A Numerically Stable Solver for Positive Semidefinite Quadratic Programs Based on Nonnegative Least Squares

Alberto Bemporad, *Fellow, IEEE*

***Abstract*—This paper proposes a new algorithm for solving convex quadratic programs (QP) subject to linear inequality and equality constraints. The method extends an approach recently proposed by the author for solving strictly convex QP's using nonnegative least squares, by making it numerically more robust and able to handle also the nonstrictly convex case, equality constraints, and warm starting from an initial guess. Robustness is achieved by introducing an outer proximal-point iteration scheme that regularizes the problem without altering the solution, and by adaptively weighting the least squares problems encountered while solving the problem. The performance of the resulting QP solver in terms of speed and robustness in the single precision arithmetic is assessed against other optimization algorithms tailored to embedded model predictive control applications.**

***Index Terms*—Active-set methods, model predictive control (MPC), nonnegative least squares, proximal-point method, quadratic programming.**

I. Introduction

The ambition of making optimization-based control a completely viable technology in embedded system applications has stimulated a quest in recent years for fast, simple, and numerically robust solvers for convex quadratic programs (QPs) of the form

$$\begin{aligned} \min_z \quad & \frac{1}{2}z'Qz + c'z \\ \text{s.t.} \quad & Az \le b \\ & Gz = g \end{aligned} \tag{1}$$

where $Q \succeq 0$ is the Hessian matrix, $Q \in \mathbb{R}^{n\times n}$ and "$\succeq$" denote positive semidefiniteness, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $G \in \mathbb{R}^{p\times n}$, $g \in \mathbb{R}^p$, and $\mathbb{R}$ are the set of real numbers.

Many good algorithms and packages for QPs are now available, such as active-set methods [1, Sec. 24-4], [2]–[4], interior-point methods [5], [6], gradient projection methods [7], [8], and the alternating directions method of multipliers (ADMM) [9], [10].

An active-set algorithm for strictly convex QPs was recently proposed in [11], based on the idea of recasting the QP into a nonnegative least-squares (NNLS) problem. A few solution methods exist for the NNLS problem, including active-set [12] and block-pivoting methods [13], and interior-point methods [14]. While the QP solver based on NNLS of [11] is simple to code, fast to execute, provides the solution in a finite number of steps, and is amenable for solving mixed-integer quadratic programs (MIQPs) [15], it is limited to strictly convex QPs, with numerical weakness increasing with the condition number of the primal Hessian matrix Q. Indeed, many solvers require $Q \succ 0$ in (1), and exhibit a numerical behavior that deteriorates when the condition number cond(Q) increases. Regularizing (1) by replacing Q with $Q + \epsilon I$, $\epsilon > 0$, can make the problem more tractable numerically but results in suboptimal solutions: The larger ϵ, the better conditioned is the problem, but the less optimal is the solution, so finding a good tradeoff may be tricky.

This paper largely extends the approach in [11] to make it numerically more robust with respect to finite-precision arithmetics, even handling also the nonstrictly convex case. The main idea is to use an outer proximal-point scheme that iteratively refines the solution of a QP problem solved via a numerically-robustified active-set NNLS algorithm. The resulting method also handles equality constraints and warm starting, features that are both particularly useful when solving MIQPs, as previously presented in the conference paper [15].

Manuscript received November 16, 2016; revised November 16, 2016 and April 13, 2017; accepted July 5, 2017. Date of publication August 3, 2017; date of current version January 26, 2018. Recommended by Associate Editor Wim Michiels.

The author is with the IMT Institute for Advanced Studies Lucca, Lucca 55100, Italy (e-mail: alberto.bemporad@imtlucca.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2017.2735938

II. Outer Proximal-Point Iterations

Problem (1) is a special case of the convex optimization problem

$$\min_z \quad f(z) \tag{2}$$

where $z \in \mathbb{R}^n$, and $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is the closed, proper, and convex function

$$f(z) = \frac{1}{2}z'Qz + c'z + \sum_{i=1}^{m} g(A_i z - b_i) + \sum_{i=1}^{p} \delta(G_i z - g_i). \tag{3}$$

In (3), $g : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is the indicator function of the negative axis (for $x \in \mathbb{R}$, $g(x) = 0$ if $x \le 0$, or $+\infty$ otherwise), $\delta : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is the indicator function of the origin ($\delta(x) = 0$, if $x = 0$, or $+\infty$ otherwise), and A_i (G_i), b_i (g_i) denote the ith row of A (G) and component of b (g), respectively.

Lemma 1: Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be closed, proper, and convex. Assume that the set of minimizers of (2) is a nonempty subset of $\mathbb{R}^n$. Then for any matrix $E = E' \succ 0$, $E \in \mathbb{R}^{n\times n}$, the iterations

$$z_{k+1} = \arg\min_z \quad f(z) + \frac{1}{2}(z - z_k)'E(z - z_k) \tag{4}$$

converge to an optimizer z^* of (2).

