MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Projection-free Parallel Quadratic Programming for Linear Model Predictive Control

Di Cairano, S.; Brand, M.; Bortoff, S.

TR2013-059 July 2013

Abstract

A key component in enabling the application of model predictive control (MPC) in fields such as automotive, aerospace and factory automation is the availability of low-complexity fast optimization algorithms to solve the MPC finite horizon optimal control problem in architectures with reduced computational capabilities. In this paper we introduce a projection-free iterative optimization algorithm and discuss its application to linear MPC. The algorithm, originally developed by Brand for non-negative quadratic programs, is based on a multiplicative update rule and it is shown to converge to a fixed point which is the optimum. An acceleration technique based on a projection-free line search is also introduced, to speed-up the convergence to the optimum. The algorithm is applied to MPC through the dual of the quadratic program (QP) formulated from the MPC finite time optimal control problem. We discuss how termination conditions with guaranteed degree of suboptimality can be enforced, and how the algorithm performance can be optimized by pre-computing the matrices in a parametric form.We show computational results of the algorithm in three common case studies and we compare such results with the results obtained by other available free and commercial QP solvers.

International Journal of Control

Projection-free Parallel Quadratic Programming for Linear Model Predictive Control

S. Di Cairano*,†, M. Brand†, S.A. Bortoff

Abstract

A key component in enabling the application of model predictive control (MPC) in fields such as automotive, aerospace and factory automation is the availability of low-complexity fast optimization algorithms to solve the MPC finite horizon optimal control problem in architectures with reduced computational capabilities. In this paper we introduce a projection-free iterative optimization algorithm and discuss its application to linear MPC. The algorithm, originally developed by Brand for non-negative quadratic programs, is based on a multiplicative update rule and it is shown to converge to a fixed point which is the optimum. An acceleration technique based on a projection-free line search is also introduced, to speed-up the convergence to the optimum. The algorithm is applied to MPC through the dual of the quadratic program (QP) formulated from the MPC finite time optimal control problem. We discuss how termination conditions with guaranteed degree of suboptimality can be enforced, and how the algorithm performance can be optimized by pre-computing the matrices in a parametric form. We show computational results of the algorithm in three common case studies and we compare such results with the results obtained by other available free and commercial QP solvers.

Index Terms

model predictive control, optimization, numerical algorithms, constrained control.

I. Introduction

Model predictive control (MPC) [1] is a powerful framework for optimal control of multivariable constrained systems. MPC is based on the repeated, receding horizon solution of a finite-time optimal control problem formulated from the system dynamics, the constraints on system states, inputs, and outputs, and the cost function describing the control objectives and their relative priority.

A fundamental component of any MPC strategy is the numerical algorithm for solving the finite-time optimal control problem. MPC was originally developed for applications in process control [2] where the systems to be controlled are significantly complex (hundreds of inputs and outputs), but the available computational power is large and the system time constants are also large. Although the resulting optimal control problems are of large scale, the readily available computer resources are generally sufficient to meet the relatively loose chronometric requirements. In recent years MPC has shown potential in other domains including automotive, aerospace, and factory automation [3]–[6], see also the recent surveys [7], [8]. In these applications the MPC controller is executed on embedded controllers with relatively low computational power, and the control update cycle is much faster (down to milliseconds or less) with the consequent need for fast algorithms for small-to-medium scale optimal control problems. Due to the cost and safety requirements, these algorithms need to be simple to code and verify, have limited memory footprint, and be able to solve problems with tens to hundreds variables within milliseconds or less in low complexity processors.

In this paper we consider MPC with linear prediction models subject to polyhedral constraints and with performance objectives formulated by convex quadratic functions. As a consequence, the MPC finite-time optimal control problem can be formulated as a constrained convex quadratic program (QP) for which convergence to the global optimum is guaranteed [9]. Several classes of algorithms are available for solving QPs, depending on the problem size, sparsity and computational and memory requirements of the target platform. Active set methods [10] and interior points methods [9], are among the most powerful and customizable methods for solving constrained quadratic programs. Their high performance is obtained by solving linear systems exploiting complex routines of linear algebra such as QR and Cholesky factorization [10], and by keeping those factorizations in memory for future

*Corresponding author. Email: dicairano@ieee.org
†These authors had equal contributions in this paper.

use. Recently, MPC-tailored interior point solvers [11]–[13] and active set solvers [14] have been introduced, and they have been proven capable of achieving high performance by taking advantage of the MPC problem structure. Indeed both active sets and interior-point methods are capable of rapidly solving large problems. On the other hand, they have significant computational requirements due to the use of complex routines for linear algebra, in particular for solving systems of linear equations, that results in complex, and hence difficult to validate, code, and large program memory occupancy (e.g., hundreds of kilobytes).

An alternative approach proposed in [15] is based on the explicit solution of the MPC parametric quadratic program. By computing the parametric solution offline, the optimization algorithm is not executed at controller runtime. The explicit MPC controllers has been proved successful, especially in automotive applications [3], [16], [17]. However, the parametric solution complexity increases combinatorially as a function of the number of constraints. Thus, explicit MPC can be implemented even in minimal computational platforms, but is applicable only to problems with few constraints and hence with a short prediction horizon.

For the intermediate cases of medium-size problems that need to be solved in simple, yet not minimal, embedded control architectures, gradient-based iterative algorithms [10] are interesting candidates. Iterative algorithms are based on (many) repeated executions of few basic operations. While the number of iterations can be large, each iteration requires little time and operations that do not require complex subroutines. In [18], a fast-gradient algorithm [19]–[21] is proposed for MPC controllers where the constraints are expressed in terms of "simple sets". Such sets can be used to enforce, for instance, input bounds. For constraints that cannot be formulated by simple sets, [22] proposes an iterative procedure based on the dual of the problem where the system equations are relaxed. In [23] a fast gradient algorithm is proposed for the dual of the MPC quadratic program. For the algorithm in [23] there are no additional restrictions on the constraints that can be handled. A key property of the algorithms in [22], [23] is that, under mild assumptions, a bound on the number of iterations to converge to the solution can be computed, thus making the algorithm maximum number of operations predictable. An algorithm based on the fast gradient method combined with the Lagrange method of multipliers was proposed in [24] and accelerated gradient methods for problems with a structure, and in particular for distributed MPC, were proposed in [25], [26]. The iterative algorithms in [22]–[24] perform two steps at every iteration. First, they update the current solution by performing simple operations on the cost function gradient without accounting for the constraints, and then project the updated solution into the feasible set. Although for "simple" sets the projection operation may be computationally inexpensive, forcefully changing the updated candidate solution a-posteriori can significantly reduce the improvement achieved in the iteration. This is a well known problem that limits the performance of classical projected-gradient iterative methods [27], and which led to the development of fast gradient methods, where such problem is reduced, but it is still present.

In this paper[1] we propose a projection-free algorithm for constrained QP for application to linear MPC where the iteration update guarantees that the update solution maintains feasibility. Hence, a-posteri projections on the feasibile set, which in general reduce the improvement achieved in the iteration itself, are not needed. The algorithm, related to the one proposed in [30], was initially formulated in [31] for solving non-negative least squares problems arising in image processing, and it was named Parallel Quadratic Programming (PQP) because it can be efficiently parallelized. The name PQP can also be interpreted as Projection-free Quadratic Programming to highlight this additional feature of the algorithm. Here, we show how PQP can be compounded with an acceleration technique that performs a projection-free line search, which is particularly useful to reduce the number of iterations needed to reach the optimum. We show how to apply PQP to the MPC optimization problem by solving the dual of the QP formulated from the MPC finite-time optimal control problem. For this approach, termination conditions guaranteeing a requested degree of suboptimality can be enforced. We also show that for MPC problems several computations can be executed offline, hence avoiding dynamic memory allocation and large input/output dataflow.

The paper is organized as follows. In Section II we introduce the PQP algorithm for non-negative least squares problems, we prove the convergence of the algorithm, and we introduce an acceleration technique based on a projection-free line search. In Section III we formulate the linear MPC problem, the resulting convex quadratic program, and the notion of suboptimal solution. Then, we discuss the application of PQP to the MPC quadratic program and the termination conditions guaranteeing the desired degree of solution suboptimality. In Section IV

[1]Preliminary results related to this subject have been presented in [28] and [29]. This paper provides more details on the algorithms and the related convergence proofs and more detailed simulation studies and comparisons. Also, the algorithm implementations have been improved, thus leading to improved performance in the simulations.

we discuss how to efficiently apply PQP to MPC by exploiting the parametric primal and dual QP form of MPC to pre-compute and pre-store several matrices and to reduce calculations. In Section V we present simulation results on thee MPC benchmark problems and compare the PQP performance in terms of computation time with the ones of available open source and commercial QP solvers. Finally, in Section VI we summarize conclusions and future work.

Notation: $\mathbb{R}$, $\mathbb{R}_{0+}$, $\mathbb{R}_{+}$, $\mathbb{Z}$, $\mathbb{Z}_{0+}$, $\mathbb{Z}_{+}$ denote real, nonnegative real, positive real, integer, nonnegative integer, and positive integer numbers, respectively, and $\mathbb{Z}_{[a,b]} \triangleq \{z \in \mathbb{Z} : \ a \leq z \leq b\}$. For a vector ϕ, $[\phi]_i$ denotes the i^{th} component, for a matrix $\Phi \in \mathbb{R}^{n\times m}$, $[\Phi]_{ij}$ denotes the element at the i^{th} row and j^{th} column. We denote the vector entirely composed of ones of dimension m by $\mathbf{1}_m$, the identity matrix by I_m, and the matrix entirely composed of zeros by 0_m, where the subscripts are dropped when clear from the context. By $\circ$ we denote the Hadamard (element-wise) product between two vectors. Inequalities between vectors are intended componentwise, while for a symmetric matrix Q, $Q > 0$ ($Q \geq 0$) indicates positive definiteness (semidefiniteness). For a vector x and a matrix $Q \geq 0$ of appropriate dimension, $\|x\|_Q^2 = x'Qx$. Given a matrix $\Phi \in \mathbb{R}^{n\times m}$, we define $\Phi^+, \Phi^- \in \mathbb{R}^{n\times m}$ such that $[\Phi^+]_{ij} = \max(0, [\Phi]_{ij})$ $[\Phi^-]_{ij} = \max(0, -[\Phi]_{ij})$. Given the vector $v \in \mathbb{R}^n$, $\mathrm{diag}(v) \in \mathbb{R}^{n\times n}$ is the diagonal matrix having v on the diagonal. Given the optimization problem $\min_{z\in\mathcal{Z}} J(z)$, the optimum is J^* and the optimal solution[2] is z^*, i.e., $J^* = J(z^*)$. Given a continuous time signal, $a_c(t)$, $t \in \mathbb{R}_{0+}$ the discrete time (sampled) signal with period T_s, $a(k)$, $k \in \mathbb{Z}_{0+}$, is such that $a(k) = a_c(kT_s)$. The notation $a(i|k)$ denotes the value of a predicted i steps ahead based on data at time k.

II. Parallel Quadratic Programming for Non-negative Least Squares

For a regressor data matrix $A_{\rm ls} \in \mathbb{R}^{n_\ell \times n_z}$ and a response data vector $b_{\rm ls} \in \mathbb{R}^{n_\ell}$, the (linear) least squares (LS) problem seeks the value of the parameter vector $z \in \mathbb{R}^{n_z}$ that minimizes the 2-norm of the residual vector $r_e \in \mathbb{R}^{n_\ell}$, $r_e = A_{\rm ls}z - b_{\rm ls}$, i.e., that minimizes $\|r_e\| = \sqrt{\sum_{i=1}^{n_\ell}([A_{\rm ls}z]_i - [b_{\rm ls}]_i)^2}$. This is achieved by minimizing the squared norm of the residual vector

$$z^* = \arg\min_{z\in\mathbb{R}^{n_z}} \frac{1}{2}\|A_{\rm ls}z - b_{\rm ls}\|^2. \tag{1}$$

For the common case where $n_\ell \geq n_z$, the solution of (1) can be computed in closed form as $z^* = (A'A)^{-1}A'b$.

Several variants of the LS problem (1) exist [32]. A problem that appears in several applications in fields as different as image processing, control theory, and data mining is the non-negative least squares (NNLS) problem, where (1) is compounded with the constraints that the parameter vector z has to belong to the non-negative cone, i.e., $z \in \mathbb{R}^{n_z}_{0+}$, i.e.,

$$z^* = \arg\min_{z\in\mathbb{R}^{n_z}_{0+}} \frac{1}{2}\|A_{\rm ls}z - b_{\rm ls}\|^2. \tag{2}$$

Although very similar to (1), for problem (2) it is not simple to compute a closed-form solution, and hence it is solved by appropriate numerical algorithms. NNLS problem (2) can be formulated as

$$\min_{z} \quad J(z) = \frac{1}{2}z'Hz + F'z + M \tag{3a}$$
$$\text{s.t.} \quad z \geq 0, \tag{3b}$$

where $F \in \mathbb{R}^{n_z}$, $H \in \mathbb{R}^{n_z\times n_z}$, $H \geq 0$, and $\frac{1}{2}z'Hz + F'z + M = \frac{1}{2}\|A_{\rm ls}z - b_{\rm ls}\|^2$, where $M \geq 0$. Problem (3) is a convex quadratic program (QP) subject to linear constraints.

