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

On a Multiplicative Update Dual Optimization Algorithm for Constrained Linear MPC

Di Cairano, S.; Brand, M.

TR2013-108 December 2013

Abstract

We discuss a multiplicative update quadratic programming algorithm with applications to model predictive control for constrained linear systems. The algorithm, named PQP, is very simple to implement and thus verify, does not require projection, offers a linear rate of convergence, and can be completely parallelized. The PQP algorithm is equipped with conditions that guarantee the desired bound on sub-optimality and with an acceleration step based on projection-free line search. We also show how PQP can take advantage of the parametric structure of the MPC problem, thus moving offline several calculations and avoiding large input/output dataflows. The algorithm is evaluated on two benchmark problems, where it is shown to compete with, and possibly outperform, other open source and commercial packages.

IEEE Conference on Decision and Control (CDC)

On a Multiplicative Update Dual Optimization Algorithm for Constrained Linear MPC

Stefano Di Cairano, Matt Brand

Abstract— **We discuss a multiplicative update quadratic programming algorithm with applications to model predictive control for constrained linear systems. The algorithm, named PQP, is very simple to implement and thus verify, does not require projection, offers a linear rate of convergence, and can be completely parallelized. The PQP algorithm is equipped with conditions that guarantee the desired bound on suboptimality and with an acceleration step based on projection-free line search. We also show how PQP can take advantage of the parametric structure of the MPC problem, thus moving offline several calculations and avoiding large input/output dataflows. The algorithm is evaluated on two benchmark problems, where it is shown to compete with, and possibly outperform, other open source and commercial packages.**

I. Introduction

Model predictive control (MPC) is a powerful framework for controlling dynamical systems subject to input and state constraints. MPC is based on the receding horizon solution of a finite time optimal control problem formulated on the system dynamics, constraints, and performance objective. Besides process control, in recent years other domains have been shown interest in MPC including automotive, aerospace, and factory automation [1]–[3]. In these applications, the MPC controller is executed on low computational power processors, and the controller update rate may be higher than 1KHz. As a consequence, a key enabler for applying MPC in these domains are fast, low-complexity algorithms for solving the optimal control problem that need to be simple to code and verify, have limited memory occupancy, and be capable of solving small scale problems (from ten to hundred variables) within milliseconds or less.

For linear prediction models with convex quadratic performance objectives subject to polyhedral constraints, the MPC finite time optimal control problem results in a constrained quadratic program (QP) for which convergence to the global optimum is guaranteed [4]. Powerful methods for solving QP are active set methods [5] and interior points methods [4]. Recently, MPC-tailored interior point solvers [6], [7] and active set solvers [8] have been introduced. Active set and interior point methods have high performance, especially when taking advantage of the MPC problem structure [6]–[8], but require the solution of severals systems of linear equations. Thus, they need complex routines for linear algebra [5], resulting in complex code and significant memory use.

An alternative approach is based on the explicit solution [9] of the MPC parametric quadratic program, which avoids online optimization. Explicit MPC has been successful especially for automotive controls [2], [10], [11]. However, due to combinatorial complexity, explicit MPC is feasible only for short prediction horizons.

Recently, gradient-based iterative algorithms have been proposed for MPC. In [12], [13], fast-gradient algorithms have been introduced. In [14], algorithms based on the fast gradient method combined with the Lagrange method of multipliers have been developed , and accelerated gradient methods for distributed MPC have been developed in [15], [16]. For fast gradient-based algorithms is that a bound on the number of iterations to converge to the solution can be computed. The iterative algorithms [12]–[14] perform only simple operations that do not require linear algebra routines to update the current solution, and then project the updated solution into the feasible set. While in these approaches projection is inexpensive, it is known to reduce (possibly significantly) the solution improvement obtained in the iteration [17].

In this paper we propose a projection-free constrained QP algorithm for MPC application. The algorithm, related to the methods in [18], was originally proposed in [19] for solving nonnegative least squares (NNLS) problems in image processing, and it was named Parallel Quadratic Programming (PQP). The PQP iteration is projection-free for NNLS because when starting from a solution in the positive cone, it guarantees that the updated solution remains in such cone. In this paper we provides several improvements for tailoring PQP to the optimization problems of MPC.

