *proportional* if the inequality

$$\|\boldsymbol{\beta}(z^k)\| \leq \Gamma \|\boldsymbol{\varphi}(z^k)\| \tag{9}$$

holds [13]. This test tries to limit both, the unnecessary expanding of the active set and the premature releasing of the constraints from the set $\mathcal{A}^k$ by balancing the violation of the KKT conditions on the active and the free set.

As the proposed algorithm can add or remove an arbitrary number of constraints at each iteration, we distinguish two phases of the active-set update:

(1) adding new constraints to $\mathcal{A}^k$,
(2) removing constraints from $\mathcal{A}^k$.

Let us assume that in the kth iteration of the algorithm, we need to add n_a and remove n_r constraints. If $|\mathcal{A}^k| = m$, then adding new constraints can be executed in $\mathcal{O}(n_a(n-m)^2)$ flops by the successive application of Given's rotations and the removal of constraints can be executed in $\mathcal{O}(n_r(n-m-n_a)^2)$ flops, see [20] for details.

Remark 3 Since the sequence of problem (1) is typically solved in the MPC for $\boldsymbol{x}_k \approx \boldsymbol{x}_{k+1}$, it is possible to use a solution from the previous sample as an initial iteration. Moreover, as the solution represents a future trajectory of control inputs in MPC, one can shift the solution from the previous sample to improve the initial iteration to establish a *warm-start* set-up, see, e.g. [33] and references therein. Further, it is possible to reuse the factor of the reduced Hessian in (7) to establish a so-called *hot start* of the algorithm.

4.4 *Proportioning step*

If an iteration is not proportional, then we employ the proportioning step

$$z^{k+1} = \mathcal{P}_\Omega(z^k - \alpha \boldsymbol{\beta}(z^k))$$

with the fixed step-length $\alpha \in (0, 2\|\mathbf{H}\|^{-1})$. Hence, all constraint indices indicated as not-optimal by nonzero component of $\boldsymbol{\beta}(z^k)$ will be removed from the active set. We select to project the chopped gradient instead of the gradient as in CGNP [32], to disable adding of new constraints to the active set by the proportioning step. This limits unwanted changes in the active set.

4.5 *Gradient updates*

The gradient at the minimizer of the face problem (6) has the form

$$\boldsymbol{g}(\tilde{z}^k) = \boldsymbol{g}(z^k - \boldsymbol{p}^k) = \mathbf{H}(z^k - \boldsymbol{p}^k) + \boldsymbol{h} = \boldsymbol{g}(z^k) - \mathbf{H}\boldsymbol{p}^k.$$

From the KKT conditions (5), we have

$$g_i(\tilde{z}^k) = 0 \quad \text{for } i \in \mathcal{I}\backslash\mathcal{A},$$

and so

$$(\mathbf{H}\boldsymbol{p}^k)_i = g_i(\boldsymbol{z}^k) \quad \text{for } i \in \mathcal{I}\backslash\mathcal{A}.$$

Hence, only the components of $(\mathbf{H}\boldsymbol{p}^k)_i$, $i \in \mathcal{A}$ with the sparsity pattern of $\boldsymbol{p}$ (7) need to be evaluated in each step, which requires $2m(n-m)$ flops.

4.6 *Projected line search*

In order to expand the active set, the proposed algorithm uses the projected path in a direction $\boldsymbol{p}^k$ evaluated via PLS of [31]. The PLS is used to find the first local minimizer along the path towards the optimizer of (6). The PLS operation starts by computing first the breakpoints, i.e. the step-sizes that correspond to the changes of the direction along the projected path. It can be observed that the resulting search direction $\boldsymbol{d}^{k,j}$ on the jth segment between two consecutive breakpoints is identical to $\boldsymbol{p}^k$ except the components corresponding to the newly activated constraints which are set to zero. We continue until the local minimizer in the segment, the Cauchy point, is found or an increase in the cost function q is detected. The search direction differs from the previous segments typically by one component only. This can be exploited to reduce the computational complexity of PLS to $\mathcal{O}(ns)$ flops, where s is the number of changes in the active set.

Moreover, as PLS gradually moves along the projected path with the updated $\mathbf{H}\boldsymbol{d}^{k,j}$, it is possible to avoid the direct computation of the gradient at the Cauchy point and update it in $\mathcal{O}(ns)$ flops. The update is done gradually with $\Delta t^{k,j} > 0$ step-size on the jth segments as

$$\boldsymbol{g}(\boldsymbol{z}^{k,j}) = \boldsymbol{g}(\boldsymbol{z}^{k,j-1}) - \Delta t^{k,j}\mathbf{H}\boldsymbol{d}^{k,j} \tag{10}$$

for $j = 0, \ldots, s-1$. The complexity of this step is described in the following lemma.

LEMMA 4.1 *Let $\boldsymbol{z} \in \mathbb{R}^n$, $m^k = |\mathcal{A}^k|$, let $\boldsymbol{p}^k$ be the solution of* (7)*, and let $\mathbf{H}\mathbf{p}^k$ be precomputed. Then the computational complexity of generating the projected path in a direction $\boldsymbol{p}^k$ utilizing the PLS algorithm with updating the gradient is*

$$\mathcal{O}_{\text{PLS}}(n) = 2(n-m^k) + \sum_{i=1}^{s}(2nr_i + 10n),$$

where s is the number of explored line segments and r_i is the number of changes in the active set on the ith explored line segment.

Proof First observe that $\boldsymbol{p}^k_{\mathcal{A}^k} = \mathbf{0}$, so the computation of the breakpoints can be executed in $2(n-m^k)$ flops. Second, at the ith segment, $\mathbf{H}\boldsymbol{d}$ can be updated in $2nr_i$ flops, where r_i is the number of the changes in the active set since r_i zeros were created in $\boldsymbol{d}$. The minimizer along the ray requires $6n$ flops, see [31] for details, the move to the end of a line segment requires $\boldsymbol{z}^{k,j} = \boldsymbol{z}^{k,j-1} + \Delta t^{k,j}\boldsymbol{d}^{k,j}$ in $2n$ flops, and the gradient can be updated (10) in $2n$ flops. The final result is the sum of all terms. ■

5. Algorithm

In each step, the proposed algorithm tries to reduce effectively the part of the gradient which dominates the error in the KKT conditions. If it is the free gradient, i.e. if the iterate is proportional, then the algorithm uses the direction $\boldsymbol{p}$, otherwise it uses the chopped gradient. The proportioning strategy has been proposed independently by Friedlander and Martinez [16] and Dostál [11] for the solvers combining the conjugate gradients and projecting steps. The proportioning with second-order information (PSOI) algorithm is presented in Algorithm 1.