In order to compute the optimal solution of (3), first the Lagrangian of (3) is computed,

$$L(z,\lambda) = \frac{1}{2}z'Hz + F'z - \lambda'z. \tag{4}$$

Based on (4), the optimality conditions for (3) are obtained by variational calculus [33] as

$$z \circ \nabla_z L(z,\lambda) = 0 \tag{5a}$$
$$\lambda \circ \nabla_\lambda L(z,\lambda) = 0, \tag{5b}$$

[2]If there exists multiple $z \in \mathcal{Z}$ such that $J(z) = J^*$ they are all called optimal solutions.

where for (3), (5a) results in

$$\begin{aligned} z \circ \nabla_z L(z,\lambda) &= z \circ (Hz + F - \lambda) \\ &= z \circ ((H^+ z + F^+) - (H^- z + F^- + \lambda)). \end{aligned} \tag{6}$$

Assume momentarily that $z_i > 0$ (possibly infinitesimally) for all $i \in \mathbb{Z}_{[1,n_z]}$. As a consequence, $[\lambda]_i = 0$ for all $i \in \mathbb{Z}_{[1,n_z]}$, and (5) reduces to the fixed point

$$[z]_i = \frac{[H^- z + F^-]_i}{[H^+ z + F^+]_i}[z]_i, \quad \forall i \in \mathbb{Z}_{[1,n_z]} \tag{7}$$

Remark 1: Optimality conditions (5) are derived from variational calculus, and hence related to optimal control theory. They are well known to be equivalent, under mild assumptions, to the KKT optimality conditions for numerical optimization [9], see [33]. An alternative way to express (5) is

$$0 \leq z \perp \nabla J(z) \geq 0 \tag{8}$$

where $\perp$ denotes orthogonality. Such a notation is common in linear complementarity problems [34]. By (8) the fixed point (7) is obtained without the need for explicitly assuming $\lambda_i = 0$, $i \in \mathbb{Z}_{[1,n_z]}$.

In what follows we use (7) to obtain an iteration that moves each z_i in the direction of the anti-gradient without leaving the feasible region. In case $\lambda_i^* > 0$, and thus $z_i^* = 0$, the iteration can be shown to drive z_i to 0 without explicitly estimating λ_i. For a fixed matrix $\phi \in \mathbb{R}^{n_z \times n_z}$ chosen to guarantee convergence as explained later, the Parallel Quadratic Programming algorithm is derived as described in Algorithm 1.

Algorithm 1 Parallel Quadratic Programming (PQP)

1: set $h = 0$, $z_{(h)} = \bar{z} > 0$.
2: **repeat**
3: **for** $i = 1 : n_z$ **do**
4: compute update

$$[z_{(h+1)}]_i = \frac{[(H^- + \phi) z_{(h)} + F^-]_i}{[(H^+ + \phi) z_{(h)} + F^+]_i}[z_{(h)}]_i, \tag{9}$$

5: **end for**
6: $h = h + 1$
7: **until** (termination condition)

In Algorithm 1 the initial value in Line 1 is an arbitrary strictly feasible value, i.e., any point in the positive cone. The termination condition in Line 7 is discussed extensively later.

The PQP algorithm, originally proposed by [31], has some interesting properties. As for projected-gradient methods, the update rule is completely parallelizable, with communications overhead determined by the bandwidth of the Hessian H. Significant speed-ups can be obtained on GPUs, multicore CPUs, and Single Instructions Multiple Data (SIMD) architectures. In contrast to projected-gradient methods, the iteration itself maintains feasibility, hence there is no need for a-posteriori projections onto the feasible set which may significantly reduce the solution improvement obtained during the iteration. Finally, the convergence rate of the algorithm is linear.

Remark 2: For the PQP algorithm the dominant operation is a matrix-vector product, resulting in a complexity of $O(m_z \cdot p)$, where p is the number of desired bits of precision and $m_z \leq n_z^2$ is the number of nonzeros in H. For parallel implementations the time complexity reduces to $O(b_z \cdot p)$, where $b_z \leq n_z$ is the bandwidth of H. In comparison, interior-point methods typically have complexity $O(n_z^3 \cdot \log(p))$, the dominant operation being matrix inversion/division. Thus, the PQP algorithm is attractive for large scale sparse problems [31]. However, as it is shown in this paper, it has also advantage for the size of problems that are typical of MPC applications.

Remark 3: Each iteration of the PQP update has been proven to reduce error by at least a constant fraction, resulting in a linear rate of convergence [31] and superlinear convergence is often observed. While no formal proof is yet available, in numerous tests the convergence rate coefficient appears to be proportional to the square root of the Hessian condition number [29], similarly to fast-gradient methods [18].

A. PQP algorithm convergence

The convergence of Algorithm 1 to the optimum z^* of (3) can be guaranteed by a suitable choice of the matrix $\phi \in \mathbb{R}^{n_z \times n_z}$ in (9). Consider a series expansion of $J(z)$ around $z \in \mathbb{R}^{n_z}$,

$$J(\xi) = J(z) + (\xi - z)'\nabla J(z) + \frac{1}{2}(\xi - z)'H(\xi - z). \tag{10}$$

For $z > 0$, we define an auxiliary function by modifying the last term in (10),

$$\mathcal{G}(\xi, z) = J(z) + (\xi - z)'\nabla J(z) + \frac{1}{2}(\xi - z)'\mathcal{K}(z)(\xi - z), \tag{11}$$

where $\mathcal{K}(z) \in \mathbb{R}^{n_z \times n_z}$ is the diagonal matrix

$$\mathcal{K}(z) = \mathrm{diag}((H^+ + \phi)z + F^+)\mathrm{diag}(z)^{-1}.$$

Clearly, $\mathcal{G}(z, z) = J(z)$. Next we prove that by properly choosing ϕ:

1) $\mathcal{G}$ upper-bounds J, i.e, for all $\xi \geq 0, z > 0$, $\mathcal{G}(\xi, z) \geq J(\xi)$;
2) the multiplicative update (9) yields $z_{(h+1)} = \arg\min_\xi \mathcal{G}(\xi, z_{(h)})$;
3) for any given $z_{(0)} > 0$, the sequence $\{z_{(h)}\}_h$ obtained from (9) is such that $J(z_{(h+1)}) < J(z_{(h)})$, for all $h \in \mathbb{R}_{0+}$;
4) the sequence $\{z_{(h)}\}_h$ converges to the optimum of (3), i.e., $\lim_{h\to\infty} z_{(h)} = z^*$, and $z^* \geq 0$.

We first prove some technical lemmas.

Lemma 1: Let $P \in \mathbb{R}_{0+}^{n_z \times n_z}$ be a nonnegative real symmetric matrix with at least one nonzero element in each row, $\xi \in \mathbb{R}_+^{n_z}$. Let $D \in \mathbb{R}^{n_z \times n_z}$ be a diagonal matrix with elements

$$[D]_{ii} = \frac{[P\xi]_i}{[\xi]_i}, \quad \text{for} \quad i = 1, 2, \cdots, n_z \ . \tag{12}$$

Then $(D - P) \geq 0$ and $(D + P) > 0$.

Proof: Consider the matrices $M_1, M_2 \in \mathbb{R}^{n_z \times n_z}$,

$$M_1 = \mathrm{diag}(\xi)(D + P)\mathrm{diag}(\xi), \quad M_2 = \mathrm{diag}(\xi)(D - P)\mathrm{diag}(\xi).$$

Since $\xi > 0$, $\mathrm{diag}(\xi)$ is invertible. Hence, $D + P$ and $D - P$ are congruent with M_1 and M_2, respectively, and $(D + P) > 0$, $(D - P) \geq 0$ if and only if $M_1 > 0$ ($M_2 \geq 0$) [35]. For any $\zeta \in \mathbb{R}_z^n$, $\zeta \neq 0$,

$$\begin{aligned}
\zeta' M_1 \zeta &= \sum_{i,j=1}^{n_z} ([D]_{ij} + [P]_{ij})[\xi]_i[\xi]_j[\zeta]_i[\zeta]_j = \sum_{i,j=1}^{n_z} [D]_{ij}[\xi]_i[\xi]_j[\zeta]_i[\zeta]_j + \sum_{i,j=1}^{n_z} [P]_{ij}[\xi]_i[\xi]_j[\zeta]_i[\zeta]_j \\
&= \sum_{i,j=1}^{n_z} [P]_{ij}[\xi]_i[\xi]_j[\zeta]_i^2 + \sum_{i,j=1}^{n_z} [P]_{ij}[\xi]_i[\xi]_j[\zeta]_i[\zeta]_j = \frac{1}{2}\sum_{i,j=1}^{n_z} [P]_{ij}[\xi]_i[\xi]_j([\zeta]_i + [\zeta]_j)^2 > 0
\end{aligned}$$

Thus, $D + P$ is positive definite. Similarly, $\zeta' M_2 \zeta = \frac{1}{2}\sum_{i,j=1}^{n_z} [P]_{ij}[\xi]_i[\xi]_j([\zeta]_i - [\zeta]_j)^2 \geq 0$. Therefore, $D - P$ is positive semidefinite. ∎

Lemma 2: For some non-negative matrix $\phi \in \mathbb{R}_{0+}^{n_z \times n_z}$ that depends only on H, $\mathcal{G}(\xi, z)$ upper-bounds $J(\xi)$, i.e., for all $\xi \geq 0, z > 0$, $J(\xi) \leq \mathcal{G}(\xi, z)$.

Proof: We prove that $\mathcal{K}(z) - H \geq 0$ from which it follows that

$$\mathcal{G}(\xi, z) - J(\xi) = (\xi - z)'(\mathcal{K}(z) - H)(\xi - z) \geq 0, \quad \forall \xi \geq 0, \ \forall z > 0.$$

Define $\bar{H}^+ \triangleq H^+ + \phi$, $\bar{H}^- \triangleq H^+ - H$. We split $\mathcal{K}(z) - H$ into the sum of two matrices,

$$\begin{aligned}
\mathcal{K}(z) - H &= \mathrm{diag}(\bar{H}^+ z + F^+)\mathrm{diag}(z)^{-1} - (\bar{H}^+ - \bar{H}^-) \\
&= (\mathrm{diag}(\bar{H}^+ z)\mathrm{diag}(z)^{-1} - \bar{H}^+) + (\mathrm{diag}(F^+)\mathrm{diag}(z)^{-1} + \bar{H}^-) \\
&= \mathcal{K}_{psd}(z) + \mathcal{K}_{nn}(z)
\end{aligned}$$

where $\mathcal{K}_{psd}(z) = \mathrm{diag}(\bar{H}^+ z)\mathrm{diag}(z)^{-1} - \bar{H}^+$ and $\mathcal{K}_{nn}(z) = \phi + \mathrm{diag}(F^+)\mathrm{diag}(z)^{-1} + H^-$. From Lemma 1 we have that $\mathcal{K}_{psd}(z) \geq 0$ for all $z > 0$. Then, we can choose ϕ so that $\mathcal{K}_{nn}(z) \geq 0$, for all $z > 0$. For example,

choose ϕ to be a nonnegative diagonal matrix, i.e., $[\phi]_{ii} \geq 0$, and $[\phi]_{ij} = 0$ if $i \neq j$, where $[\phi]_{ii} \geq [H^- \cdot \mathbf{1}]_i$ for all $i = \mathbb{Z}_{[1,n_z]}$. Then, $\bar{H}^-$ is diagonally dominant and thus positive semidefinite, and hence also $\mathcal{K}_{nn}$ and $\mathcal{K}(z) - H$ are positive semidefinite. ■

In this paper we use the above definition for ϕ, i.e., $[\phi]_{ij} = 0$ if $i \neq j$, $[\phi]_{ii} \geq [H^- \cdot \mathbf{1}]_i$ for all $i, j = \mathbb{Z}_{[1,n_z]}$. However, all the proofs reported next do not depend on this specific choice.

Remark 4: For the considered choice of ϕ, larger values of $[\phi]_{ii}$ reduce the speed of convergence of Algorithm 1 by moving the gain in the multiplicative update (9) towards 1. In many useful cases it can be shown that convergence can be obtained for small values of $[\phi]_{ii}$, $i = \mathbb{Z}_{[1,n_z]}$. In several cases, for instance when H^- is already diagonally dominant, it is possible to set $[\phi]_{ii} = 0$ for all $i = \mathbb{Z}_{[1,n_z]}$.