In this paper, in Section II we discuss the QP problem of linear MPC and the notion of suboptimal solution. In Section III we review the PQP algorithm, we introduce the acceleration technique, and we define the termination conditions. In Section IV we propose a synthesis for the PQP-based MPC controller that exploits the parametric form of the QP. In Section V we present simulations on two benchmark problems and compare PQP with other 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 $\phi \in \mathbb{R}^n$, $[\phi]_i$ 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 a vector of size m entirely composed of ones by $\mathbf{1}_m$, the identity matrix of size m by I_m, and the matrix entirely composed of zeros by 0_m, where subscripts are dropped when clear from context. For vectors, absolute value, maximum, and inequalities

S. Di Cairano and M. Brand are with the Mitsubishi Electric Research Laboratories, Cambridge, MA, {dicairano,brand}@merl.com

are intended componentwise, while for a symmetric matrix Q, $Q > 0$ ($Q \geq 0$) indicates positive (semi)definiteness. For a vector x and a matrix $Q \geq 0$, $\|x\|_Q^2 = x'Qx$. For a signal a sampled with period T_s, a_k is the value at the k^{th} sampling instant, i.e., at time kT_s, and $a_{i|k}$ denotes the predicted value of a at step $k+i$, based on data at step k. Given $\Phi \in \mathbb{R}^{n\times m}$, we define $\Phi^+, \Phi^- \in \mathbb{R}^{n\times m}$ where $[\Phi^+]_{ij} = \max(0, [\Phi]_{ij})$ $[\Phi^-]_{ij} = \max(0, -[\Phi]_{ij})$. For the optimization problem $\min_{z\in\mathcal{Z}} J(z)$, the optimum is J^* and the optimal solution[1] is z^*, i.e., $J^* = J(z^*)$.

II. Constrained linear MPC

Linear MPC is based on the prediction model

$$x_{k+1} = Ax_k + Bu_k \tag{1a}$$
$$y_k = Cx_k + Du_k, \tag{1b}$$

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

$$x_{\min} \leq x_k \leq x_{\max}, \tag{2a}$$
$$u_{\min} \leq u_k \leq u_{\max}, \tag{2b}$$
$$y_{\min} \leq y_k \leq y_{\max}, \tag{2c}$$

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 lower and upper bounds on state, input, and output vectors, respectively. At step $k \in \mathbb{Z}_{0+}$, MPC solves the finite-horizon optimal control problem

$$\min_{U_k} \quad \|x(N|k)\|_{P_{\rm M}}^2 + \sum_{i=0}^{N-1} \|x_{i|k}\|_{Q_{\rm M}}^2 + \|u_{i|k}\|_{R_{\rm M}}^2 \tag{3a}$$
$$\text{s.t.} \quad x_{i+1|k} = Ax_{i|k} + Bu_{i|k} \tag{3b}$$
$$y_{i|k} = Cx_{i|k} + Du_{i|k} \tag{3c}$$
$$x_{\min} \leq x_{i|k} \leq x_{\max},\ i \in \mathbb{Z}_{[1,N]} \tag{3d}$$
$$u_{\min} \leq u_{i|k} \leq u_{\max},\ i \in \mathbb{Z}_{[0,N-1]} \tag{3e}$$
$$y_{\min} \leq y_{i|k} \leq y_{\max},\ i \in \mathbb{Z}_{[0,N-1]} \tag{3f}$$
$$x_{0|k} = x_k, \tag{3g}$$

where $Q_{\rm M} \geq 0$, $P_{\rm M}, R_{\rm M} > 0$ are matrices of appropriate dimensions, N is the problem horizon, $U_k = [u'_{0|k} \dots u'(N-1|k)]' \in \mathbb{R}^{Nm}$ is the input vector to be optimized. At step k, (3) initialized from x_k is solved to obtain the optimal sequence U_k^*. Then, $u_k = u_{0|k}^*$ is applied to system (1). A new optimization problem is solved at step $k+1$.

Given x_k, (3) is formulated as the QP[2]

$$\min_U \quad J_p(U) = \frac{1}{2}U'Q_pU + F_p'U + \frac{1}{2}M_p \tag{4a}$$
$$\text{s.t.} \quad G_pU \leq K_p, \tag{4b}$$

where $U = U_k$, $Q_p \in \mathbb{R}^{n_u\times n_u}$, $n_u = N_u m$, $Q_p > 0$, $G_p \in \mathbb{R}^{n_q\times n_u}$, $K_p \in \mathbb{R}^{n_q}$, $M_p \in \mathbb{R}_{0+}$.