Algorithm 1 PSOI Algorithm. Given an SPD matrix $\mathbf{H}$ of the order n, n-vectors $\boldsymbol{h}$, $\underline{z}$, $\bar{z}$, $\Omega = \{z : \underline{z} \leq z \leq \bar{z}\}$, $z^0 \in \Omega$, $\Gamma > 0$, $\alpha \in (0, 2\|\mathbf{H}\|^{-1})$ and $\varepsilon > 0$.

{Initialization}
Set $k = 0$, $\boldsymbol{g}^k = \mathbf{H}z^k + \boldsymbol{h}$
while $\|\boldsymbol{v}(z^k)\| \geq \varepsilon$ **do**
  if $\|\boldsymbol{\beta}(z^k)\| \leq \Gamma\|\boldsymbol{\varphi}(z^k)\|$ **then**
    {Proportional z^k}
    Solve (7) to obtain direction $\boldsymbol{p}^k$.
    Compute $\mathbf{H}\boldsymbol{p}^k$ as described in Section 4.5.
    $\alpha_f = \max\{\alpha : z^k - \alpha\boldsymbol{p}^k \in \Omega\}$
    if $\alpha_f < 1$ **then**
      {Expansion step}
      $[z_c^k, \boldsymbol{g}(z_c^k)] = \mathrm{PLS}(z^k, \boldsymbol{p}^k, \mathbf{H}\boldsymbol{p}^k, \boldsymbol{g}^k)$
      $z^{k+1} = z_c^k$
      $\boldsymbol{g}^{k+1} = \boldsymbol{g}(z_c^k)$
    else
      {Full direction $\boldsymbol{p}^k$ can be applied to remain feasible}
      $z^{k+1} = z^k - \boldsymbol{p}^k$
      $\boldsymbol{g}^{k+1} = \boldsymbol{g}^k - \mathbf{H}\boldsymbol{p}^k$
    end if
  else
    {Proportioning step}
    $z^{k+1} = \mathcal{P}_\Omega(z^k - \alpha\boldsymbol{\beta}(z^k))$
    $\boldsymbol{g}^{k+1} = \mathbf{H}z^{k+1} + \boldsymbol{h}$
  end if
  $k = k + 1$
end while
$z^* = z_k$

5.1 *Computational complexity and memory requirements*

The overview of computational complexity of each step of Algorithm 1 shows that the complexity of each iteration is $\mathcal{O}(n^2)$ flops. The cost of iterates varies, as it depends on the test (9) which part of the algorithm will be executed. The most computationally expensive part of Algorithm 1 is the computation of the direction $\boldsymbol{p}$ in (7), since it involves the update of the Cholesky factorization of the reduced Hessian and two substitutions.

The memory requirements of the proposed algorithm can be divided into two parts. First, the QP problem data have to be stored in read only memory. In our implementation, we exploit the symmetry of $\mathbf{H}$, hence only the upper triangular part, $0.5(n^2 + n) + n$ floating-point numbers, has to be stored. The proposed method has been implemented without any dynamic memory allocation. This is important for the embedded application, where the dynamic allocation is slow or prohibited due to the safety reasons. Our implementation involves allocation of six temporary n-vectors and a matrix with n^2 floating numbers. This is used to speed-up the process of the Cholesky factor update, using the lower triangular part of the matrix as temporary space when a new constraint is added to the active set. In this case, the column of the factor is removed and the rest of the factor is moved to fill in the gap in the column. Potential reduction of the memory requirements to $0.5(n^2 + n) + 6n$ can be achieved by using pointers, so that there is no need to move the factor columns in the memory.

6. Convergence

The proof of convergence is based on a simple estimate of the decrease of the cost function q in the proportioning step.

LEMMA 6.1 *Let*

$$\boldsymbol{d} = \bar{z} - \underline{z}, \quad z \in \Omega, \quad \boldsymbol{\beta} = \boldsymbol{\beta}(z), \quad \alpha \in (0, 2\|\mathbf{H}\|^{-1}),$$

and let $\tilde{\boldsymbol{\beta}}$ *be defined by its components*

$$\tilde{\beta}_i = \begin{cases} \min\{\beta_i, d_i/\alpha\} & \text{for } z_i = \bar{z}_i, \\ -\min\{-\beta_i, d_i/\alpha\} & \text{for } z_i = \underline{z}_i, \\ 0 & \text{elsewhere.} \end{cases} \tag{11}$$

Then there exists $\mu > 0$ *independent of* $\boldsymbol{z}$, *such that*

$$q(z) - q(P_\Omega(z - \alpha\boldsymbol{\beta})) \geq \mu\|\tilde{\boldsymbol{\beta}}\|^2. \tag{12}$$

Proof Let $\alpha = \delta\|\mathbf{H}\|^{-1}$, $\delta \in (0, 2)$. Using the Taylor formula, the observation $\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{g} = \tilde{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{\beta}$ (from definitions (5) and (11)), and

$$\alpha\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\mathbf{H}\tilde{\boldsymbol{\beta}} = \delta\|\mathbf{H}\|^{-1}\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\mathbf{H}\tilde{\boldsymbol{\beta}} \leq \delta\|\tilde{\boldsymbol{\beta}}\|^2,$$

we get

$$\begin{aligned} q(z) - q(P_\Omega(z - \alpha\boldsymbol{\beta})) &= q(z) - q(z - \alpha\tilde{\boldsymbol{\beta}}) \\ &= \alpha\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{g} - \frac{\alpha^2}{2}\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\mathbf{H}\tilde{\boldsymbol{\beta}} = \alpha\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{\beta} - \frac{\alpha^2}{2}\tilde{\boldsymbol{\beta}}^{\mathrm{T}}\mathbf{H}\tilde{\boldsymbol{\beta}} \\ &\geq \alpha\|\tilde{\boldsymbol{\beta}}\|^2 - \frac{\alpha\delta}{2}\|\tilde{\boldsymbol{\beta}}\|^2 = \left(\delta - \frac{\delta^2}{2}\right)\|\mathbf{H}\|^{-1}\|\tilde{\boldsymbol{\beta}}\|^2 \\ &= \mu\|\tilde{\boldsymbol{\beta}}\|^2. \end{aligned}$$

This proves (12) with $\mu = (\delta - \delta^2/2)\|\mathbf{H}\|^{-1}$. ■

Now we are ready to prove the convergence of the proposed algorithm.

THEOREM 6.2 *Let* $\{z^k\}$ *denote the iterates generated by the PSOI algorithm for the solution of* (1) *with* $z^0 \in \Omega$, $\alpha \in (0, 2\|\mathbf{H}\|^{-1})$, *and* $\Gamma > 0$. *Then,* $\{z^k\}$ *converges to the solution* z^* *of* (1).