Lemma 3: Given any $z > 0$, $\xi \in \mathbb{R}^{n_z}$ obtained by

$$[\xi]_i = \frac{[(H^- + \phi)z_{(h)} + F^-]_i}{[(H^+ + \phi)z_{(h)} + F^+]_i}[z_{(h)}]_i, \;\; i \in \mathbb{Z}_{[1,n_z]}, \tag{13}$$

is such that $\xi > 0$, and it yields the minimum of $\mathcal{G}(\xi, z)$.

Proof: For any fixed $z > 0$, $\mathcal{G}(\xi, z)$ is quadratic and positive semidefinite in ξ, hence it has global minimum for ξ such that

$$\nabla_\xi \mathcal{G}(\xi, z) = \nabla J(z) + \mathcal{K}(z)(\xi - z) = 0. \tag{14}$$

Solving (14) for ξ we recover (13) in matrix form,

$$\begin{aligned}
\xi &= z - \mathcal{K}(z)^{-1}\nabla J(z) = z - \mathcal{K}(z)^{-1}(Hz + F) \\
&= z - \mathcal{K}(z)^{-1}((H^+ + \phi)z + F^+) + \mathcal{K}(z)^{-1}((H^- + \phi)z + F^-) \\
&= z - z + \mathcal{K}(z)^{-1}((H^- + \phi)z + F^-) \\
&= \mathrm{diag}(z)\mathrm{diag}((H^+ + \phi)z + F^+)^{-1}((H^- + \phi)z + F^-).
\end{aligned}$$

Since $z > 0$ and the matrices in (13) are all nonnegative, it is straightforward that $\xi \geq 0$. ■

Lemma 4: Given any $z_{(h)} \neq z^*$, if there exists $i \in \mathbb{Z}_{[1,n_z]}$ such that $[z]_{i(h)}((H^+ + \phi)z_{(h)} + F^+)_i \neq 0$, the objective function decreases with iteration (9), i.e.,

$$J(z_{(h+1)}) \leq \mathcal{G}(z_{(h+1)}, z_{(h)}) < \mathcal{G}(z_{(h)}, z_{(h)}) = J(z_{(h)}). \tag{15}$$

Proof: The first inequality in (15) follows from Lemma 2. Next, we prove the second inequality. $\mathcal{G}(z_{(h+1)}, z_{(h)})$ is convex in $z_{(h+1)}$ by construction, and strictly convex with respect to $[z_{(h+1)}]_i$, $i \in \mathbb{Z}_{[1,n_z]}$, such that $[z_{(h)}]_i[(H^+ z_{(h)} + F^+]_i \neq 0$, because

$$\frac{\partial^2}{\partial [z_{(h+1)}]_i^2}\mathcal{G}(z_{(h+1)}, z_{(h)}) = [\mathcal{K}(z_{(h)})]_{ii} = \frac{[(H^+ + \phi)z_{(h)} + F^+]_i}{[z_{(h)}]_i} > 0 \; .$$

Since $[\mathcal{K}(z_{(h)})]_{ii} > 0$ and $z_{(h)} \neq z^*$, $[z_{(h+1)}]_i = [\arg\min_\xi \mathcal{G}(\xi, z_{(h)})]_i \neq [z_{(h)}]_i$, because

$$[z_{(h+1)}]_i - [z_{(h)}]_i = -[\mathcal{K}(z_{(h)})^{-1}\nabla J(z_{(h)})]_i \neq 0. \tag{16}$$

Thus, $\xi = z_{(h)}$ is not a minimizer of $G(\xi, z_{(h)})$, and $\mathcal{G}(z_{(h+1)}, z_{(h)}) < \mathcal{G}(z_{(h)}, z_{(h)})$. ■

Theorem 1: Let $\phi \in \mathbb{R}_{0+}^{n_z \times n_z}$ be chosen to satisfy the assumptions of Lemma 2, and such that $[H + \phi]_{ii} > 0$, for all $i \in \mathbb{Z}_{[1,n_z]}$. Given any positive $z^0 > 0$ the update (9) generates a sequence $\{z_{(h)}\}_h$ such that the sequence $\{J(z_{(h)})\}_h$ converges to the optimum, i.e., $\lim_{h\to\infty} J(z_{(h)}) = J^*$, and

$$\lim_{h\to\infty} z_{(h)} = z^*. \tag{17}$$

Proof: Since $[H + \phi]_{ii} > 0$, for all $i \in \mathbb{Z}_{[1,n_z]}$, the assumptions of Lemma 4 are satisfied and monotonic decrease of $\{J(z_{(h)})\}_h$ is guaranteed. From (16) we know that z is a stationary point if and only if

$$z \circ \nabla_z J(z) = 0.$$

According to (8), this is also the KKT optimality condition for (3) when $\nabla(z) \geq 0$ and $z \geq 0$. Next we show that the update has no other fixpoints in the positive cone. Assuming $z > 0$ (possibly infinitesimally), a fixpoint requires

$\nabla J(z) = 0$. Due to convexity of $J(z)$, $\nabla J(z) = 0$ only for $z = z^*$. Consequently any fixpoint of the update (9) is a solution of (3). Since $J(z)$ is lower bounded, the decreasing sequence $\{J(z_{(h)})\}_h$ converges to the lower bound J^*. Since the update is stationary only at z^* where $J(z^*) = J^*$, $z^* = \lim_{h\to\infty} z_{(h)}$. Thus, $\{J(z_{(h)})\}_h$ converges to the minimum $J(z^*)$ as $\{z_{(h)}\}_h$ approaches the limit z^*. ■

Remark 5: Although all iterates remain in the positive cone, elements of z that correspond to active constraints (at optimum) in (3) are seen to rapidly (albeit asymptotically) decay to zero. These can be thresholded to zero and the corresponding rows and columns of H, F, z can be removed from subsequent iterations of the update. If the QP is only weakly convex and therefore does not have a unique global optimum, the sequence $z_(h)_h$ will converge to an optimal point determined by ϕ.

B. PQP acceleration by line search

While classical optimization algorithms usually compute the solution update by performing two steps, a descent direction selection, and a step size selection (also called line search), Algorithm 1 performs the two actions at once by (9). In fact, PQP may be interpreted as a dynamically scaled gradient method. In order to see this, consider the PQP update (9), where for simplicity $\phi = 0$, and sum and subtract from each component the denominator to obtain

$$[z_{(h+1)}]_i = [z_{(h)}]_i - \frac{[z_{(h)}]_i}{[H^+ z_{(h)} + F^+]_i}[Hz_{(h)} + F]_i,$$

and hence

$$z_{(h+1)} = z_{(h)} - T(z_{(h)})\nabla_z J(z).$$

The matrix $T(z_{(h)})$ is diagonal matrix where

$$[T(z_{(h)})]_{ii} = \frac{[z_{(h)}]_i}{[H^+ z_{(h)} + F^+]_i}, \ \forall i \in \mathbb{Z}_{[1,n_z]},$$

thus it scales the gradient, or alternatively it preconditions the Hessian [36], such that the solution remains feasible. The gradient scaling view reveals a weakness of the PQP fixpoint: progress toward the optimum can be slowed if the numerator $[z_{(h)}]_i$ is very close to zero but the optimal value of this variable $[z^*]_i$ is not. This can be remedied by moving all such variables away from zero. The following lemma provides a locally optimal move that can be computed in the same number of flops as the PQP iteration.

Lemma 5: Let the $z_{(h)} \in \mathbb{R}^{n_z}$ be a feasible solution for NNLS (3). The vector $z_{(h+1)} \in \mathbb{R}^{n_z}$ obtained by

$$\alpha(z_{(h)}) = \begin{cases} -\frac{\nabla_z J(z_{(h)})' p_h}{p_h' H p_h} & \text{if} \quad p_h' H p_h > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{18a}$$

$$z_{(h+1)} = z_{(h)} + \alpha(z_{(h)}) p_h, \tag{18b}$$

where

$$p_h = (\nabla_z J(z_{(h)}))^-, \tag{19}$$

is feasible ($z_{(h+1)} \in \mathbb{R}^{n_z}_{0+}$) and $J(z_{(h+1)}) \le J(z_{(h)})$.

Proof: The decrease direction p_h is the component of the (negative) gradient[3] that points inside the feasible cone. Thus, given $z_{(h)} \ge 0$, for any $\alpha \ge 0$, $z_{(h+1)} = z_{(h)} + \alpha p_h$ is such that $z_{(h+1)} \in \mathbb{R}^{n_z}_{0+}$, and hence it is a feasible solution of (3). For the quadratic function $J(z)$, (18a) selects the optimal step length [10] for direction (19). □

The update iteration in Lemma 5 selects the decrease direction in the subspace spanned by the non-negative components of the anti-gradient. Thus, for any step length the value of each variable cannot decrease, and, since it was nonnegative at step h, it is nonnegative at step $h+1$, hence guaranteeing recursive feasibility. In general, $J(z_{(h+1)}) < J(_{(h)})$ whenever $p_h \neq 0$ and $\alpha(z_{(h)}) \neq 0$. When $\alpha(z_{(h)}) = 0$, (18) does not have any effect.

By using (18) in combination with Algorithm 1, the accelerated PQP algorithm is obtained, as described in Algorithm 2. The following theorem guarantees convergence of the accelerated PQP algorithm.

[3]This is obtained by applying to the gradient the operator $(\cdot)^-$, which changes the sign to the negative components and set the positive components to 0.

Algorithm 2 Line Search-accelerated PQP

1: set $h=0$, $z_{(h)}=\bar{z}>0$.
2: **repeat**
3: **if** LS condition **then**
4: compute decrease direction and step length

$$\begin{aligned} p_h &= (\nabla_z J(z_{(h)}))^- \\ \alpha(z_{(h)}) &= \begin{cases} -\frac{\nabla_z J(z_{(h)})' p_h}{p_h' H p_h} & \text{if} \quad p_h' H p_h > 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

5: compute update

$$z_{(h+1)} = z_{(h)} + \alpha(z_{(h)}) p_h$$

6: **else**
7: **for** $i=1:n_z$ **do**
8: compute update

$$[z_{(h+1)}]_i = \frac{[(H^- + \phi) z_{(h)} + F^-]_i}{[(H^+ + \phi) z^{(h)} + F^+]_i} [z_{(h)}]_i$$

9: **end for**
10: **end if**
11: $h = h+1$
12: **until** (termination condition)

Theorem 2: Let $z_{(h)}$ for $h=0$ be a feasible solution for the NNLS (3), and let ϕ be chosen such that Theorem 1 holds. Then, Algorithm 2 where the acceleration activation policy in Line 9 is any policy such that at least one PQP iteration is executed between any two acceleration iterations converges asymptotically to the optimum z^*.

Proof: The proof is obtained in a similar manner to the stability proofs of switched systems, while noting that z^* is an equilibrium for (9) and for (18) and that $\mathcal{V}(z) = J(z) - J^* \geq 0$, with equality only at (an) optimal solution, thus making $\mathcal{V}$ similar to a Lyapunov function [37].

Algorithm 2 switches between PQP iteration and projection-free acceleration. Consider the sequence of iteration indices $\{h\}_{h\in\mathbb{Z}_{0+}}$, and the subsequence of indices where the acceleration is performed, $\{h_\varsigma\}_{\varsigma\in\mathbb{Z}_{0+}}$. Let $\|z_{(h)} - z^*\| > 0$, possibly infinitesimally. Due to Theorem 1, for all $\varsigma \in \mathbb{Z}_{0+}$, $J(h+1) < J(h)$, for all $h \in \mathbb{Z}_{[h_\varsigma+1, h_{\varsigma+1}]}$. Also, due to the properties of the projection-free acceleration $J(h_\varsigma+1) \leq J(h_\varsigma)$, for all $\varsigma \in \mathbb{Z}_{0+}$. Due to the properties guaranteed by the linear search activation condition (Line 3), for all $\varsigma \in \mathbb{Z}_{0+}$ we have $J(h_{\varsigma+1}) < J(h_\varsigma)$, and $J(h_{\varsigma+1}+1) < J(h_{\varsigma+1})$. Thus, whenever entering any of the two modes, the value from the previous time that mode was entered has decreased. Convergence of $J(z)$ to J^* (i.e., to $\mathcal{V}(z)=0$) independently of the line search activation policy used is thus straightforwardly obtained as in [38] for the stability of switched systems. □

Theorem 2 guarantees convergence of Algorithm 2, for many choices of activation of line search (18). A simple yet effective strategy is to perform a line search iteration every $n_{\text{ls}} \in \mathbb{Z}_{(1,\infty)}$ PQP iterations. If the problem is only weekly convex, the line search iteration should not be performed excessively often, because it may be expensive to compute compared to the resulting speedup. In experimental tests $n_{\text{ls}} \in \mathbb{Z}_{[10,50]}$ seems to provide the best results. An alternative approach is based on comparing consecutive gradient directions while guaranteeing the condition in Theorem 2. If these directions are almost aligned, the line search should be executed, since it may increase the step size with respect to what the PQP iteration would provide.