The following concept of solution is considered.

Definition 1: Consider (4) and the non-negative 4-tuple $\varepsilon \in \mathbb{R}_{0+}^4$, $\varepsilon = (\varepsilon_J^r\ \varepsilon_J^a\ \varepsilon_c^r\ \varepsilon_c^a)$. An ε-solution for (4) is a vector $\tilde{U}$ such that the constraint violation and the duality gap are ε-bounded in either relative $(\varepsilon_c^r, \varepsilon_J^r)$ or absolute $(\varepsilon_c^a, \varepsilon_J^a)$ errors

$$G_p\tilde{U} \leq K_p + \max\{\varepsilon_c^r|K_p|, \varepsilon_c^a\mathbf{1}\} \tag{5a}$$
$$J(\tilde{U}) - J(U^*) \leq \max\{\varepsilon_J^r|J(U^*)|, \varepsilon_J^a\}. \tag{5b}$$

III. Parallel Quadratic Programming

The Parallel Quadratic Programming (PQP) algorithm was originally developed in [19] to solve the NNLS problem

$$\min_z \quad J(z) = \frac{1}{2}z'Hz + F'z + M \tag{6a}$$
$$\text{s.t.} \quad z \geq 0, \tag{6b}$$

where $M \in \mathbb{R}_{0+}$, $z, F \in \mathbb{R}^{n_z}$, $H \in \mathbb{R}^{n_z\times n_z}$, $H \geq 0$, and $\frac{1}{2}z'Hz + F'z + M = \frac{1}{2}\|A_{\rm ls}z - b_{\rm ls}\|^2$, for appropriate matrices $A_{\rm ls}$, $b_{\rm ls}$. The KKT optimality conditions for (6) imply

$$[z]_i \cdot [\nabla_z L(z,\lambda)]_i = 0,\ i \in \mathbb{Z}_{[0,n_z]}, \tag{7a}$$
$$[\lambda]_i \cdot [\nabla_\lambda L(z,\lambda)]_i = 0,\ i \in \mathbb{Z}_{[0,n_z]}, \tag{7b}$$

where $L(z,\lambda) = \frac{1}{2}z'Hz + F'z - \lambda'z$ is the Lagrangian of (6). We derive the PQP iteration by considering

$$\begin{aligned}&[z]_i \cdot [\nabla_z L(z,\lambda)]_i = \\ &[z]_i \cdot [(H^+z + F^+) - (H^-z + F^- + \lambda)]_i,\ i \in \mathbb{Z}_{[0,n_z]}.\end{aligned} \tag{8}$$

Assume 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

$$[z]_i = \frac{[H^-z + F^-]_i}{[H^+z + F^+]_i}[z]_i, \quad \forall i \in \mathbb{Z}_{[1,n_z]}. \tag{9}$$

Thus, For a fixed diagonal matrix $\phi \in \mathbb{R}^{n_z\times n_z}$ that, as explained later, guarantees convergence, we define the iteration based on (9),

$$[z_{(h+1)}]_i = \frac{[(H^- + \phi)z_{(h)} + F^-]_i}{[(H^+ + \phi)z_{(h)} + F^+]_i}[z_{(h)}]_i \tag{10}$$

The iteration (10) needs to be initialized from an arbitrary strictly feasible value of (6), while the termination conditions are discussed extensively later. The convergence of the iteration (10) to the optimizer is guaranteed for a suitable choice of the matrix ϕ by the following theorem.

Theorem 1: Let be $z^* \in \mathbb{R}_{0+}^{n_z}$ be the optimum of (6) . For a proper choice of the diagonal matrix $\phi \geq 0$, (10) converges asymptotically to z^*, i.e., $\lim_{h\to\infty} z_{(h)} = z^*$.

Due to limited space the proofs are reported in [20].

While, the optimal choice of ϕ is not yet known, several choices that guarantee convergence are available, the simplest being $[\phi]_{ii} \geq [H^-\mathbf{1}]$. Next, we summarize some of the properties of the PQP algorithm.