Proof First notice that $\{q(z^k)\}$ is decreasing and bounded from below by the unconstrained minimum of q. Moreover, if $\{z^k\}$ is infinite, then it has an infinite subsequence of disproportional iterates—there can be at most n consecutive expansion steps since the active set is expanding at each step by at least one index. Since Ω is compact, it follows that there is a set of indices $\mathcal{K}$ of

the proportioning steps such that $\{z^k\}_{k\in\mathcal{K}}$ converges to $\hat{z}$ and

$$q(\hat{z}) = \inf\{q(z^k)\}_{k\in\mathcal{K}}.$$

Using Lemma 6.1, we get $\mu > 0$ such that

$$q(z^0) - q(\hat{z}) \geq \sum_{k\in\mathcal{K}} \mu \|\tilde{\boldsymbol{\beta}}(z^k)\|^2,$$

so $\|\tilde{\boldsymbol{\beta}}(z^k)\|$ converges to zero. However, after inspecting the definition of $\tilde{\boldsymbol{\beta}}$, it follows that also $\|\boldsymbol{\beta}(z^k)\|$ converges to zero. Since for a given $\Gamma > 0$ the disproportional iterates satisfy

$$\Gamma\|\varphi(z^k)\| \leq \|\beta(z^k)\|,$$

we conclude that also $\varphi(z^k)$ converges to zero and so does $\boldsymbol{\nu}(z^k)$. Thus $\boldsymbol{\nu}(\hat{z}) = \mathbf{0}$ and, since the solution is unique, also $\hat{z} = z^*$ and $\{z^k\}$ converges to z^*. ■

Remark 4 The proportioning step is a simple and efficient way of releasing the indices from the active set. However, there are some other options. For example, we can try to use (possibly reduced) conjugate gradient step length in the direction $\boldsymbol{\beta}(z^k)$ or the GP step. The latter is supported by the following theorem which guarantees sufficient decrease in the cost function.

THEOREM 6.3 *Let z^* denote the unique solution of* (1), *let $\lambda_{\min}(\mathbf{H})$ denote the smallest eigenvalue of $\mathbf{H}$, $z \in \Omega$, $\boldsymbol{g} = \mathbf{H}z + \boldsymbol{h}$, and $\alpha \in (0, 2\|\mathbf{H}\|^{-1})$. Then*

$$q(P_\Omega(z - \alpha\boldsymbol{g})) - q(z^*) \leq \eta(\alpha)(q(z) - q(z^*)),$$

where

$$\eta(\alpha) := \max\{1 - \alpha\lambda_{\min}, 1 - (2\|\mathbf{H}\|^{-1} - \alpha)\lambda_{\min}\}.$$

Proof See Bouchala *et al.* [6]. ■

7. Numerical experiments

In this section, the performance of the proposed method is compared with the state-of-the-art solvers in several experiments. In particular, we are referring to qpOASES, an ASM implementation of Ferreau *et al.* [14] with hot start functionality used in all experiments together with the option for MPC problems; FiOrdOs [47], FGM with an automatically generated code and diagonal preconditioner, and FORCES [10] with automatically generated IPM solver for (MPC). All algorithms use default settings except FiOrdOs, where we set the maximum number of iterations to 200 and stopping condition to reach ε_{FGM}-solution with respect to the optimal cost function value via the so-called gradient map-based stopping condition as proposed in [42]. We use $\varepsilon_{\text{FGM}} = 1e{-}3$, otherwise no effort has been taken to optimize it to the experiments.

The presented algorithm PSOI was implemented in ANSI-C language using the single-precision floating-point arithmetic (4 bytes per floating-point number) and was, as the other solvers, called from MATLAB environment via C-MEX interface with $\Gamma = 1$, $\alpha = 1.95\|\mathbf{H}\|^{-1}$, and $\varepsilon = 1e{-}6\|\boldsymbol{h}\|$. All numerical experiments were executed on a laptop with Intel i7-4800MQ 2.7 GHz processor. Since in the embedded application, the solution has to be ready before the next sampling time under all circumstances, we recorded the maximum computation time for each simulation. To eliminate outliers in execution times, the solvers were called 50 times for

each QP problem and the minimum time measured was accepted. The reported times are the results of the max–min operation for each simulation, if not stated otherwise.

All solvers have been using warm-start strategy using a shifted solution from the previous sample instant, cf. Remark 3. Additionally, PSOI and qpOASES reused the factors from the previous sample instant.

7.1 *Oscillating masses*

The set-up of the first experiment is similar to [48]. It consists of a sequence of six masses connected by spring dampers to each other. The first and the last masses are connected to the walls. The weight of each mass is 1 kg and 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 three control inputs, i.e. $\boldsymbol{u} \in \mathbb{R}^3$, 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 $\boldsymbol{w} \in \mathbb{R}^6$ with a uniform distribution on $[-0.5, 0.5]$ which acts additionally on the displacement state of each mass, see [48] for more details about the set-up. The control objective is to stabilize each mass in its origin, i.e. to solve (3) with $\mathbf{R} = \mathbf{I}$ and $\mathbf{Q} = \mathbf{P} = \mu\mathbf{I}$, $\mu > 0$ with sampling time $T_s = 0.5$ s. In order to show the dependency of the solver performance on the number of variables, we perform simulation of the model and controller with the prediction horizon $N \in \{10, 20, \ldots 70\}$. Furthermore, to test the solver performance for different conditioning of the QP problems, we choose two different μ: 1) *low-conditioned* with $\mu = 1$ where $\kappa(\mathbf{H}) \approx 1e2$ and 2) *moderate-conditioned* with $\mu = 1e3$ with $\kappa(\mathbf{H}) \approx 1e3$. Hence, the growth in $\kappa(\mathbf{H})$ is caused only by the change of controller tuning. Note that we are referring to the conditioning of the QP problem in the sense of the spectral condition number of the $\mathbf{H}$. All simulations were carried out for 1000 seconds, i.e. 2000 sampling instants and the computation times during the simulation for a fixed prediction horizon were saved.