III. Constrained linear MPC

Model predictive control [1] is an advanced technique for optimal control of constrained dynamical systems that has found several applications in different domain from process control [2] to automotive [3], [17], from aerospace [5], [6] to mechatronic systems [39], [40]. MPC selects the control input by computing the optimal control sequence along a finite future horizon for the predicted system dynamics with respect to a user-defined performance (cost) function and subject to constraints on system state, input, and output. Different classes of model

predictive control have been developed, differentiated by the properties of the system model, cost function and constraints. In this paper we focus on linear-quadratic model predictive control for regulation and tracking, also known simply as linear model predictive control.

Linear model predictive control is based on the linear prediction model

$$\begin{aligned} x(k+1) &= Ax(k)+Bu(k) && \text{(21a)}\\ y(k) &= Cx(k)+Du(k), && \text{(21b)}\end{aligned}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$ are the state, input, and output vectors subject to constraints

$$\begin{aligned} x_{\min} \le x(k) \le x_{\max}\ , && \text{(22a)}\\ u_{\min} \le u(k) \le u_{\max}\ , && \text{(22b)}\\ y_{\min} \le y(k) \le y_{\max}\ , && \text{(22c)}\end{aligned}$$

where $x_{\min}, x_{\max} \in \mathbb{R}^n$, $u_{\min}, u_{\max} \in \mathbb{R}^m$, and $y_{\min}, y_{\max} \in \mathbb{R}^p$ are the lower and upper bounds on the state, input, and output vectors, respectively. At every control cycle $k \in \mathbb{Z}_{0+}$, model predictive control solves the finite horizon optimal control problem

$$\begin{aligned} \min_{\bar{U}(k)} \quad & \|x(N|k)\|^2_{P_{\rm M}} + \sum_{i=0}^{N-1} \|x(i|k)\|^2_{Q_{\rm M}} + \|u(i|k)\|^2_{R_{\rm M}} && \text{(23a)}\\ \text{s.t.} \quad & x(i+1|k) = Ax(i|k)+Bu(i|k) && \text{(23b)}\\ & y(i|k) = Cx(i|k)+Du(i|k) && \text{(23c)}\\ & x_{\min} \le x(i|k) \le x_{\max},\ i \in \mathbb{Z}_{[1,N_c]} && \text{(23d)}\\ & u_{\min} \le u(i|k) \le u_{\max},\ i \in \mathbb{Z}_{[0,N_{cu}-1]} && \text{(23e)}\\ & y_{\min} \le y(i|k) \le y_{\max},\ i \in \mathbb{Z}_{[1,N_c]} && \text{(23f)}\\ & u(i|k) = K_f x(i|k),\ i \in \mathbb{Z}_{[N_u,N-1]} && \text{(23g)}\\ & x(0|k) = x(k), && \text{(23h)}\end{aligned}$$

where $Q_{\rm M} \ge 0$, $P_{\rm M}, R_{\rm M} > 0$ are symmetric weight matrices of appropriate dimensions, N is the prediction horizon, $N_u \le N$ is the control horizon (the number of free control moves), $N_{cu} \le N$, $N_c \le N$ are the input and output constraint horizons along which constraints are enforced, and $\bar{U}(k) = [u'(0|k) \dots u'(N-1|k)]' \in \mathbb{R}^{Nm}$ is the vector to be optimized. The performance criterion is defined by (23a), and (23d)–(23f) enforce the constraints. Equation (23g) defines the pre-assigned terminal controller where $K_f \in \mathbb{R}^{m\times n}$, so that the optimization vector effectively is $U(k) = [u'(0|k) \dots u'(N_u-1|k)]' \in \mathbb{R}^{N_u m}$.

Remark 6: Although the optimal control problem (23) does not explicitly mention a reference, tracking is achieved by including in the state update equation (21a) the reference prediction dynamics

$$r_r(k+1) = A_r r_r(k),$$

and an additional output in (21b) representing the tracking error

$$y_e(k) = Cx(k) - C_r r_r(k),$$

which is then accounted for in the cost function (23a) as shown later in the case studies, see also [6], [17] for practical cases.

At time k, the MPC problem (23) is initialized with the current state value $x(k)$ by (23h) and solved to obtain the optimal sequence $\bar{U}^*(k)$. Then, the input $u(k) = u_{\rm MPC}(k) = u^*(0|k) = [I_m\ 0\ \dots\ 0]\bar{U}(k)$ is applied to the system.

Given the current state $x(k)$, problem (23) can be formulated as the quadratic program

$$\begin{aligned} \min_{U} \quad & J_p(U) = \frac{1}{2}U'Q_pU + F_p'U + \frac{1}{2}M_p && \text{(24a)}\\ \text{s.t.} \quad & G_pU \le K_p, && \text{(24b)}\end{aligned}$$

where $U = U(k)$, $Q_p \in \mathbb{R}^{n_u \times n_u}$, $n_u = N_u m$, $Q_p > 0$, and $G_p \in \mathbb{R}^{n_q \times n_u}$, $F_p \in \mathbb{R}^{n_u}$, and M_p are computed as explained, for instance, in [41].

For numerical algorithms, termination conditions yielding appropriate approximations of the solution of (24) have to be defined.

Definition 1: Consider problem (24). Given the non-negative 4-tuple $\varepsilon \in \mathbb{R}^4_{0+}$, $\varepsilon = (\varepsilon^r_J \ \varepsilon^a_J \ \varepsilon^r_c \ \varepsilon^a_c)$, we call an ε-solution for problem (24) a vector $\tilde{U}$ such that

$$G_p \tilde{U} \leq K_p + \max\{\varepsilon^r_c |K_p|, \varepsilon^a_c\} \tag{25a}$$

$$J(\tilde{U}) - J(U^*) \leq \max\{\varepsilon^r_J |J(U^*)|, \varepsilon^a_J\} \ . \tag{25b}$$

According to Definition 1 an ε-solution is a vector $\tilde{U}$ such that the constraint violation and the duality gap are ε-bounded in either relative $(\varepsilon^r_c, \varepsilon^r_J)$ or absolute $(\varepsilon^a_c, \varepsilon^a_J)$ errors

A. PQP-based solution of MPC quadratic programs

The NNLS problem (3) is a subclass of the general (convex) QP (24) that needs to be solved to compute the MPC command. While Algorithm 1 cannot be directly applied to (24), it can still be exploited to obtain a solution of (24) through duality [9]. The dual problem of (24) is

$$\min_z \quad J_d(Y) = \frac{1}{2} Y' Q_d Y + F_d' Y + \frac{1}{2} M_d \tag{26a}$$

$$\text{s.t.} \quad Y \geq 0 \ , \tag{26b}$$

where $Q_d = G_p Q_p^{-1} G_p'$, $F_d = (K_p + G_p Q_p^{-1} F_p)$ and $Y \in \mathbb{R}^{n_q}$, i.e., the number of variables in (26) is equal to the number of constraints in (24). In (26), $M_d = F_p' Q_p^{-1} F_p - M_p$ does not affect the optimal solution, but it affects the value of the optimum, and it is included for the subsequent discussion. Let Y^* be the (bounded) optimal solution of (26). Then, the optimal solution of the primal QP (26) is

$$U^* = \psi_{d2p}(Y^*) = -Q_p^{-1}(F_p + G_p' Y^*). \tag{27}$$

The approach for solving the MPC finite-time optimal control problem through Algorithm 1 consists of the following steps. At step k, given the current state $x(k)$:

(i) formulate (24);
(ii) construct (26);
(iii) solve (26) by Algorithm 2;
(iv) compute the solution of (24) by (27);
(v) apply the MPC command $u(k) = u_{\mathrm{MPC}}(k)$.

Solving the QP problem (24) via its dual (26) has the drawback that if in (24) there are more constraints than variables ($n_z < n_y$), (26) has more variables than (24), and $Q_d \geq 0$ (while $Q_p > 0$). On the other hand, solving the dual allows us to enforce termination conditions guaranteeing an ε-solution according to Definition 1 by using the duality gap.

Let $Y_{(h)}$, $h \in \mathbb{Z}_{0+}$ be a candidate solution of (26), a "candidate" primal solution $U_{(h)}$ is found by (27). Assume $U_{(h)}$ and $Y_{(h)}$ are primal and dual feasible, respectively, and let

$$J_p(U_{(h)}) + J_d(Y_{(h)}) \leq \varepsilon^a_J. \tag{28}$$

By duality, $-J_d(Y_{(h)}) \leq J_p(U_{(h)})$, where equality holds at optimum if strong duality holds, e.g. by Slater's conditions, see [9]. Indeed,

$$-J_d(Y_{(h)}) \leq -J_d(Y^*) \leq J_p(U^*) \leq J_p(U_{(h)}), \tag{29}$$

and hence (28) implies $J_p(U_{(h)}) - J_p(U^*) \leq \varepsilon^a_J$.

Similarly, if the condition

$$\frac{J_p(U_{(h)}) + J_d(Y_{(h)})}{-J_d(Y_{(h)})} \qquad \text{if} \qquad -J_d(Y_{(h)}) > 0, \tag{30a}$$

$$\frac{J_p(U_{(h)}) + J_d(Y_{(h)})}{-J_p(Y_{(h)})} \qquad \text{if} \qquad J_p(U_{(h)}) < 0, \tag{30b}$$

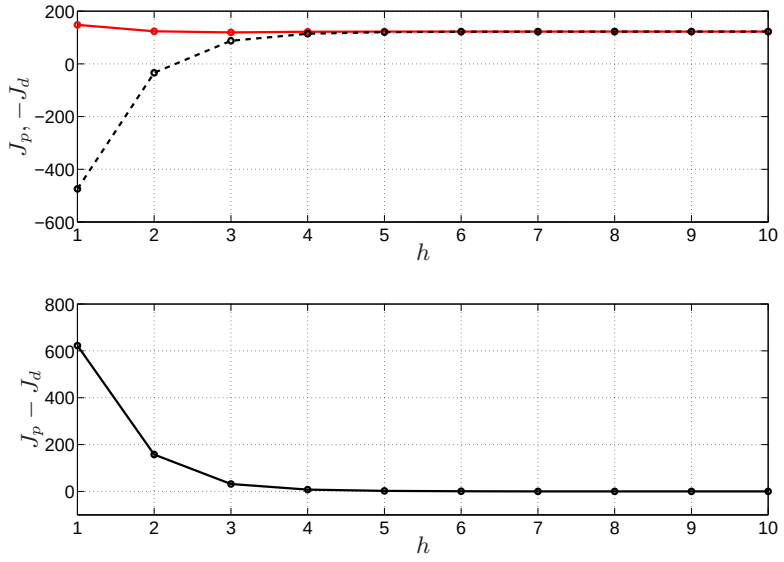

Fig. 1: Cost value throughout the iterations of Algorithm 2. Upper plot: Primal cost (solid), and dual cost (dash). Lower plot: Duality gap.

holds, (29) guarantees that $J_p(U_{(h)}) - J_p(U^*) \leq \varepsilon_J^r |J_p(U^*)|$.

Remark 7: Condition (30) does not account for the case where $J_p(U_{(h)}) > 0$ and $J_d(Y_{(h)}) < 0$, which may occur. However, for this to occur close to the optimum, $J_p(U^*) \approx 0$, and hence, due to the $\max$ operator in (25b), the termination condition to be considered is the absolute error.

An example of the primal and dual cost throughout the iterations of Algorithm 1 is shown in Figure 1. Termination conditions (28), (30) are valid only for solutions that are primal and dual feasible. While dual feasibility is guaranteed by the PQP properties, primal feasibility of $U_{(h)}$ has to be verified before checking (28), (30). By checking feasibility according to (25) an error is induced in (25b), because only ε-feasibility is verified. However, for reasonably small values of ε_c^a, ε_c^r such error will be small, and it can be also accounted for in conditions (28), (30). Alternatively, ε_c^a, ε_c^r can be set to 0.

IV. PQPMPC Controller Design

As described in Section III-A, the application of PQP to the MPC problem (23) requires at every step the formulation of the dual QP, its solution, and the calculation of the primal solution. The first step may be computationally demanding due to the need to generate and perform I/O operations on possibly large matrices (hundreds of bytes).

However, the special structure of the QP problem formulated from MPC can be exploited to perform part of the calculations offline. From (23), the MPC controller is a static (nonlinear) state feedback. Thus, a control algorithm implementing MPC needs only the state to generate the control input. In fact, the MPC problem (23) where the current state $x(k) = x$ is considered a parameter, can be written in parametric form [15], [42] as

$$\min_{U} \quad \frac{1}{2}U'Q_pU + x'C_p'U + \frac{1}{2}x'\Omega_p x \tag{31a}$$

$$\text{s.t.} \quad G_pU \leq S_px + W_p \ . \tag{31b}$$

The dual problem of (31) is the parametric QP

$$\min_{U} \quad \frac{1}{2}Y'Q_dY + x'S_d'Y + W_d'Y + \frac{1}{2}x'\Omega_d x \tag{32a}$$

$$\text{s.t.} \quad Y \geq 0 \ , \tag{32b}$$

where $Q_d = G_pQ_P^{-1}G_p'$, $S_d = (G_pQ_p^{-1}C_p + S_p)$, $W_d = W_p$, $\Omega_d = C_p'Q_p^{-1}C_p - \Omega_p$.