Corollary 1: For (10), (i) the update rule is completely parallelizable, (ii) the problem objective decreases monotonically while the iteration maintains feasibility, (iii) the convergence rate is linear.

For PQP the dominant operation is a matrix-vector product, resulting in a computational complexity of $O(m_z p_z)$,

[1] If multiple optimal solutions exist, the definition applies to all.

[2] Extensions to (3) for output tracking, control and constraints horizons, stabilizing constraints, etc., are straightforward and still result in (4).

where p_z is the number of desired bits of precision and m_z is the number of nonzeros in H. For parallel implementations in SIMD and GPU that have almost constant communication time, the time complexity reduces to $O(b_z p_z)$, where $b_z \leq n_z$ is the bandwidth of H [19].

Remark 1: Although two matrix-vector products appear in (10), since the matrices have complementary non-zero elements, the computations are the same as for a single gradient calculation. The update requires n_z scalar divisions that are more computationally expensive than multiplications.

A. PQP acceleration by line search

While several unconstrained optimization algorithms compute the solution update by two steps, a descent direction selection, and a step size selection (also called line search), (10) performs the two actions at once in (10). PQP can be seen as a dynamically scaled gradient method by reformulating (10), where for simplicity $\phi = 0$, as

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

hence obtaining $z_{(h+1)} = z_{(h)} - T(z_{(h)})\nabla_z J(z)$. $T(z_{(h)})$ is a diagonal matrix, where $[T(z_{(h)})]_{ii} = \frac{[z_{(h)}]_i}{[H^+ z_{(h)} + F^+]_i}$, that preconditions the gradient, such that feasibility of the updated solution is preserved. Indeed, one degree of freedom (the step-size) is lost, which can cause the algorithm to slow down if the numerators of some variables are very close to zero but the optimal values of the variable are not.

Slowdown can be mitigated by compounding PQP with a gradient descent method that preserves feasibility.

Lemma 1: Let $z_{(h)} \in \mathbb{R}^{n_z}$ be a feasible solution for NNLS (6), and let

$$p_h = (\nabla_z J(z_{(h)}))^-. \tag{11}$$

Then $z_{(h+1)} \in \mathbb{R}^{n_z}$ obtained as

$$\alpha(z_{(h)}) = \begin{cases} -\frac{\nabla_z J(z_{(h)})' p_h}{p_h' H p_h} & \text{if } p_h' H p_h > 0 \\ 0 & \text{otherwise} \end{cases} \tag{12a}$$

$$z_{(h+1)} = z_{(h)} + \alpha(z_{(h)}) p_h, \tag{12b}$$

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

Proof: see [20].

The iteration in Corollary 1 selects the direction of the non-negative components of the anti-gradient, hence maintaining feasibility. If $p_h = 0$ or $\alpha(z_{(h)}) = 0$, $z_{(h+1)} = z_{(h)}$.

Theorem 2: Let $z_{(h)}$ for $h = 0$ be a feasible solution for (6), let ϕ be chosen such that Theorem 1 holds. Then, by alternating iteration (10) and iteration (12) such that after every iteration (12), (10) is executed at least once, converges asymptotically to the optimum z^*.

Proof: see [20].

Line search (12) can be activated in different ways. A simple yet effective strategy is to perform a line search iteration every few PQP iterations. In experimental tests, for weakly convex problems, a rate of $2\% - 10\%$ seems to provide the best results.

B. Application to general quadratic programs

Problem (6) is a subclass of the general (convex) QP (4). We can solve (4) by (10) through duality [4]. The dual problem of (4) is

$$\min_z \quad J_d(Y) = \frac{1}{2} Y' Q_d Y + F_d' Y + \frac{1}{2} M_d \tag{13a}$$

$$\text{s.t.} \quad Y \geq 0, \tag{13b}$$

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 (13) is equal to the number of constraints in (4). In (13), $M_d = F_p' Q_p^{-1} F_p - M_p$ does not affect the optimal solution, but affects the optimum value. Let (4) admit a strictly feasible point. Then strong duality holds, and the optimal solution Y^* of (13) is bounded. From Y^*, the optimal solution of (4) is

$$U^* = \psi_{d2p}(Y^*) = -Q_p^{-1}(F_p + G_p Y^*). \tag{14}$$

Thus, solving (3) through the algorithm alternating iterations (10) and (12) consists of the following steps. At step k, given the current state x_k: (i) formulate (4); (ii) formulate (13); (iii) solve (13); (iv) compute (14); (v) apply $u_k = u^*_{0|k}$ to (1).