Both Figure 1 and 2 show that all the solvers are several times faster than FORCES, although it is the only solver in test which has linear computational complexity of each iteration with respect

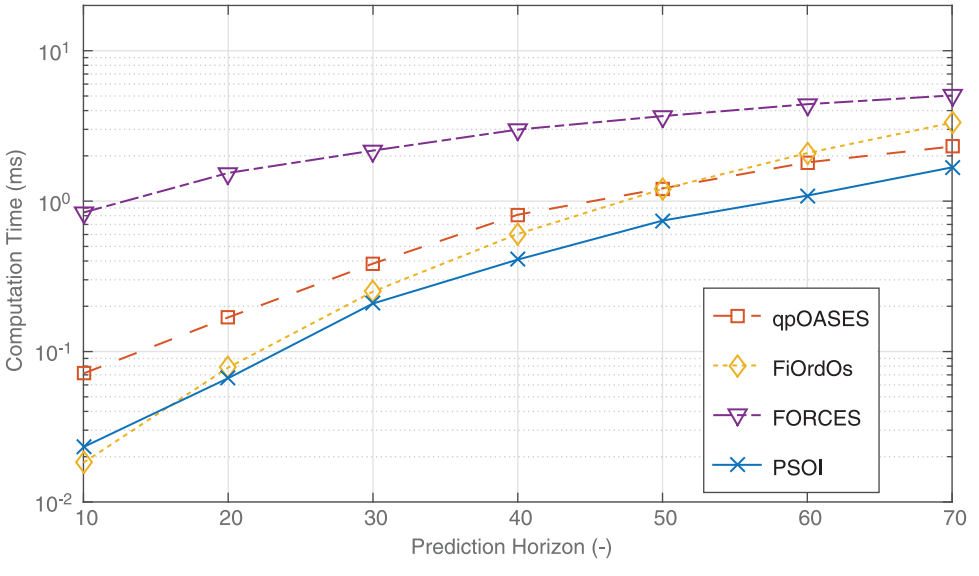

Figure 1. Maximum computation time of simulation of low-conditioned oscillating masses as a function of prediction horizon.

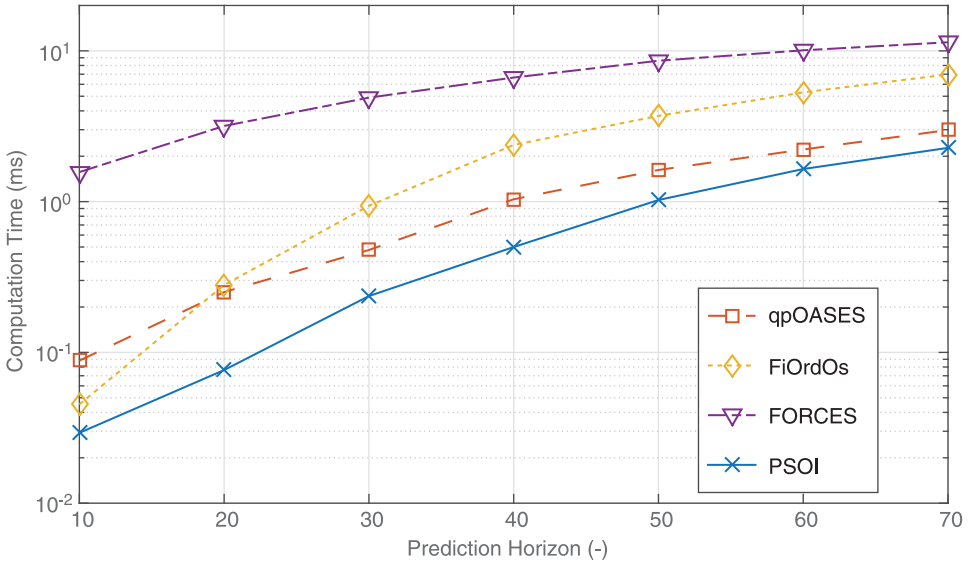

Figure 2. Maximum computation time of simulation of moderate-conditioned oscillating masses as a function of prediction horizon.

to the prediction horizon. To our understanding, this is because the multiplicative constant of linear complexity is high; hence on a given prediction horizon range the complexity of each iteration is actually higher than for other solvers. It is also evident that FiOrdOs is very sensitive to the problem conditioning, leading to the fact that it is faster than PSOI up to $N = 10$ for low-conditioned problems but more than three times slower than PSOI for moderate-conditioned problems. It should be also noted that FiOrdOs uses automatic diagonal preconditioning of the problem Hessian off-line to minimize the condition number, hence it solves numerically more favourable QP problem than other solvers. On the other hand, qpOASES and PSOI indicate small sensitivity to QP problem conditioning. However, compared to qpOASES, which needs as many iterations as is the difference in the active set from one sample instant to another, the projection in PSOI helps to reduce the computation time.

To compare the methods in more detail, we employ the standard Dolan–More performance profiles presented in [9]. These profiles visualize how many percent of all problem instances are solved within τ—factor times the time of the fastest solver for this instance. Hence, the computation times of each QP problem from the simulation with oscillating masses have been merged for all selection of prediction horizon N separately for each solver. Hence, there were exactly $2000 \cdot 7 = 14,000$ QP problem instances solved by each solver and the particular choice of μ. Figure 3 shows that 74.4% of QP problem instances are solved by the PSOI as the fastest solver and that 96.9% instances are solved by PSOI solver within two times (i.e. $\tau = 2$) the fastest solver for low-conditioned oscillating masses. The qpOASES solves 73.3% of instances for $\tau = 2$, while FiOrdOs 60.1% instances for $\tau = 3$. The poor performance of the FORCES solver is indicated by the fact that it solves only 30.6% instances within 10 times the computation time of the fastest solver.

The sensitivity of the FiOrdOs to the QP problem conditioning is demonstrated in Figure 4 for moderate-conditioned oscillating masses as it solves only 5.5% problem instances for $\tau = 3$. On the other hand, PSOI solves 80.5% of problem instances with the lowest computation time and 97.7 % with $\tau = 2$. The qpOASES then solves 60.5% of problem instances for $\tau = 2$ showing only small sensitivity to the QP problem conditioning.

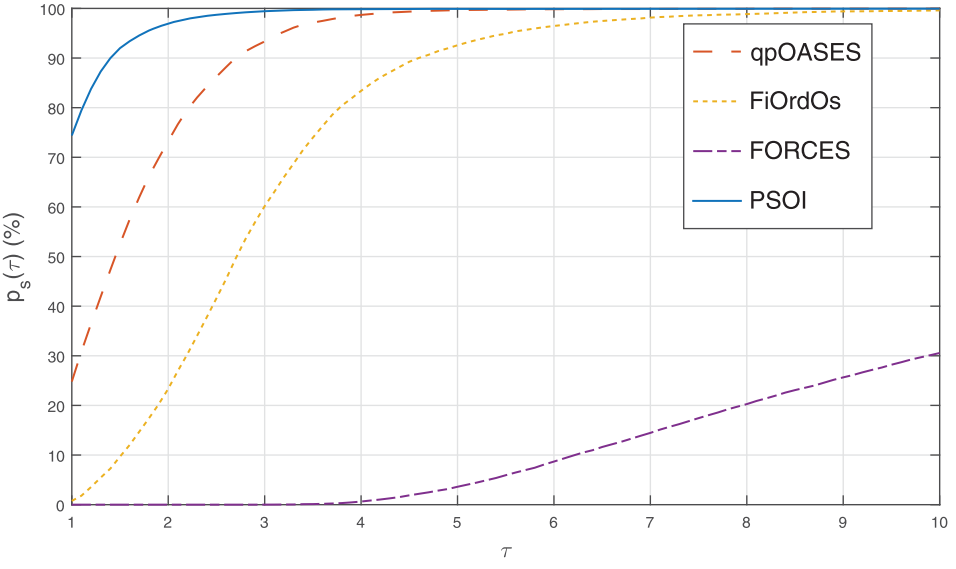

Figure 3. Performance profiles of Dolan and Moré [9] for computation times of solvers in test for low-conditioned oscillating masses with all settings of prediction horizon.