Furthermore, the primal optimal solution can be computed from the dual optimal solution via

$$U(Y^*) = \Psi_{d2p}(x, Y^*) = \Gamma_d x + \Xi_d Y^*, \tag{33}$$

where $\Xi_d = -Q_p^{-1}G_p'$, $\Gamma_d = -Q_p^{-1}C_p$.

Thus, the matrices of the dual parametric problem (32) and the parametric expression (33) can be computed beforehand. Given x, the dual problem (26) is instantiated by substituting x into (32). In order to account for the termination conditions, by substituting (33) into (31b), a vector $Y \geq 0$ results in a primal feasible solution when

$$-S_d x - W_d - Q_d Y \leq 0. \tag{34}$$

Thus, (25a) can be expressed as

$$-S_d x - W_d - Q_d Y \leq \max\{\varepsilon_c^r(|S_p x + W_d|), \varepsilon_c^a \mathbf{1}\}. \tag{35}$$

Similarly, by substituting (33) into (31a), the primal solution cost is

$$J_p(\Psi_{d2p}(x, Y)) = \frac{1}{2} Y' Q_d Y - \frac{1}{2} x' \Omega_d x. \tag{36}$$

Since $J_p(U) \geq 0$ due to the MPC cost (23a), the termination condition (25b) is expressed as

$$Y' Q_d Y + (x' S_d + W_d')Y \leq \max\left\{\frac{\varepsilon_J^r}{2}(Y' Q_d Y + 2(x' S_d' + W_d')Y + x'\Omega_d x), \varepsilon_J^a\right\},$$

where obviously the maximizations on the right hand side is equal to ε_J^a, when $J_d(Y_{(h)}) < 0$. Thus, the relative duality gap is ignored when $J_d(Y_{(h)}) < 0$, because either $J_p(U_{(h)}) < 0$ and hence, due to (23a), $U_{(h)}$ cannot be feasible, or it is the case where $J_d(Y_{(h)}) < 0$, $J_p(U_{(h)}) > 0$ and the relative duality gap should not be used.

Based on the above considerations, an MPC controller based on Algorithm 2 is synthesized and executed as described in Algorithm 3.

Remark 8: Pre-computing and pre-storing in memory the matrices of the parametric primal and dual QP as done in Algorithm 3 is fundamental for allowing the execution of the algorithm in low complexity embedded controllers for at least three reasons. First, it reduces the number of computations during the controller execution, especially for the cases when the dual problem matrices are used. Second, it reduces significantly the amount of I/O operations (from several matrices to a single vector), which are slower than computations, and the amounts of operating system calls, which are also time consuming. Third, it avoids entirely dynamic memory allocation, which is recommended in real-time embedded controllers due to the non-determinism of the operating system calls and the consequent complexity in guaranteeing real-time task execution [43]. In fact, thanks to the simplicity of the operations in Algorithm 3, the reduced I/O operations, and the absence of dynamic memory allocation, an operating system is basically not needed to execute the controller proposed here.

V. Case studies

In this section we discuss the results in applying the Algorithm 2 and 3 to three benchmark case studies: stabilization of a double integrator, control of pitch and angle of attack of a jet aircraft, and position control of a DC-motor. The case studies are available in the free toolbox [44]. All the case studies results in small-to-medium scale optimization problems, with less than 100 (primal) variables, and less than 400 constraints. While large scale problems may arise in process control applications of MPC, in applications with fast dynamics and low computational platform such values are reasonable. In fact, in practical applications the sampling period is chosen approximately one order of magnitude smaller than the system dominant time constant, and the MPC prediction horizon is chosen to cover the system response time, and hence approximately 4-25 sampling steps (i.e., 40%-250% the dominant time constant). Thus, for 1 to 4 control inputs, this results in up to 100 optimization variables. Some extremal cases where longer prediction horizons are needed may arise when the system modes have large time scale separation (due to the need of oversampling to ensure constraint satisfaction for the faster modes), which however is not common in the applications that are the focus of this paper, namely, automotive, aerospace, and factory automation [7].

A. QP solvers used in the case studies

In the case studies we simulate the system model in closed-loop with the model predictive controllers. The simulations are executed in a MacBook-Pro with Intel i7 (quad-core) processor 2.8GHz, and 8GB RAM, in Matlab

Algorithm 3 PQP-based Model Predictive Controller (PQPMPC)

OFFLINE:
Compute

$$Q_p, C_p, \Omega_p, G_p, W_p$$

Compute and store in memory

$$S_p, Q_d, C_d, \Omega_d, W_d, S_d,\ \Xi_d, \Gamma_d,\ Q_d^+ + \phi,\ Q_d^- + \phi$$

ONLINE:
$k = 0$
loop
  set $K_p = S_p x(k) + W_d$, $F_d = S_d x(k) + W_d$, $M_d = x(k)'\Omega_d x(k)$, $\gamma_d = \Gamma_d x(k)$
  initialize $h = 0$, $Y_{(h)} = \bar{Y} > 0$.
  repeat
    if LS condition **then**

$$\begin{aligned} p_h &= (\nabla_Y J_d(Y_{(h)}))^- \\ \alpha(Y_{(h)}) &= \begin{cases} -\frac{\nabla_Y J_d(Y_{(h)})' p_h}{p_h' Q_d p_h} & \text{if} \quad p_h' Q_d p_h > 0 \\ 0 & \text{otherwise} \end{cases} \\ Y_{(h+1)} &= Y_{(h)} + \alpha(Y_{(h)}) p_h \end{aligned}$$

    else
      for $i = 1 : n_d$ **do**

$$[Y_{(h+1)}]_i = \frac{[(Q_d^- + \phi)Y_{(h)} + F_d^-]_i}{[(Q_d^+ + \phi)Y_{(h)} + F_d^+]_i} [Y_{(h)}]_i,$$

      end for
    end if
    $h = h + 1$
  until (35) **and** (37)
  set $u(k) = \begin{bmatrix} I_m & 0 & \dots & 0 \end{bmatrix} (\gamma_d + \Xi_d Y)$
  $k = k + 1$
end loop

R2010b, executed in a single core of the processor by disabling multithreading. In order to assess the behavior of the algorithms introduced in this paper, in the simulations we use all the following QP solvers.

- PQP-M: Algorithm 2 for solving QP (24) implemented in Matlab M-code.
- PQPMEX: Algorithm 2 for solving QP (24) implemented in a C-coded mex function.
- PQPMPC: Algorithm 3 that synthesizes the MPC controller, implemented in a C-coded mex function.
- QPROG: `QUADPROG` routine of the Matlab Optimization Toolbox ver. 5.1 [45] implemented mainly in C-coded mex functions and partly in M-code. Due to the problem size in the case studies, which is consistent with standard size of the MPC optimization problems in mechatronics applications such as automotive, aerospace and factory automation, the medium scale algorithm is used, which is an active set method.
- DANTZ: Dantzig's active set algorithm [46] implemented by a C-coded mex function. This method has also been used in the Matlab MPC Toolbox [47] for several years[4].
- NAG: Inertia-controlling active set method using Cholesky-factorization in the commercial NAG toolbox for Matlab [48] implemented in a C-coded mex function.

[4]In recent versions of the toolbox, the QP algorithm is changed, mainly due to a different toolbox architecture.

- GPAD-M: Accelerated dual projected gradient method [23] implemented in M-code[5]. GPAD is based on the fast gradient method originally proposed in [19]. Similarly to PQP, and differently from [18], GPAD allows for any type of primal QP constraints, because it solves the dual problem. Thus, it appears the best candidate among the fast gradient methods to be compared with PQP. The termination conditions from [23] have been relaxed to include also the relative errors on cost function and constraints, according to Definition 1.
- QPOAS: `qpOASES` solver [14], which implements an active set method tailored to MPC problems.

Note that PQP-M and GPAD-M are implemented in M-code and hence significantly slower than the other solvers. However, including these results allows to assess the behavior of PQP with respect to a type of fast gradient algorithm. Also, from the relative performance of PQP with respect to GPAD, additional conclusions can be drawn as regards its behavior, by exploiting the comparison of C-coded implementation of GPAD-M with many other solvers reported in [49].

All the algorithms receive the data of the primal QP problem (24), and the inverse of the Hessian matrix, Q_p^{-1}. PQP, PQPMEX, PQPMPC, and GPAD construct and solve the dual problem, and compute the solution of the primal problem. When possible, the solvers have been set with the same numerical precision ($\varepsilon_c^a, \varepsilon_J^a = 10^{-6}$, $\varepsilon_c^r, \varepsilon_J^r = 10^{-4}$), and warm-starting techniques are not used in any algorithm. We report the time taken to execute the main function of each solver as called from Matlab and run on a serial (single-core) processor, averaged for each test on five runs. Note that a parallel implementation of PQP would offer a speed-up approximately linear in the number of cores/processors, especially when there is no-communication delay such as in GPU, multiprocessors, and multicore architectures. In fact such a speed up is already visible for the M-coded implementation of PQP when executed in Matlab with enabled multithreading, due to the automated parallelization of core functions such as vector sums and matrix-vector products.

Remark 9: By the results in the following test cases, we do not aim at claiming the superiority in terms of execution speed of the approach proposed here. In fact, there are always problem instances where an algorithm is outperformed by another one. By the following results we only aim at showing that the approaches proposed here are capable of operating at rates with the same orders of magnitude as other well established and currently proposed methods. This, together with the code simplicity, makes the algorithm proposed here very attractive for several applications domains.

B. Stabilization of a constrained double integrator

In the first case study we consider a system modeling a double integrator with state $x_c \in \mathbb{R}^2$, input $u_c \in \mathbb{R}$, output $y_c \in \mathbb{R}$, and for which the dynamics are expressed in continuous time as

$$\dot{x}_c(t) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_c(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_c(t), \tag{38a}$$

$$y_c(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x_c(t). \tag{38b}$$

The double integrator (38) is subject to constraints on the inputs and on the "velocity" state

$$-1 \leq u_c(t) \leq 1, \tag{39a}$$

$$-1 \leq [x_c(t)]_2. \tag{39b}$$

The double integrator model (38) is sampled with period $T_s = 1$s to obtain the discrete-time model (21). The objective is to drive the system state to the origin.

We design an MPC controller with prediction, constraint, and control horizons $N = N_c = N_{cu} = N_u = 4$, resulting in a parametric QP with $n_q = 12$ constraints, and $n_u = 4$ variables. The MPC cost function (23a) is implemented with

$$Q_M = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \; R_M = 0.8,$$

[5]This algorithm was implemented by the authors of this paper, but it was shown to and discussed with the authors of [23] to verify its correct implementation.

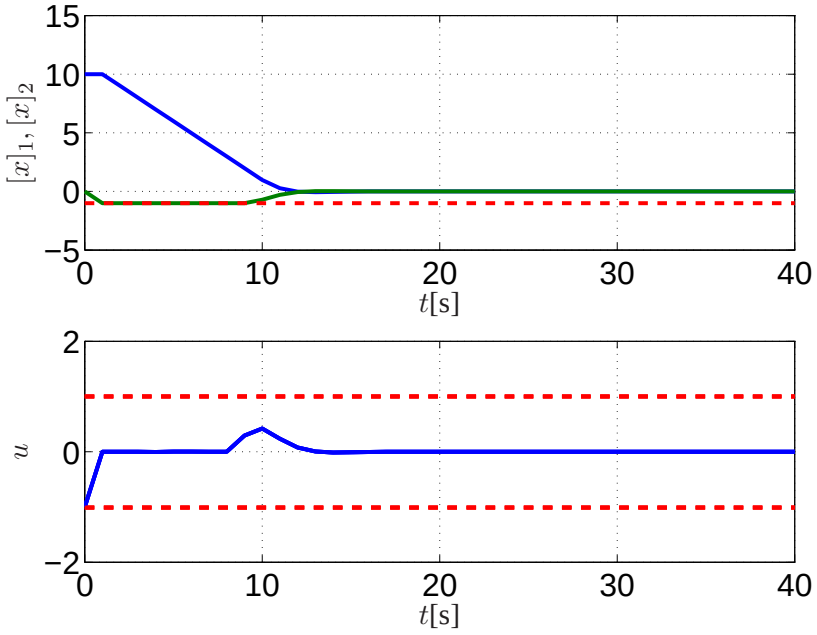

Fig. 2: Trajectories for the double integrator case study. Upper plot: states (solid), constraint (dash). Lower plot: control inputs (solid), constraints (dash).

and the matrix P_M and the terminal controller K_f in (23) are obtained by solving the discrete-time Riccati equation

$$P_M = A'P_MA - A'P_MB(B'P_MB + R_M)^{-1}B'P_MA + Q_M, \quad (40a)$$

$$K_f = -(B'P_MB + R_M)^{-1}B'P_MA\ . \quad (40b)$$

We consider the initial state $x(0) = [10\ 0]'$ and we simulate the system in closed-loop with the MPC controller for 40s. The state trajectory and the input profile are reported in Figure 2.