Solving (4) via (13) has the drawback that in the MPC problems where there are more constraints than variables ($n_z < n_y$), (13) has more variables than (4), and $Q_d \geq 0$, even if $Q_p > 0$. However, solving the dual allows to enforce the termination conditions in Definition 1 through the duality gap.

Let $Y_{(h)}$, $h \in \mathbb{Z}_{0+}$ be a candidate solution of (13), and compute a candidate primal solution $U_{(h)}$ by (14). Assume $U_{(h)}$, $Y_{(h)}$ are primal and dual feasible, respectively, and let

$$J_p(U_{(h)}) + J_d(Y_{(h)}) \leq \varepsilon_J^a. \tag{15}$$

By duality, $-J_d(Y_{(h)}) \leq J_p(U_{(h)})$, where, if strong duality holds, equality holds at optimum. Indeed,

$$-J_d(Y_{(h)}) \leq -J_d(Y^*) \leq J_p(U^{*)}) \leq J_p(U_{(h)}), \tag{16}$$

and hence (15) implies $J_p(U_{(h)}) - J_p(U^*) \leq \varepsilon_J^a$.

Similarly, if the condition

$$\frac{J_p(U_{(h)}) + J_d(Y_{(h)})}{-J_d(Y_{(h)})} \leq \varepsilon_J^r \quad \text{if} \quad -J_d(Y_{(h)}) > 0 \tag{17a}$$

$$\frac{J_p(U_{(h)}) + J_d(Y_{(h)})}{-J_p(Y_{(h)})} \leq \varepsilon_J^r \quad \text{if} \quad J_p(U_{(h)}) < 0, \tag{17b}$$

holds, (16) guarantees $J_p(U_{(h)}) - J_p(U^*) \leq \varepsilon_J^r |J_p(U^*)|$.

Remark 2: Before computing (15), (17), we have assumed feasibility of $U_{(h)}$ and $Y_{(h)}$. Primal feasibility of $U_{(h)}$ according to (5a) has to be verified before checking (15), (17).

IV. PQPMPC Controller Synthesis

The application of PQP to MPC problem (3) would require at every step the formulation of the dual QP, its solution, and the generation of the primal solution. However, the structure of the QP problem of MPC can be exploited to perform part of the calculations offline, and to synthesize a controller embedding the optimization algorithm. The MPC

problem (3) where the current state is a parameter, can be written in parametric form [9], [21]

$$\min_{U} \quad \frac{1}{2}U'Q_pU + x'C_p'U + \frac{1}{2}x'\Omega_p x \tag{18a}$$
$$\text{s.t.} \quad G_pU \leq S_px + W. \tag{18b}$$

The dual problem of (18) is the parametric QP,

$$\min_{U} \quad \frac{1}{2}Y'Q_dY + x'S_d'Y + W_dY + \frac{1}{2}x'\Omega_d x \tag{19a}$$
$$\text{s.t.} \quad Y \geq 0, \tag{19b}$$

where $Q_d = G_pQ_P^{-1}G_p'$, $S_d = (G_pQ_p^{-1}C_p + S_p)$, $W_d = W_p$, $\Omega_d = C_p'Q_p^{-1}C_p - \Omega_p$. The primal optimizer is computed from the dual optimizer by

$$U(Y^*) = \Psi_{d2p}(x, Y^*) = \Gamma_d x + \Xi_d Y^*, \tag{20}$$

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

The matrices of (19), (20) can be computed beforehand. At every step k, the dual problem (13) is instantiated by substituting $x = x_k$ into (19). For the termination conditions, by substituting (20) into (18b), (5a) results in

$$-S_dx - W_d - Q_dY \leq \max\{\varepsilon_c^r(|S_px + W_p|), \varepsilon_c^a\mathbf{1}\}. \tag{21}$$