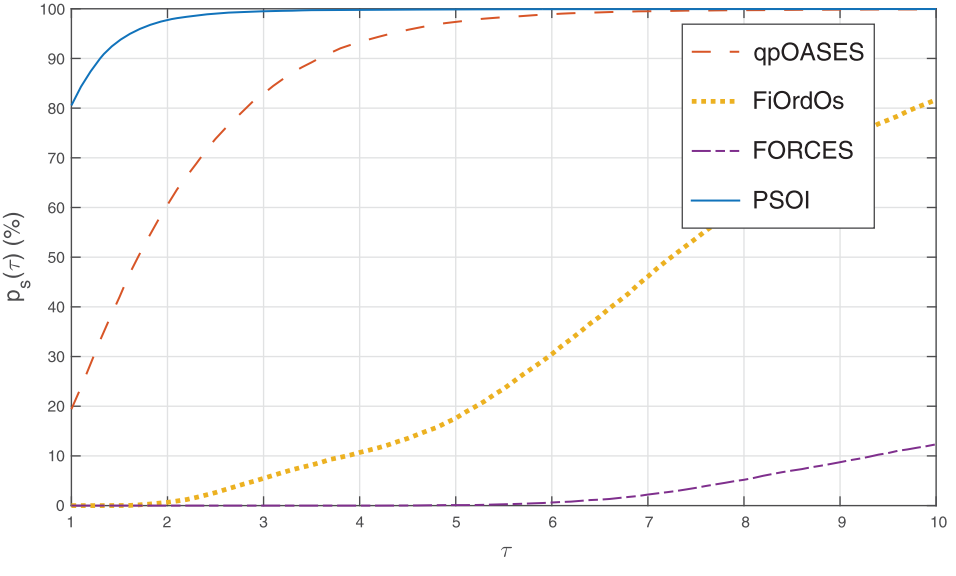

Figure 4. Performance profiles of Dolan and Moré [9] for computation times of solvers in test for moderate-conditioned oscillating masses with all settings of prediction horizon.

7.2 *Random system*

The controlled system is often close to the steady-state conditions where typically no constraints are activated. This is modified by either change of the setpoints, limits, or the effect of external disturbances leading to transient operation of the MPC controller which tries to stabilize the system back. During the transient operation, the constraints are activated as the actuators of the plant are saturated to speed-up the stabilization process. This leads to the fact that the optimal active set has to be updated by the QP solver.

In the following experiment, the proposed method is compared to the state-of-the-art solvers to investigate how quickly the optimal active set is identified during the MPC controller transient operation when the external disturbance causes changes in the system state $\boldsymbol{x}$ from the steady state $\boldsymbol{x} = \mathbf{0}$. In the previous experiment, it was shown how the proposed method can use a solution from the previous sample time when there is only slight change in $\boldsymbol{x}$. Contrarily, the experiment suggested in this section clarifies the behaviour of the proposed method when the solution from previous sample time serves as poor initial iteration. The reason for that is a large change of the system state due to the external disturbance. As such, this experiment serves as a tool for estimating the worst-case performance of the methods which can be expected during the transient operation of MPC controller.

To this end, a random linear Schur stable[2] discrete-time system with $n_x = 15$ and $n_u = 5$ was generated by Matlab's function `drss`. The control limits were set as $-0.1 \leq \boldsymbol{u} \leq 0.1$ and the controller was tuned with $\mathbf{R} = \mathbf{I}$, $\mathbf{Q} = 1e3 \cdot \mathbf{I}$ and $\mathbf{P}$ as solution of Lyapunov equation, see [2]. The controller has been set with enlarging prediction horizon $N \in \{10, 20, \ldots, 70\}$. The resulting QP problem conditioning has been $\kappa(\mathbf{H}) \approx 1e3$. For each value of N, there were randomly generated 1000 current state measurements $\boldsymbol{x}$ where each component has a uniform distribution on $[-10, 10]$. The centre of the feasible set has been used as an initial iteration for all generated $\boldsymbol{x}$, and all solvers were initialized before running at steady-state condition $\boldsymbol{x} = \mathbf{0}$. The solution from the steady-state conditions have been used as an initial iteration for all solvers.

Figure 5 illustrates the maximal computation times for various state measurements $\boldsymbol{x}$ leading to the same cardinality of optimal active set $|\mathcal{A}^*|$. It is evident that qpOASES needs more than twice the computation time compared to PSOI for large $|\mathcal{A}^*|$ since only one constraint can be added at