The average, minimum, and maximum computation time for solving the QP problem along a 40s simulation are reported in Table I. The solution time of the M-coded PQP has the same order of magnitude of QPROG, which

TABLE I: Computation time results for the double integrator case study.

Solver	Avg[ms]	Min[ms]	Max[ms]
PQP-M:	**1.086**	0.323	**5.097**
GPAD-M:	1.750	**0.207**	9.440
QPROG:	1.597	1.349	2.380
QPACT:	0.450	0.356	0.629
QPOAS:	0.146	0.116	0.227
NAG:	0.614	0.355	0.839
PQPMEX:	0.065	0.038	0.135
PQPMPC:	**0.031**	**0.013**	**0.104**

exploits C-mex routines. The computation time of PQPMEX and of PQPMPC are well below 1ms, and outperform the other solvers.

Figure 3 shows the distribution of the computation time for PQPMEX (in red) and NAG (in blue). Note that the largest computation time of PQPMEX is smaller than the smallest computation time of NAG.

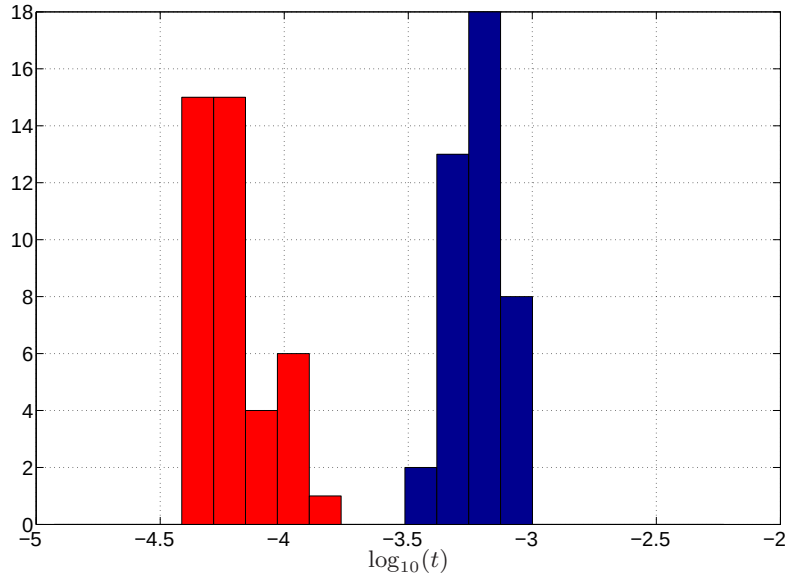

Fig. 3: Distribution of the computation time for the double integrator case study for PQPMEX (red) and NAG (blue).

C. Control of pitch and angle of attack of an unstable jet aircraft

The second case study is the control of pitch and angle of attack of a jet aircraft [50]. The linearized continuous time dynamics results in a model with $x_c \in \mathbb{R}^4$, $u_c \in \mathbb{R}^2$, $y_c \in \mathbb{R}^2$, and

$$\dot{x}_c(t) = \begin{bmatrix} -0.015 & -60.6 & 0 & -32.2 \\ -0.0001 & -1.34 & 0.992 & 0 \\ 0.0002 & 43.2 & -0.869 & 0 \\ 0 & 0 & 1.0 & 0 \end{bmatrix} x_c(t) + \begin{bmatrix} -2.51 & -13.1 \\ -0.169 & -0.251 \\ -17.2 & -1.58 \\ 0 & 0 \end{bmatrix} u_c(t), \tag{41a}$$

$$y_c(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_c(t) \ , \tag{41b}$$

where y_c models the angle of attack and the pitch angle, and u_c models the elevator and flaperon angles. Model (41) has one real stable pole, one real unstable pole and two lightly damped stable complex poles. The aircraft is subject to the constraints

$$\begin{bmatrix} -0.5 \\ -90 \end{bmatrix} \leq y_c(t) \leq \begin{bmatrix} 0.5 \\ 90 \end{bmatrix}, \tag{42a}$$

$$\begin{bmatrix} -25 \\ -25 \end{bmatrix} \leq u_c(t) \leq \begin{bmatrix} 25 \\ 25 \end{bmatrix}, \tag{42b}$$

where all the angles are measured in degrees.

The objective of the controller is to track references for pitch and angle of attack. Thus model (41) is sampled with period $T_s = 0.05$s, and augmented with a reference prediction model that assumes a constant reference,

$$r_r(k+1) = r_r(k),$$

where $r_r \in \mathbb{R}^2$, and with an incremental formulation of the control input

$$v(k+1) = v(k) + \Delta v(k) = v(k) + u(k),$$

where $v \in \mathbb{R}^2$ is the vector of the elevator and flaperon angle commands and $u \in \mathbb{R}^2$ is the vector of step-to-step variation of such angles, thus resulting in prediction model (21), subject to constraints (22) that enforce (42).

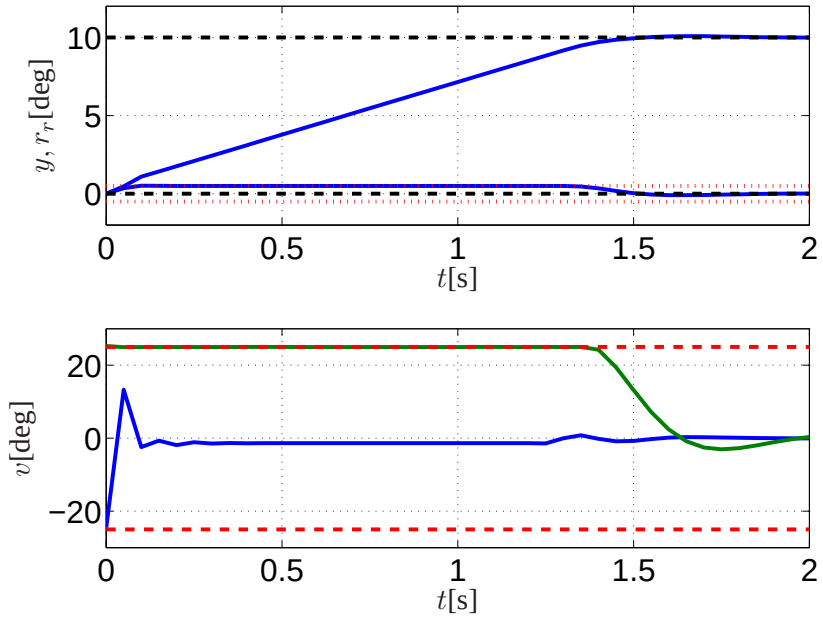

Fig. 4: Trajectories for the jet aircraft control case study. Upper plot: outputs (solid), references (dash), constraints (dot). Lower plot: control inputs (solid), constraints (dash).

We design an MPC controller where the cost function is

$$J_{\text{MPC}} = \sum_{i=1}^{N} \|y(i|k) - r_r(i|k)\|^2_{Q_y} + \|\Delta v(i|k)\|^2_{R_v},$$

where $Q_y = \left[\begin{smallmatrix} 10 & 0 \\ 0 & 10 \end{smallmatrix}\right]$ and $R_v = \left[\begin{smallmatrix} 0.01 & 0 \\ 0 & 0.01 \end{smallmatrix}\right]$, which is then formulated as (23a). The prediction, constraints, and control horizons are set equal to $N = N_c = N_{cu} = N_u = 6$, resulting in a parametric QP (31) with $n_q = 48$ constraints, and $n_u = 12$ variables.

TABLE II: Computation time results for the jet aircraft control case study.

Solver	Avg[ms]	Min[ms]	Max[ms]
PQP-M:	**19.168**	0.297	**51.778**
GPAD-M:	174.898	**0.230**	398.593
QPROG:	4.146	1.545	8.591
DANTZ:	0.661	0.483	1.063
QPOAS:	0.663	0.496	1.033
NAG:	0.791	0.388	1.584
PQPMEX:	0.890	0.103	1.593
PQPMPC:	**0.452**	**0.049**	**0.986**

The output and reference trajectories and the input profile for a 2s simulation where $x(0) = [0\ 0\ 0\ 0]'$ and $r_r = [10\ 0]'$ are shown in Figure 4, where one can see that during most of the simulation, several constraints are active. The average, minimum, and maximum computation time for solving the QPs along the simulation are reported in Table II.

The PQPMEX solver is slightly slower than other algorithms, but the PQPMPC controller is faster, due to moving offline several matrix calculations thus reducing the dataflows and avoiding dynamic memory allocation. Still it is worth noting that the PQP algorithms are much simpler than the ones they are compared with, even though the PQP code is not optimized. The variability between minimum and maximum computation time is due to the cases where the constraints are not active, versus the ones where many constraints are active. In the former ones, PQP

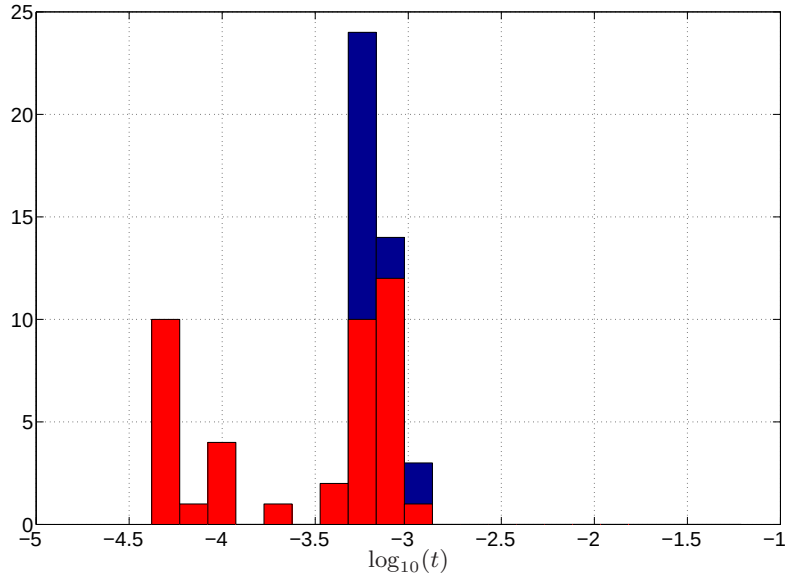

Fig. 5: Distribution of the computation time for the jet aircraft control case study for PQPMPC (red) and DANTZ (blue).

converges extremely fast to the solution. The distribution of the computation time of the PQPMPC controller (red) and DANTZ (blue) is shown in Figure 5, where one can see that the computation time for PQPMPC is distributed in two areas. The one with smaller time value is due to few or no active constraints, and the one at larger time value is due to many active constraints. In any case, the largest computation time of PQPMPC is slightly smaller than the largest computation time of the other solvers.

D. DC-motor position control

The third case study is the angular position control of a load connected by a flexible shaft to a voltage actuated DC motor [51]. The states are the load angle and angular rate, and the motor angle and angular rate, the control input is the motor voltage, and the outputs are the load angle and the torque acting on the flexible shaft. The model for the system is

$$\dot{x}_c(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_l}{\mathcal{J}_l} & -\frac{\beta_l}{\mathcal{J}_l} & \frac{k_l}{g\mathcal{J}_l} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_l}{g\mathcal{J}_m} & 0 & -\frac{k_l}{g^2\mathcal{J}_m} & -\frac{\beta_m + R_A^{-1}K_m^2}{\mathcal{J}_m} \end{bmatrix} x_c(t) \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K_m}{R_A\mathcal{J}_m} \end{bmatrix} u_c(t) \tag{43a}$$

$$y_c(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ k_l & 0 & -\frac{k_l}{g} & 0 \end{bmatrix} x_c(t) \tag{43b}$$

where $x_c \in \mathbb{R}^4$ is the state vector, $u_c \in \mathbb{R}$ is the input vector, and $y_c \in \mathbb{R}^2$ is the output vector. In (43) $R_A[\Omega]$ is the armature resistance, K_m[Nm/A] is the motor constant, $\mathcal{J}_l$[kgm^2], β_l[Nms/rad], k_l[Nm/rad], are the inertia, friction and stiffness of load and flexible shaft, $\mathcal{J}_m$[kgm^2], β_m[Nms/rad], are the inertia and friction of the motor, and g is the gear ratio between motor and load. The numerical values used in the simulations are $R_A = 10\Omega$, $K_m = 10$Nm/A, $\mathcal{J}_l = 25$kgm^2, $\beta_l = 25$Nms/rad, $k_l = 1.28 \cdot 10^3$Nm/rad, $\mathcal{J}_m = 0.5$kgm^2, $\beta_m = 0.1$Nms/rad. The system is subject to constraints on motor voltage and shaft torque

$$-78.5 \leq [y_c(t)]_2 \leq 78.5, \tag{44a}$$

$$-220 \leq u_c(t) \leq 220. \tag{44b}$$

The control objective is to track a time varying load angle position reference signal $r_l(t)$, and the prediction