Similarly, by substituting (20) into (18a), the primal solution cost is $J_p(\Psi_{d2p}(x, Y)) = \frac{1}{2}(Y'Q_dY - x'\Omega_d x)$. Since, due to the MPC cost (3a), $J_p(U) \geq 0$ the termination condition (5b) results in

$$Y'Q_dY + (x'S_d' + W_d)Y \leq \max\{-\varepsilon_J^r\frac{1}{2}(Y'Q_dY + 2(x'S_d + W_d)'Y + x'\Omega_d x), \varepsilon_J^a\}. \tag{22}$$

where for the duality gap only (17a) is checked, because by (3a), $J_p(U_{(h)}) < 0$ is infeasible. Accordingly the maximization in (22) ignores the relative duality gap in the cases where $-J_d(Y_{(h)}) < 0$.

As a result, the PQP-based MPC controller is synthesized and executed as described in Algorithm 1.

Algorithm 1 PQP-based Model Predictive Control (PQPMPC)

OFFLINE:
Compute Q_p, C_p, Ω_p, G_p
Compute and store W_p, S_p, Q_d, C_d, Ω_d, W_d, S_d, Ξ_d, Γ_d, $Q_d^+ + \phi$, $Q_d^- + \phi$
ONLINE: $k = 0$
loop
  $F_d = S_dx_k + W_d$, $M_d = x_k'\Omega_d x_k$, $\gamma_d = \Gamma_d x_k$, $K_p = S_px_k + W_p$
  $h = 0$, $Y_{(h)} = \bar{Y} > 0$
  repeat
    if LS condition **then**

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

    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 (21) **and** (22)
  $u_k = \begin{bmatrix} I_m & 0 & \dots & 0 \end{bmatrix}(\gamma_d + \Xi_d Y)$
  $k = k + 1$
end loop

V. Numerical Simulations

In this section we discuss the results from two benchmark examples from [22], the control of pitch and angle of attack of a jet aircraft, and the position control of a DC-motor. We compare the algorithm based on alternating iterations (10) and (12) in Matlab M-code (PQPM) and C-mex (PQPMEX) and Algorithm 1 (PQPMPC) with: (i) the `QUADPROG` routine (QPROG) of the Matlab Optimization Toolbox 5.1(part M-code, part C-mex) that for medium size problems implements an active set method; (ii) a C-mex implementation of Dantzig's active set algorithm (DANTZ) [23]; (iii), the QP solver (C-mex) in the commercial NAG toolbox for Matlab (NAG) [24], which is an inertia-controlling active set method exploiting Cholesky factorization. PQP-based solvers generate and solve the dual problem, while the others solve the primal problem. For all solvers, termination conditions have the same numerical precision. We have also implemented the fast gradient algorithm GPAD in [13] in Matlab M-code (GPADM) as described in [13, Sec. 4]. The data reported here is obtained by using as Lipschitz constant the Frobenious norm of the Hessian ([13, Sec. 5]), but we did not notice any significant difference by using the maximum eigenvalue of the Hessian instead. For GPAD, we have used the same termination conditions (5) for both, even though in [13] only absolute tolerances are used.

The simulations are executed in Matlab 2010b, in a MacBook-Pro with Intel i7 processor 2.8GHz, and 8GB RAM. We report the time to execute the main function of each solver on a single core.

A. Aircraft control

We consider the control of pitch and angle of attack of a jet aircraft. The linearized dynamics (1) is an unstable fourth order systems with elevator and flaperon angles as control inputs. The objective is to track references on pitch angle and angle of attack. The constraints (2) enforce upper and lower bounds on elevator, flaperon, pitch angles and angle of attack. We design an LQR controller that stabilizes the unconstrained plant, and let MPC control the resulting closed-loop system. Thus, the command to elevators and flaperons is $v_k = \mathcal{K}_{\text{LQR}}x_k + u_k^{\text{MPC}}$ where $\mathcal{K}_{\text{LQR}}$ is the LQR gain and u_k^{MPC} is the MPC command at step k. Because of this, the constraints on elevators and flaperons become state-input constraints. The sampling period is $T_s = 50$ms, and the prediction horizon is $N = 5$ steps, resulting in a

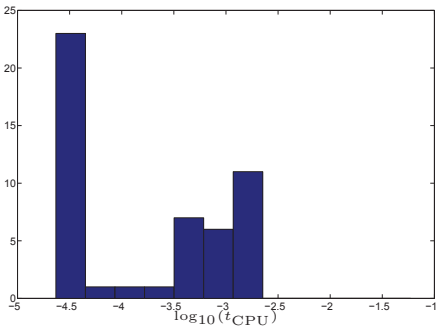

Fig. 1: Distribution of PQPMPC computation time in the aircraft example.