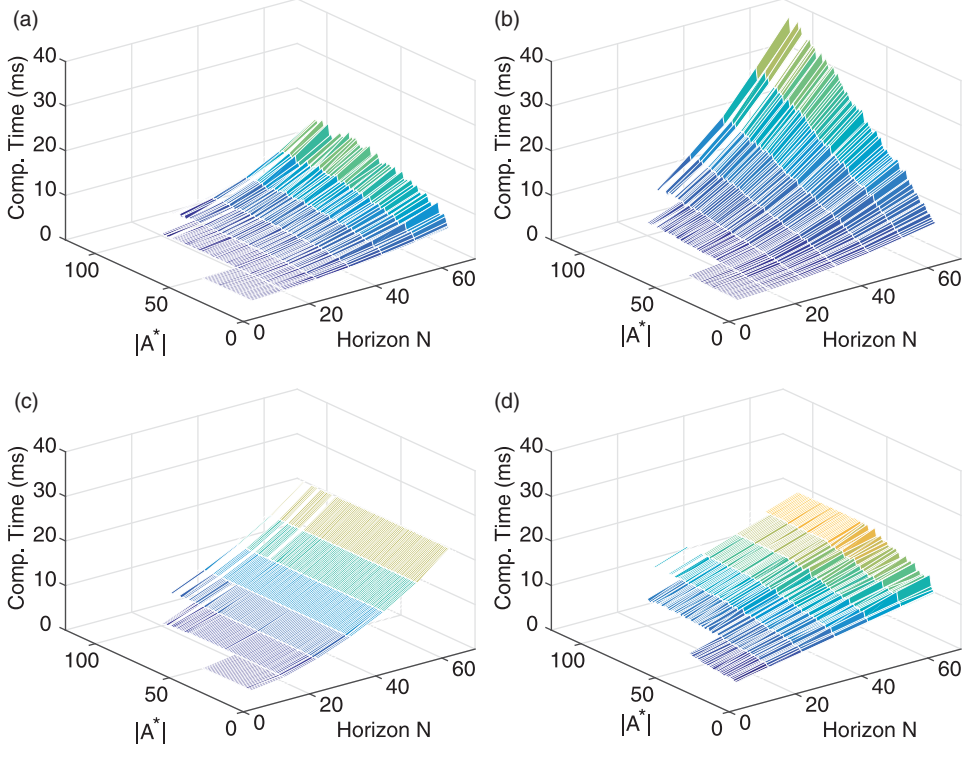

Figure 5. Comparison of maximal solver times for random system with variable state measurement $\boldsymbol{x}$ leading to various cardinality of optimal active set $|\mathcal{A}^*|$. (a) PSOI, (b) qpOASES, (c) FiOrdOs, and (d) FORCES.

one iteration. On the other hand, FiOrdOs shows insensitivity to $|\mathcal{A}^*|$ since the maximum number of iterations is exceeded with maximal relative cost function error of 0.001%.

7.3 *Diesel engine tracking*

To show practical use of the proposed method, the solvers were used for the solution of MPC problem arising in the control of the turbo diesel engine. A nonlinear control-oriented model of turbocharged 2.2 litre diesel engine has been built using OnRAMP Design Suite tool [23]. A linear model at an engine speed 1500 rpm and an injection quantity of 40 mg/stroke was derived by linearization inside the tool with resulting 23 states. An exhaust gas recirculation valve (EGR) and turbo with variable geometry (VGT) were used as actuators. Similarly to [15], the mass air flow (MAF) and the intake manifold absolute pressure (MAP) were tracked to the references. Hence, we designed an MPC controller leading to the following optimization problem

$$
\begin{aligned}
f(\boldsymbol{x})^* = \min_{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{N-1}} \quad & \frac{1}{2}\sum_{k=1}^{N}((\boldsymbol{y}_k - \boldsymbol{r}_k)^{\mathrm{T}}\mathbf{Q}_y(\boldsymbol{y}_k - \boldsymbol{r}_k) + \Delta\boldsymbol{u}_k{}^{\mathrm{T}}\mathbf{R}\Delta\boldsymbol{u}_k), \\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} = \mathbf{A}\boldsymbol{x}_k + \mathbf{B}\boldsymbol{u}_k, \quad k = 0,\ldots,N-1, \\
& \boldsymbol{y}_k = \mathbf{C}\boldsymbol{x}_k, \quad k = 1,\ldots,N, \\
& \Delta\boldsymbol{u}_k = \boldsymbol{u}_k - \boldsymbol{u}_{k-1}, \quad k = 1,\ldots,N, \\
& \boldsymbol{u}_k \in \tilde{\Omega}, \quad k = 0,\ldots,N-1, \\
& \boldsymbol{x}_0 = \boldsymbol{x},
\end{aligned}
\tag{13}
$$

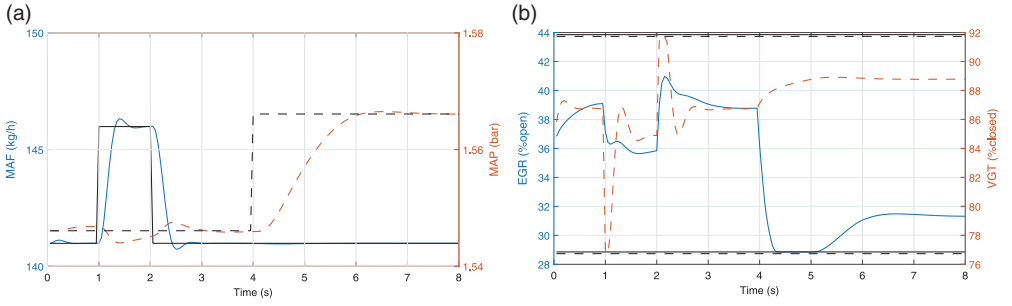

Figure 6. Diesel engine tracking problem simulation for $N_{\text{block}} = 100$. (a) The MAF and its setpoint are in solid lines, whereas MAP are in dashed lines. (b) The EGR and its limits are in solid lines, while VGT are in dashed lines.

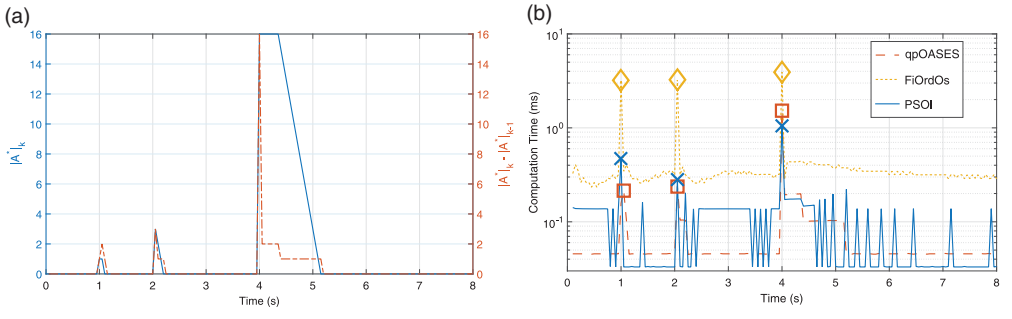

Figure 7. Diesel engine tracking problem simulation for $N_{\text{block}} = 100$. (a) The number of active constraints is in solid line and the number of changes in the optimal active set from last sample instant in dashed line. (b) The computation times during experiment. Markers show three largest computation times for each solver.

Table 1. Maximal and mean involved solver iterations (in bracket) during the engine tracking problem simulation.

N_{block}	n	$\kappa(\mathbf{H})$	PSOI (–)	qpOASES (–)	FiOrdOs (–)
10	20	2.2e4	4 (1.2)	4 (0.2)	63 (9.8)
20	40	3.0e4	5 (1.1)	11 (0.3)	64 (5.8)
30	60	4.3e4	5 (1.1)	17 (0.3)	67 (4.2)
40	80	4.5e4	7 (1.1)	15 (0.3)	69 (3.2)
50	100	3.8e4	9 (1.1)	15 (0.3)	70 (3.0)
60	120	3.1e4	7 (1.0)	16 (0.3)	73 (3.0)
70	140	2.1e4	7 (1.0)	16 (0.3)	78 (3.0)
80	160	9.6e3	7 (0.9)	16 (0.3)	81 (3.0)
90	180	2.4e3	7 (0.7)	16 (0.3)	88 (3.6)
100	200	5.1e3	7 (0.5)	16 (0.3)	128 (10.9)

Table 2. Maximal and mean solver computation times (in bracket) during the engine tracking problem simulation.