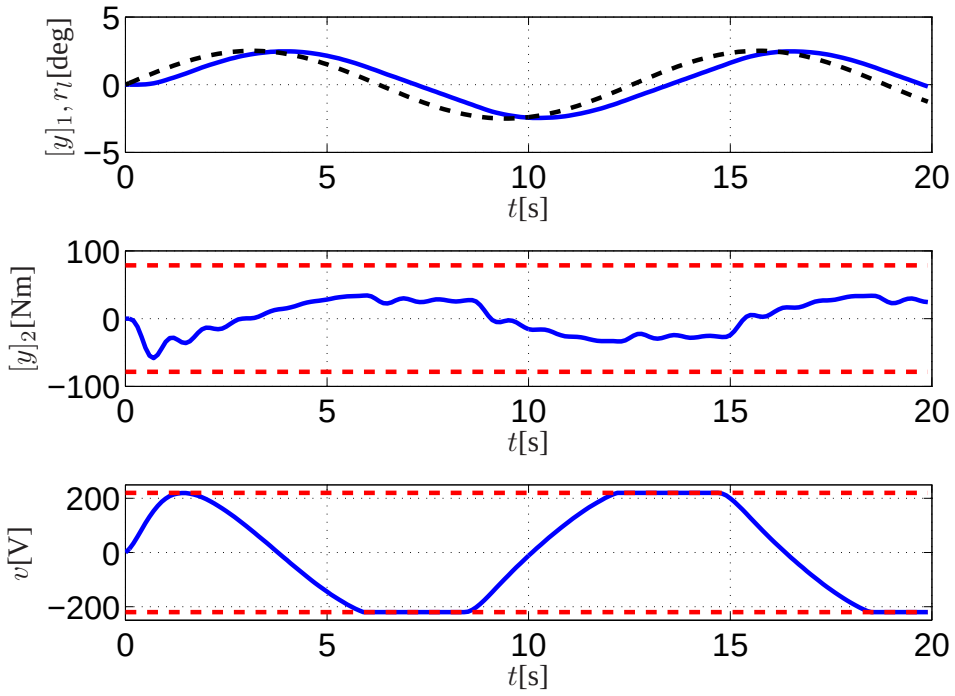

Fig. 6: Trajectories in the DC-motor control simulation for reference with $a_r = 2.5$. Upper plot: load angle (solid), reference (dash). Middle plot: shaft torque (solid), constraints (dash). Lower plot: control inputs (solid), constraints (dash).

model is obtained by sampling (43) with period $T_s = 0.1s$, and augmenting the model with a constant reference prediction model and with an incremental input formulation as for the case study in Section V-C. The cost function is

$$J_{\text{MPC}} = \sum_{i=1}^{N} \|[y(i|k)]_1 - r_l(i|k)\|^2_{Q_y} + \|\Delta v(i|k)\|^2_{R_v},$$

where $Q_y = 10^3$ and $R_v = 0.05$, prediction, constraints, and control horizons are $N = 20$, $N_c = N_{cu} = N_u = 4$, and $K_f = 0$ in (23g), resulting in a parametric QP with $n_q = 16$ constraints, and $n_u = 4$ variables. In the subsequent simulations, we consider initial state $x(0) = [0\ 0\ 0\ 0]'$ and reference $r_l(t) = a_r \sin(0.5t)$, where $a_r > 0$ changes in the different simulations.

The trajectories for reference and output and the input profile for a simulation of 20s from initial state where $a_r = 2.5$ are reported in Figure 6 that shows that only voltage constraints are active. The computation time is

TABLE III: Computation time results for the DC-motor case study for reference with $a_r = 2.5$.

Solver	Avg[ms]	Min[ms]	Max[ms]
PQP-M:	**0.588**	0.304	**2.264**
GPAD-M:	3.450	**0.252**	18.733
QPROG:	1.545	1.348	2.557
QPACT:	0.454	0.321	0.920
QPOAS:	0.210	0.155	0.376
NAG:	0.610	0.410	1.115
PQPMEX:	0.068	0.041	0.154
PQPMPC:	**0.022**	**0.018**	**0.049**

reported in Table III.

Next, we simulate the trajectory obtained for an aggressive reference signal where $a_r = 4.0$. The obtained trajectory and control input profile are reported in Figure 7, where one can see that both voltage and torque

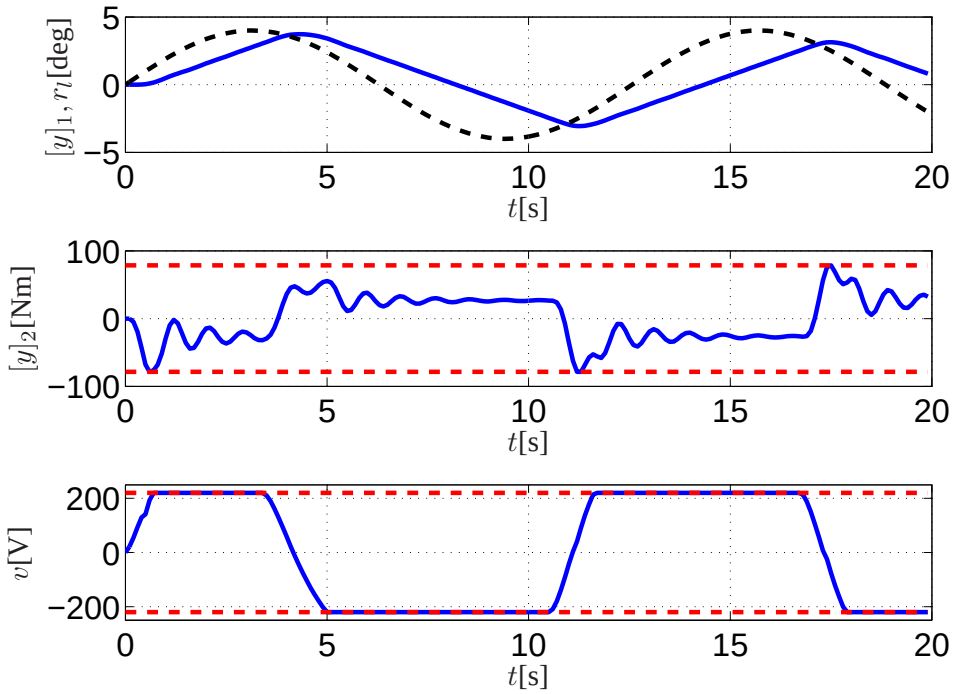

Fig. 7: Trajectories in the DC-motor control simulation for reference with $a_r = 4.0$. Upper plot: load angle (solid), reference (dash). Middle plot: shaft torque (solid), constraints (dash). Lower plot: control inputs (solid), constraints (dash).

constraints are active. The system response is dominated by the active constraints, so that the position is not achieving the reference profile. The computation time for this case is reported in Table IV, and the comparison between the PQPMPC computation time distribution for the cases where $a_r = 2.5$ (blue) and $a_r = 4.0$ (red) is reported in Figure 8. The results show that PQP usually needs more time when (many) constraints are active in

TABLE IV: Computation time results for the DC-motor case study for reference with $a_r = 4.0$.

Solver	Avg[ms]	Min[ms]	Max[ms]
PQP-M:	**2.037**	0.305	**5.819**
GPAD-M:	20.596	**0.246**	38.820
QPROG:	2.139	1.344	11.121
QPACT:	0.534	0.364	0.721
QPOAS:	0.190	0.114	0.371
NAG:	0.516	0.267	0.955
PQPMEX:	0.113	0.042	0.242
PQPMPC:	**0.062**	**0.028**	**0.107**

the QP (24), that is when many dual variables in (26) are non-zero. This is the opposite of solvers such as NAG, which are most effective when many constraints are active [48]. The behavior of PQP can certainly be influenced by appropriate initialization, but such warm-starting techniques are not used here.

The impact of the acceleration technique described in Section II-B executed once every 20 PQP iterations is shown in Figure 9 in terms of number of iterations executed in the algorithm. The acceleration is particularly effective in reducing the high peaks in number of iterations, and hence the optimization of the acceleration strategy, which is currently being studied, may lead to an effective reduction of the variance in the computation time.

For this case study we evaluate how the computation time changes while increasing the problem size. We select the "aggressive" reference signal ($a_r = 4$), and we compare the computation time results of PQPMPC and NAG, for the case in Table IV, and for the cases where $N = 30$, $N_c = N_{cu} = N_u = 10$, where $N = 40$,

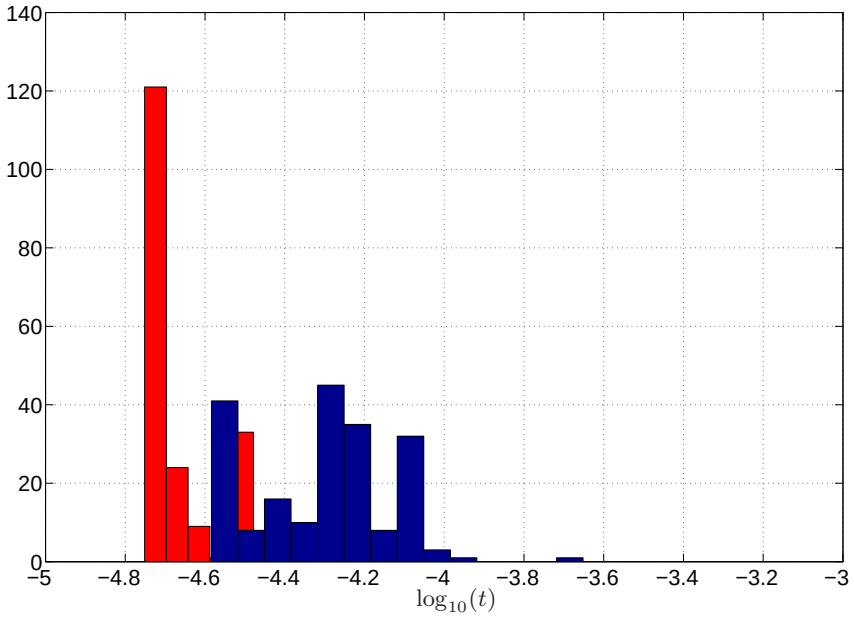

Fig. 8: Distribution of the computation time of PQPMPC in DC-motor control simulations for references with $a_r = 2.5$ (blue) and $a_r = 4.0$ (red).

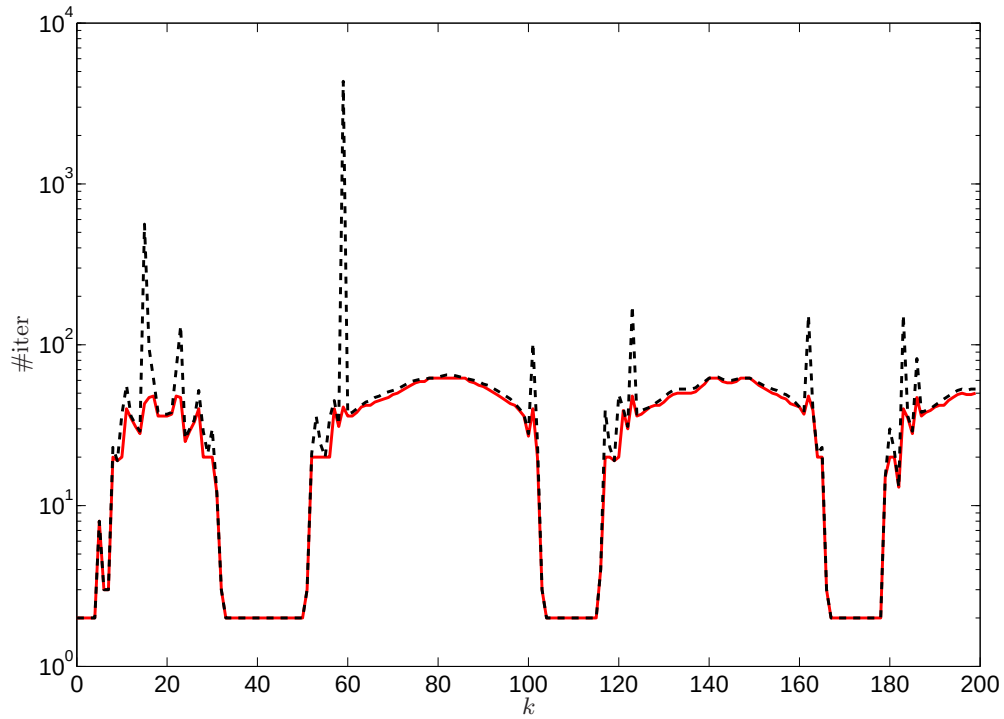

Fig. 9: Effect of the acceleration technique on the DC-motor control case study for reference with $a_r = 4.0$. Number of iterations without acceleration (dash) and with acceleration (solid) every 20 PQP iterations.