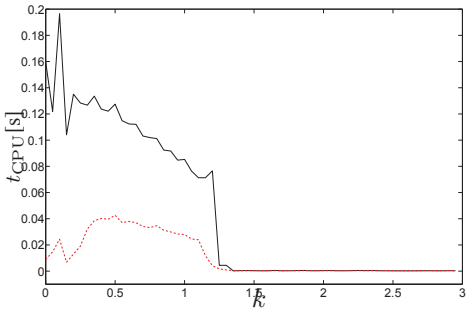

Fig. 2: Computation time for GPADM (solid) and PQPM (dash) in the aircraft example.

parametric QP with $n_q = 32$ constraints, and $n_u = 10$ variables. To assess the complexity, the optimization problem was solved explicitly [9] resulting in more than 1700 regions which, for 8B double precision variables, requires more than 1.25MB storage memory. Such an amount is prohibitive for many embedded control systems [2], [11]. The total memory required by the PQPMPC controller, without any optimization, is approximately 50KB.

Solver	Avg[ms]	Min[ms]	Max[ms]
GPADM:	46.567	0.222	196.481
PQPM:	11.515	0.319	42.618
QUADPROG:	2.954	1.462	7.948
QPACT:	0.597	0.445	0.986
NAG:	0.917	0.615	1.387
PQPMEX:	0.937	0.069	3.553
PQPMPC:	0.444	0.032	1.985

TABLE I: Computation time for the aircraft example.

The average, minimum, and maximum computation time for solving the QP along a simulation of 50 steps, where several constraints are active, are reported in Table I. PQPMEX algorithm is in line with the fastest algorithms, although slightly slower than some. The PQPMPC controller is the fastest approach, due to moving offline several matrix calculations. It shall be noted that the PQP algorithms are much simpler than the ones they are compared with, with the exception of GPAD. The variability between minimum and maximum computing time is due to the cases when the constraints are not active (PQP converges extremely fast to the solution), versus the ones where many constraints are active. The distribution of the computation time of the PQPMPC controller is shown in Figure 1, where one can clearly see the two peaks. The execution time for PQP and the dual fast gradient algorithm GPAD during the simulation are compared in Figure 2. In the initial part of the simulation, when several constraints are active, PQP is faster. In the final part of the simulation, when the constraints are all inactive, GPAD is sometimes slightly faster because it terminates in a single iteration [13], while that is not necessarily the case for PQP.

B. DC-motor position control

We consider a 4^{th} order linear system (1) modeling a voltage controlled DC motor connected by a flexible shaft to a load. The control objective is to track a time varying load angle position reference signal $r_\theta(t)$. The constraints (2) enforce upper and lower bounds on input voltage and motor torque. We design an MPC controller where the control input is the step-to-step voltage variation, with sampling period $T_s = 100$ms, prediction horizon $N = 10$, resulting in a parametric QP with $n_q = 40$ constraints, and $n_u = 10$ variables. The explicit solution [9] of this problem results in more than 4000 regions, and more than 2MB memory, which is often prohibitive. The total memory required by the PQPMPC controller, without any optimization, is approximately 60KB. The computation times for a simulation of 200 steps are reported in Table II, for a "mild" reference that causes few constraints to be active.

Solver	Avg[ms]	Min[ms]	Max[ms]
GPADM:	9.481	0.268	43.356
PQPM:	1.418	0.367	8.891
QUADPROG:	1.847	1.406	4.082
QPACT:	0.571	0.452	0.896
NAG:	0.865	0.599	1.637
PQPMEX:	0.178	0.077	0.987
PQPMPC:	0.084	0.040	0.491

TABLE II: Computation time for the DC-motor example ("mild" reference).