N_{block}	n	$\kappa(\mathbf{H})$	PSOI (ms)	qpOASES (ms)	FiOrdOs (ms)
10	20	2.2e4	0.010 (0.003)	0.023 (0.006)	0.024 (0.011)
20	40	3.0e4	0.032 (0.006)	0.081 (0.009)	0.071 (0.016)
30	60	4.3e4	0.074 (0.013)	0.174 (0.012)	0.159 (0.022)
40	80	4.5e4	0.146 (0.022)	0.249 (0.017)	0.283 (0.030)
50	100	3.8e4	0.262 (0.033)	0.366 (0.023)	0.466 (0.041)
60	120	3.1e4	0.336 (0.046)	0.543 (0.030)	0.750 (0.058)
70	140	2.1e4	0.471 (0.063)	0.719 (0.039)	1.143 (0.078)
80	160	9.6e3	0.671 (0.086)	1.078 (0.049)	1.552 (0.097)
90	180	2.4e3	0.843 (0.092)	1.196 (0.058)	2.184 (0.137)
100	200	5.1e3	1.049 (0.092)	1.529 (0.071)	3.907 (0.384)

where $\mathbf{C} \in \mathbb{R}^{n_x \times n_y}$ is the output matrix, $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$ denotes the system output, $\boldsymbol{y} = [MAF, MAP]^{\mathrm{T}}$, $\boldsymbol{r}_k \in \mathbb{R}^{n_y}$ is the reference vector, $\mathbf{Q}_y \in \mathbb{R}^{n_y \times n_y}$ is the SPS matrix, and $\Delta \boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the change of control input $\boldsymbol{u} = [EGR, VGT]^{\mathrm{T}}$ where $\Delta \boldsymbol{u}_{-1}$ denotes the last control input from the previous sample. The form of the problem (13) assures zero tracking error in nominal conditions [28] and can be formulated as (1), see, e.g. [43] for details. The engine model was discretized with a sampling period 50 ms and the prediction horizon was selected as 5 s. The controller was tuned with $\mathbf{R} = \mathrm{diag}(0.1, 0.1)$ and $\mathbf{Q}_y = \mathrm{diag}(1, 200)$. Additionally, to decrease the number of optimization variables to $n = N_{block} n_u$, the input blocking strategy described in [8] was employed. It keeps the inputs constant over N_{block} time-steps, so-called blocks, while having usually a minor effect on the controller performance.

To benchmark the solvers, practical trajectories of MAF and MAP references were set in the simulation with number of blocks as $N_{block} \in \{10, 20, \ldots, 100\}$. The resulting QP problem conditioning has been $\kappa(\mathbf{H}) \approx 5e4$, see Table 2. The limits for both actuators were selected as -10% and $+5\%$ from the linearization point. Step changes of 5 kg/h in MAF and 0.02 bar in MAP reference in Figure 6 led to the saturation of actuators, leading to the activation of new constraints, see Figure 7(a). This results in an increase in the demanded number of iterations of all solvers, which is evident from the difference of maximal and mean values of iterations in Table 1 and solution times in Figure 7(b) and in Table 2. While the number of iterations increases dramatically for solvers in the test, the maximum number of iterations of PSOI remains under nine in all instances since the fast identification of the optimal active set by projection is used. As a result, it appears that PSOI is two to three times faster than FiOrdOs and about 50% faster in the worst solution time than qpOASES for all choices of number of blocks. Since FORCES package cannot directly solve the problem with the cost function of type (13), it was not involved in this test.

8. Conclusion

A new algorithm for a strictly convex box constrained QP problem arising in MPC has been introduced. The proposed method uses a combination of $\boldsymbol{p}$ direction projection and proportioning. While the first ingredient allows for quick expansion of the active set, the latter one avoids premature releasing of the constraints. This enables fast identification of the optimal active set and reduces the computational complexity of the factor update scheme used in the face problem solution. The algorithm has been proved to converge, and the experiments show that the performance of the proposed method is comparable to or better than other state-of-the-art methods.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was partly supported (MJ, ZD) by the Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project 'IT4Innovations excellence in science—LQ1602 '. OS was partly supported by the Technology Agency of the Czech Republic (TA01030170).

Notes

1. Note that problem (1) is not subject to equality constraints, hence the associated Lagrange multiplier is zero. The condition of the non-negativity of the Lagrange multipliers of the simple constraints is reduced to the first two rows of (4).
2. Schur stable system is such that all eigenvalues of **A** are within the open unit disk.

References

[1] D. Axehill and A. Hansson, *A dual gradient projection quadratic programming algorithm tailored for model predictive control*, in *47th IEEE Conference on Decision and Control*, December, IEEE, M. Egerstedt, ed., Cancun, 2008, pp. 3057–3064.
[2] A. Bemporad, M. Morari, V. Dua and E.N. Pistikopoulos, *The explicit linear quadratic regulator for constrained systems*, Automatica J. IFAC 38 (2002), pp. 3–20.
[3] A. Bemporad, A. Oliveri, T. Poggi and M. Storace, *Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations*, IEEE Trans. Autom. Control 56 (2011), pp. 2883–2897.
[4] D.P. Bertsekas, *On the Goldstein–Levitin–Polyak gradient projection method*, IEEE Trans. Automat. Control 21 (1976), pp. 174–184.
[5] F. Borrelli, M. Baotić, J. Pekar and G. Stewart, *On the computation of linear model predictive control laws*, Automatica J. IFAC 46 (2010), pp. 1035–1041.
[6] J. Bouchala, Z. Dostal and P. Vodstrcil, *Separable spherical constraints and the decrease of a quadratic function in the gradient projection step.*, J. Optim. Theory Appl. 157 (2013), pp. 132–140.
[7] S. Boyd and L. Vandenberghe, *Convex Optimization*, Vol. 98, Cambridge University Press, New York, 2004.
[8] R. Cagienard, P. Grieder, E. Kerrigan and M. Morari, *Move blocking strategies in receding horizon control*, J. Process Control 17 (2007), pp. 563–570.
[9] E.D. Dolan and J.J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program. 91 (2002), pp. 201–213.
[10] A. Domahidi, A.U. Zgraggen, M.N. Zeilinger, M. Morari and C.N. Jones, *Efficient interior point methods for multistage problems arising in receding horizon control*, in *51st IEEE Conference on Decision and Control*, IEEE, A. Astolfi, ed., Maui, 2012, pp. 668–674.
[11] Z. Dostál, *Box constrained quadratic programming with proportioning and projections*, SIAM J. Optim. 7 (1997), pp. 871–887.
[12] Z. Dostál, *Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities*, 1st ed., Springer Publishing Company Incorporated, New York, 2009.
[13] Z. Dostál and J. Schöberl, *Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination*, Comput. Optim. Appl. 30 (2005), pp. 23–43.
[14] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock and M. Diehl, *qpOASES: a parametric active-set algorithm for quadratic programming*, Math. Program. Comput. 6 (2014), pp. 1–37.