$N_c = N_{cu} = N_u = 20$, where $N = 80$, $N_c = N_{cu} = N_u = 40$, and where $N = 160$, $N_c = N_{cu} = N_u = 80$. Due to the many active constraints, the inertia-controlling algorithm in the NAG solver is particularly effective

TABLE V: Computation time results for different problem sizes for PQPMPC and NAG solver for the DC motor case study.

Solver	Avg[ms]	Min[ms]	Max[ms]
NAG-04:	0.516	0.267	1.153
NAG-10:	0.693	0.328	2.023
NAG-20:	1.306	0.520	4.840
NAG-40:	6.237	1.538	20.233
NAG-80:	28.847	3.318	348.110
PQPMPC-04:	0.062	0.028	1.933
PQPMPC-10:	0.302	0.044	1.063
PQPMPC-20:	1.019	0.117	3.190
PQPMPC-40:	3.494	0.384	7.917
PQPMPC-80:	7.301	5.478	12.237

for this case, and in fact its computation time varies less throughout the different cases. However, the PQPMPC algorithm tends to always execute slightly faster, partly due to computing offline part of the QP matrices. It shall also be remarked that PQPMPC is significantly simpler than the NAG algorithm and it can obtain clear benefits from (massive) parallelization.

E. Memory occupancy and code complexity

For the three case studies presented so far we have mainly discussed the computing time of PQP algorithms when compared to other established and developing algorithms. These comparisons have to be considered as indicative, because in many cases small optimizations in the code and in the options may drastically change the performance. However, those are known only to the developers or to extremely experienced users of the algorithm. We suggest that these results are to be read as indicating that the algorithms proposed here can execute at the same time scales of the algorithms used for comparison. In fact, for applications with fast dynamics and limited computational resources, such as automotive, aerospace, and factory automation, the most appealing qualities of the PQP algorithms are the simplicity and the memory occupancy. The simplicity of PQP, which is evident from Algorithms 1–3, results in a code that has very small memory occupancy, and that is simple to validate and certify. Code certification before its application to real products is a complex and time consuming step. An algorithm with few lines of code may be certified in some weeks, while complex algorithms may require years of testing and validation before being certified. The simplicity of the code is an element shared also by other iterative algorithms, such as the fast gradient algorithm in [18], [23], and in fact the importance of code simplicity for solver certification was highlighted in [23].

As regards memory, too often memory occupancy is ignored when discussing numerical control algorithms. In applications such as automotive [7], [8], [52], the memory requirements are actually several orders of magnitude more stringent than the chronometrics requirements. For instance, automotive control units running several tens of control loops and significant amount of control logics may have only in the order of a megabyte of memory for code and data, for all the controllers and logics. Due to the simplicity of the code, the PQPMEX function code requires less than 30kB memory, including the Matlab interfaces, as opposed to the more than 300kB of qpOASES. Also, for the PQPMPC controller, the memory occupancy of code and data can be precisely evaluated, since all the matrices are pre-allocated. For all the PQPMPC controllers but the last three cases in Table V, the memory occupancy for code and data was less than 50kB. For the largest case in Table V with dense matrices, the memory occupancy of the PQPMPC controller reached almost 1MB, with only less than 30kB due to the code. In fact, memory occupancy is one of the reasons that limit the number of variables and constraints of MPC problems for the application domains mentioned above.

VI. Conclusions

In this paper we have introduced an algorithm for the solution of non-negative least squares problems and we have shown how the base algorithm can be accelerated using a projection-free line search. The algorithm is extremely simple, offers a linear convergence rate, does not require a-posteriori projection, and is easily parallelizable. We

have discussed how the algorithm can be applied to the quadratic program formulated for linear MPC, and how it can be equipped with termination conditions that guarantee the desired degree of suboptimality. Finally, we have shown how the online computations can be reduced by to pre-computing and pre-allocating most of the data, thus reducing the computation time. The algorithm has been compared with some available free and commercial solvers in three classical case studies for linear MPC, showing interesting performance in terms of computation time and memory occupancy. Future work will involve optimizing the acceleration strategy, defining warm-start and termination strategies that are specific MPC, and deriving a bound on the number of iterations.

ACKNOWLEDGEMENTS

The authors acknowledge Prof. A.V. Knyazev, Univ. Colorado, Denver and MERL, for useful insight in interpreting PQP as a variable-preconditioned iterative algorithm, Dr. P. Patrinos, IMT Lucca, for useful insights in the implementation of the GPAD algorithm, and several colleagues and interns at MERL for testing the implementation of PQP algorithms and providing useful data and comments.

REFERENCES

[1] J. Rawlings and D. Mayne, *Model predictive control: Theory and design*. Madison, WI: Nob Hill Publishing, LLC, 2009.
[2] S. Qin and T. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 93, no. 316, pp. 733–764, 2003.
[3] S. Di Cairano, D. Yanakiev, A. Bemporad, I. Kolmanovsky, and D. Hrovat, "Model predictive idle speed control: Design, analysis, and experimental evaluation," *IEEE Tr. Contr. Sys. Technology*, vol. 20, no. 1, pp. 84 –97, 2012.
[4] T. Fan and C. De Silva, "Dynamic modelling and model predictive control of flexible-link manipulators," *Int. Jour. Robotics and Automation*, vol. 23, no. 6, pp. 227–234, 2008.
[5] E. Hartley, J. Jerez, A. Suardi, J. Maciejowski, E. Kerrigan, and G. Constantinides, "Predictive control of a boeing 747 aircraft using an fpga," in *Proc. 4th IFAC Nonlinear Model Predictive Control Conference*, Noordwijkerhout, The Netherlands, 2012, pp. 80–85.
[6] S. Di Cairano, H. Park, and I. Kolmanovsky, "Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering," *Int. J. Rob. Nonlinear Control*, 2012, special Issue dedicated to Prof. David W. Clarke. In press.
[7] S. Di Cairano, "An industry perspective on MPC in large volumes applications: Potential Benefits and Open Challenges," in *Proc. 4th IFAC Nonlinear Model Predictive Control Conference*, Noordwijkerhout, The Netherlands, 2012, pp. 52–59.
[8] D. Hrovat, S. Di Cairano, H. Tseng, and I. Kolmanovsky, "The development of model predictive control in automotive industry: A survey," in *IEEE Int. Conf. Control Applications*, Dubrovnik, Croatia, 2012, pp. 295–302.
[9] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.
[10] J. Nocedal and S. Wright, *Numerical optimization*. Springer verlag, 1999.
[11] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *J. optimization theory and applications*, vol. 99, no. 3, pp. 723–757, 1998.
[12] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Tr. Contr. Sys. Technology*, vol. 18, no. 2, pp. 267–278, 2010.
[13] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, pp. 1–27, 2012.
[14] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Int. J. Rob. Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.
[15] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The Explicit Linear Quadratic Regulator for Constrained Systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.
[16] G. Stewart and F. Borrelli, "A Model Predictive Control Framework for Industrial Turbodiesel Engine Control," in *Proc. 47th IEEE Conf. on Dec. and Control*, Cancun, Mexico, Dec 2008, pp. 5704–5711.
[17] S. Di Cairano, H. Tseng, D. Bernardini, and A. Bemporad, "Vehicle yaw stability control by coordinated active front steering and differential braking in the tire sideslip angles domain," *IEEE Tr. Contr. Sys. Technology*, pp. 1–13, 2012, in press.
[18] S. Richter, C. Jones, and M. Morari, "Real-time input-constrained mpc using fast gradient methods," in *Proc. 48th IEEE Conf. on Dec. and Control*, Shangai, China, 2009, pp. 7387–7393.
[19] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.
[20] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," University of Washingthon, Dept. Mathematics, Tech. Rep., 2008.
[21] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.
[22] S. Richter, M. Morari, and C. Jones, "Towards computational complexity certification for constrained mpc based on lagrange relaxation and the fast gradient method," in *Proc. 50th IEEE Conf. on Dec. and Control*, Orlando,FL, 2011, pp. 5223–5229.
[23] A. Bemporad and P. Patrinos, "Simple and certifiable quadratic programming algorithms for embedded control," in *Proc. 4th IFAC Nonlinear Model Predictive Control Conference*, Noordwijkerhout, The Netherlands, 2012.
[24] M. Kögel and R. Findeisen, "A fast gradient method for embedded linear predictive control," in *Proc. IFAC World Congress*, Milan, Italy, 2011, pp. 1362–1367.

[25] I. Necoara and J. Suykens, "Application of a smoothing technique to decomposition in convex optimization," *IEEE Tr. Automatic Control*, vol. 53, no. 11, pp. 2674–2679, 2008.
[26] P. Giselsson, "Execution time certification for gradient-based optimization in model predictive control," in *Proc. 51st IEEE Conf. on Dec. and Control*, Maui, HI, 2012, pp. 3165–3170.
[27] D. P. Bertsekas, "Projected newton methods for optimization problems with simple constraints," *SIAM J. Control and Optimization*, vol. 20, no. 2, pp. 221–246, 1982.
[28] M. Brand, V. Shilpiekandula, C. Yao, S. Bortoff, T. Nishiyama, S. Yoshikawa, and T. Iwasaki, "A parallel quadratic programming algorithm for model predictive control," in *Proc. IFAC World Congress*, Milan, Italy, 2011.
[29] S. Di Cairano and M. Brand, "On a multiplicative update dual optimization algorithm for constrained linear mpc," 2013, submitted.
[30] F. Sha, Y. Lin, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming," *Neural Computation*, vol. 19, no. 8, pp. 2004–2031, 2007.
[31] M. Brand and D. Chen, "Parallel quadratic programming for image processing," in *Proc. 18th IEEE Int. Conf. Image Processing*, 2011, pp. 2261–2264.
[32] R. Myers, *Classical and modern regression with applications*. Duxbury Press Belmont, CA, 1990.
[33] G. C. Goodwin, M. M. Seron, and J. A. De Doná, *Constrained Control and Estimation: And Optimization Approach*. London, UK: Springer, 2005.
[34] K. G. Murty, *Linear complementarity, linear and nonlinear programming*. Berlin, Germany: Heldermann, 1988.
[35] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 1990.
[36] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM Jour. scientific computing*, vol. 23, no. 2, pp. 517–541, 2001.
[37] H. Khalil, *Nonlinear systems*. Prentice hall, New Jersey, 1996.
[38] D. Liberzon and A. Morse, "Basic problems in stability and design of switched systems," *Control Systems Magazine*, vol. 19, no. 5, pp. 59–70, 1999.
[39] S. Di Cairano, A. Bemporad, I. Kolmanovsky, and D. Hrovat, "Model predictive control of magnetically actuated mass spring dampers for automotive applications," *Int. J. Control*, vol. 80, no. 11, pp. 1701–1716, 2007.
[40] A. Grancharova and T. Johansen, "Explicit model predictive control of an electropneumtaic clutch actuator using on/off valves and pulsewidth modulation," in *Proc. European Control Conf.*, Budapest, Hungary, 2009, pp. 4278–4283.
[41] J. Maciejowski, *Predictive control with constraints*. Englewood Cliffs, NJ: Prentice Hall., 2002.
[42] S. Di Cairano and A. Bemporad, "Model predictive control tuning by controller matching," *IEEE Tr. Automatic Control*, vol. 55, no. 1, pp. 185–190, 2010.
[43] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. New York, NY: Springer, 2011, vol. 24.
[44] A. Bemporad, *Hybrid Toolbox – User's Guide*, Dec. 2003.
[45] The Mathworks Inc., *Matlab Optimization Toolbox*, Natick, MA, 2011, http://www.mathworks.com.
[46] N. Ricker, "Use of quadratic programming for constrained internal model control," *Ind. Eng. Chem. Process Design Devel.*, vol. 24, no. 4, pp. 925–936, 1985.
[47] A. Bemporad, M. Morari, and N. L. Ricker, *Model Predictive Control Toolbox User's Guide v.3*. Mathworks Inc., 2008.
[48] NAG Ltd., `e04nf`, *E04 Minimizing or Maximizing a Function*, in *NAG Toolbox for MATLAB*, Oxford, U.K., 2011, http://www.nag.co.uk.
[49] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for linear model predictive control," in *Proc. 51st IEEE Conf. on Dec. and Control*, Maui, HI, 2012, pp. 662–667.
[50] P. Kapasouris, M. Athans, and G. Stein, "Robust vehicle stability controller based on multiple sliding mode control," in *Proc. 1st IFAC Symp. Nonlin. Control Design*. New York: Pergamon, 1990, pp. 302–307.
[51] A. Bemporad and E. Mosca, "Fulfilling hard constraints in uncertain linear systems by reference managing," *Automatica*, vol. 34, no. 4, pp. 451–461, 1998.
[52] S. Di Cairano, W. Liang, I. Kolmanovsky, M. Kuang, and A. Phillips, "Engine power smoothing energy management strategy for a series hybrid electric vehicle," in *Proc. of the American Control Conference*, San Francisco, CA, 2011, pp. 2101–2106.