The computation times for the case of an "aggressive" reference that causes several constraints to be active at every step are reported in Table III. The results show that PQP is more sensitive than other solvers to the activation of constraints, but still converges rapidly. For the case in

Solver	Avg[ms]	Min[ms]	Max[ms]
GPADM:	38.896	0.270	127.560
PQPM:	8.771	0.371	68.309
QUADPROG:	2.363	1.489	7.551
QPACT:	0.644	0.451	2.525
NAG:	0.852	0.371	1.669
PQPMEX:	0.727	0.078	6.469
PQPMPC:	0.367	0.040	3.037

TABLE III: Computation time for the DC-motor example ("aggressive" reference).

Table III , the impact of the acceleration technique described in Section III-A executed once every 20 PQP iterations is shown in Figure 3, in terms of number of iterations executed by the algorithm during the simulation. The acceleration is particularly effective in reducing the peaks of 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.

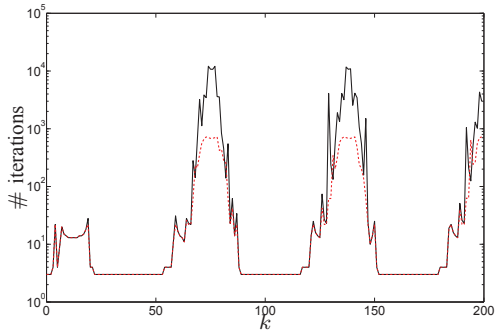

Fig. 3: Effect of the acceleration technique on the DC-motor example reported in Table III. Number of iterations without acceleration (solid) and with acceleration every 20 iterations (dash) during the simulation.

For this example we have also studied how the computation time changes while increasing the problem size. For the "aggressive" reference signal, (Table III), we compare the computing time results of PQPMPC and NAG, for $N = 10$, $N = 25$, $N = 40$, and $N = 80$. The results in Table IV show

Solver	Avg[ms]	Min[ms]	Max[ms]
NAG-10:	0.852	0.371	1.669
NAG-25:	3.587	1.051	8.212
NAG-40:	7.198	1.602	16.226
NAG-80:	34.764	3.357	81.130
PQPMPC-10:	0.367	0.040	3.037
PQPMPC-25:	0.424	0.157	3.704
PQPMPC-40:	0.941	0.377	6.481
PQPMPC-80:	5.058	3.939	10.228

TABLE IV: Computation time for different problem sizes for the DC-motor example ("aggressive" reference).

that PQPMPC maintains a margin over the NAG solver.

VI. Conclusions

We have discussed how the PQP algorithm can be effectively applied to linear MPC. We have introduced termination conditions that guarantee the desired degree of suboptimality, and an acceleration based on projection-free line search. We have also shown how a PQP-based MPC controllers can be synthesized by pre-computing and pre-allocating several matrices. The algorithms were compared with free and commercial solvers showing interesting performance. Ongoing work involves improving the acceleration, developing warm-starting, and deriving bounds on the number of iterations.

References

[1] 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.

[2] 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.

[3] 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.

[4] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[5] J. Nocedal and S. Wright, *Numerical optimization*. Springer verlag, 1999.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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*, vol. 21, no. 4, pp. 1236–1248, 2013.

[11] S. Di Cairano, W. Liang, I. V. Kolmanovsky, M. L. Kuang, and A. M. Phillips, "Power smoothing energy management and its application to a series hybrid powertrain," *IEEE Tr. Contr. Sys. Technology*, 2013, to appear, available on `ieeexplore.org`.

[12] 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.

[13] 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.

[14] M. Kögel and R. Findeisen, "Fast predictive control of linear systems combining Nesterovs gradient method and the method of multipliers," in *Proc. 51st IEEE Conf. on Dec. and Control*, Orlando, FL, 2011, pp. 501–506.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] M. Brand and D. Chen, "Parallel quadratic programming for image processing," in *Proc. 18th IEEE Int. Conf. Image Processing*, 2011, pp. 2261–2264.

[20] S. Di Cairano, M. Brand, and S. Bortoff, "Projection-free parallel quadratic programming for linear model predictive control," *Int. J. Control*, 2013, in press, available online at www.tandfonline.com.

[21] 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.

[22] A. Bemporad, *Hybrid Toolbox – User's Guide*, Dec. 2003.

[23] 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.

[24] NAG Ltd., *NAG Toolbox for MATLAB*, Oxford, U.K., 2011, http://www.nag.co.uk.