[15] H. Ferreau, P. Ortner, P. Langthaler, L. Re and M. Diehl, *Predictive control of a real-world Diesel engine using an extended online active set strategy*, Annu. Rev. Control 31 (2007), pp. 293–301.
[16] A. Friedlander and J. Martínez, *On the maximization of a concave quadratic function with box constraints*, SIAM J. Optim. 4 (1994), pp. 177–192.
[17] C. Garcia, D. Prett and M. Morari, *Model predictive control: theory and practice survey*, Automatica J. IFAC 25 (1989), pp. 335–348.
[18] P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright, *Procedures for optimization problems with a mixture of bounds and general linear constraints*, ACM Trans. Math. Softw. 10 (1984), p. 282.
[19] P. Giselsson, *Execution time certification for gradient-based optimization in model predictive control*, in *51st IEEE Conference on Decision and Control*, December, IEEE, A. Astolfi, ed., Maui, 2012, pp. 3165–3170.
[20] G.H. Golub and C.F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, MD, 1996.
[21] P. Grieder, *Complexity reduction of receding horizon control*, in *42nd IEEE Conference on Decision and Control*, 9–12, IEEE, A. Astolfi, ed., Maui, 2003, pp. 3179–3190.
[22] V. Havlena and J. Findejs, *Application of model predictive control to advanced combustion control*, Control Eng. Pract. 13 (2005), pp. 671–680.
[23] Honeywell: *OnRAMP Design Suite* (2016), http://www.honeywellonramp.com/.
[24] C.N. Jones and M. Morari, *Polytopic approximation of explicit model predictive controllers*, IEEE Trans. Autom. Control 55 (2010), pp. 2542–2553.
[25] A. Karnik, D. Pachner, A. Fuxman, D. Germann, M. Jankovic and C. House, *Model predictive control for engine powertrain thermal management applications*, SAE Technical Paper 2015-01-0336, SAE, Detroit, 2015.
[26] N. Khaled, J. Pekar, A. Fuxman, M. Cunningham and O. Santin, *Multivariable control of dual loop EGR diesel engine with a variable geometry turbo*, SAE Technical Paper 2014-01-1357, SAE, Detroit, 2014.
[27] Y. Kim, M. Van Nieuwstadt, G. Stewart and J. Pekar, *Model predictive control of DOC temperature during DPF regeneration*, SAE Technical Paper 2014-01-1165, SAE, Detroit, 2014.
[28] J. Maciejowski, *Predictive Control with Constraints*, Pearson Education, Harlow, 2001.
[29] D. Mayne, J.B. Rawlings, C.V. Rao and P.O.M. Scokaert, *Constrained model predictive control: Stability and optimality*, Automatica J. IFAC 36 (2000), pp. 789–814.
[30] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer Academic Publishers, New York, 2004.
[31] J. Nocedal and S. Wright, *Numerical Optimization*, Springer, New York, 1999.
[32] P. Otta, O. Šantin and V. Havlena, *Tailored QP algorithm for predictive control with dynamics penalty*, in *22nd Mediterranean Conference on Control & Automation*, IEEE, A. Fagiolini, ed., Palermo, 2014, pp. 384–389.
[33] P. Otta, O. Šantin and V. Havlena, *Measured-state driven warm-start strategy for linear MPC*, in *Proceedings of the European Control Conference*, Vol. 1, IEEE, T. Parisini, ed., Brussels, 2015, pp. 3137–3141.
[34] P. Patrinos and A. Bemporad, *An Accelerated Dual Gradient-Projection Algorithm for Embedded Linear Model Predictive Control*, IEEE Trans. Autom. Control 59 (2014), pp. 18–33.
[35] J. Pekar and G. Stewart, *Using model predictive control to optimize variable trajectories and system control* (2013), US Patent 8,504,175.
[36] S.J. Qin and T.A. Badgwell, *A survey of industrial model predictive control technology*, Control Eng. Pract. 11 (2003), pp. 733–764.
[37] C. Rao, J. Rawlings and D. Mayne, *Constrained state estimation for nonlinear discrete-time systems: stability and moving horizon approximations*, IEEE Trans. Autom. Control 48 (2003), pp. 246–258.
[38] C. Rao, S. Wright and J. Rawlings, *Application of interior-point methods to model predictive control*, J. Optim. Theory Appl. 99 (1998), pp. 723–757.
[39] S. Richter, C. Jones and M. Morari, *Real-time input-constrained MPC using fast gradient methods*, in *48th IEEE Conference on Decision and Control Conference*, December, IEEE, T. Parisini, ed., Shanghai, 2009, pp. 7387–7393.
[40] S. Richter, C.N. Jones and M. Morari, *Certification aspects of the fast gradient method for solving the dual of parametric convex programs*, Math. Methods Oper. Res. 77 (2012), pp. 305–321.
[41] S. Richter, C.N. Jones and M. Morari, *Computational Complexity Certification for Real-Time MPC With Input Constraints*, IEEE Trans. Autom. Control 57 (2012), pp. 1391–1403.
[42] S. Richter and M. Morari, *Stopping criteria for first-order methods*, Tech. Rep., ETH Zurich, 2012.
[43] O. Šantin and V. Havlena, *Combined partial conjugate gradient and gradient projection solver for MPC*, in *2011 IEEE Multi-Conference on Systems and Control*, IEEE, A. Astolfi, ed., Denver, 2011, pp. 1270–1275.
[44] O. Šantin and V. Havlena, *Gradient projection based algorithm for large scale real time model predictive control*, in *2011 Chinese Control and Decision Conference*, IEEE, H. Cao, ed., Mianyang, 2011, pp. 3906–3911.
[45] O. Šantin and V. Havlena, *Combined gradient and Newton projection quadratic programming solver for MPC*, in *18th IFAC World Congress*, IFAC, S. Andreassen, ed., Milano, 2011, pp. 5567–5572.
[46] O. Santin, J. Pekar, J. Beran, A. DAmato, E. Ozatay, J. Michelini, S. Szwabowski and D. Filev, *Cruise controller with fuel optimization based on adaptive nonlinear model predictive control*, SAE Technical Paper 2016-01-0155, SAE, Detroit, 2016.
[47] 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.
[48] Y. Wang and S. Boyd, *Fast model predictive control using online optimization*, IEEE Trans. Control Syst. Technol. 18 (2010), pp. 267–278.