Proof: Let $E = LDL'$ be a decomposition of E, with $D \in \mathbb{R}^{n\times n}$ diagonal and $L \in \mathbb{R}^{n\times n}$ unit lower triangular, $D_{ii} > 0, \forall i = 1, \ldots, n$. The iterations in (4) are equivalent to the following proximal-point algorithm iterations

$$y_{k+1} = \arg\min_y \quad \left\{ h(y) + \frac{1}{2}\|y - y_k\|^2 \right\} = \text{prox}_h(y_k)$$

where $y = T^{-1}z$, $h(y) = f(Ty)$, and $T = L^{-1}D^{-\frac{1}{2}}$ is a nonsingular real matrix. Since f is convex, h is also convex, being the composition of a convex and affine function [16]. Hence, $y_{k+1} = \text{prox}_h(y_k)$ converges to a minimizer y^* of $f(L^{-T}D^{-\frac{1}{2}}y)$ as $k \to \infty$ [17, Th. 27.1], and, therefore, z_k converges to the minimizer Ty^* of (2). ■

Note that Lemma 1 holds in particular for $E = \epsilon I$, $\forall \epsilon > 0$. In this case (4) simplifies to

$$z_{k+1} = \arg\ \min_x \quad f(x) + \frac{1}{2\gamma_k}\|z - z_k\|_2^2 = \text{prox}_{\gamma_k f}(z_k) \qquad (5)$$

where $\gamma_k \equiv \frac{1}{\epsilon}, \forall k \geq 0$. By [17, Th. 27.1], the iterations in (5) converge to an optimizer z^* of f for any sequence $\{\gamma_k\}_{k=0}^{\infty}$ such that $\gamma_k > 0$, $\sum_{k=0}^{\infty} \gamma_k = +\infty$.

Corollary 1: Let $Q = Q' \succeq 0$ in (1). For any $\epsilon > 0$, the sequence $\{z_k\}$ of solutions to the following strictly convex QPs

$$\begin{aligned} z_{k+1} = \arg\ \min_z & \frac{1}{2} z'Qz + c'z + \frac{\epsilon}{2}\|z - z_k\|_2^2 \\ \text{s.t.} \quad & Az \leq b \\ & Gz = g \end{aligned} \qquad (6)$$

converges to a solution z^* of (1) as k tends to infinity.

Proof: The iterations in (6) are equivalent to the sequence of convex nonsmooth problems

$$\begin{aligned} z_{k+1} = \arg\ \min_z & \frac{1}{2} z'Qz + c'z + \sum_{i=1}^{m} g(A_i z - b_i) \\ & + \sum_{i=1}^{p} \delta(G_i z - g_i) + \frac{\epsilon}{2}\|z - z_k\|_2^2. \end{aligned} \qquad (7)$$

By Lemma 1, Problem (7) converges to the solution of (3), and therefore of (1). ■

The iterations in (6) can be solved by any QP solver, even by those limited to strictly positive definite Q since $\epsilon > 0$. Note that (6) can be warm started at each iteration from the previous solution z_k, where z_0 is an arbitrary initial guess.

The iterations in (6) are equivalent to

$$z_{k+1} = \text{prox}_{\gamma f}(z_k) \qquad (8)$$

where f is defined in (3) and $\gamma = \frac{1}{\epsilon}$. Clearly, the smaller γ the slower the iterations defined by (8) will converge to an optimizer z^* of (1). Therefore, there is a tradeoff between robustness in solving (6) (large ϵ, small γ) and speed of convergence of (8) (large γ, small ϵ). Note that, in the absence of constraints, the iterations in (6) corresponds to solving the symmetric linear system $Qz = -c$ by the "iterative refinements" method $(Q + \epsilon I)z_{k+1} = -c + \epsilon z_k$, or equivalently $(Q + \epsilon I)\zeta_k = -c - Qz_k$, $z_{k+1} = z_k + \zeta_k$, see [18, Sec. 4.1.2].

III. Inner Active-Set Solver

A solution approach was proposed in [11] to minimize strictly convex quadratic functions subject to inequality constraints. The approach is based on the idea of rephrasing the QP as a least distance problem (LDP) that is solved via a NNLS algorithm. The method was proved very efficient compared to existing state-of-the-art QP algorithms. However, its numerical robustness quickly deteriorates with increasing cond(Q) and does not handle the case of Q singular. Moreover, the original formulation in [11] suffers from numerical issues in case the right-hand sides b, g of the constraints and/or the linear term c of the cost function have large entries compared to the corresponding entries in the left-hand-sides A, G. In this paper, we extend such a method to handle equality constraints, to make it more robust w.r.t. large entries in b, g, c, and to exploit warm starts from the previous iterate z_k to solve (6) more efficiently, so that it can be used to solve the proximal-point iterations (8).

The dual of the QP problem (6) is the following convex QP

$$\begin{aligned} \min_{\lambda,\mu} \quad & \frac{1}{2}\begin{bmatrix}\lambda\\ \mu\end{bmatrix}' \begin{bmatrix}A\\ G\end{bmatrix} (Q + \epsilon I)^{-1} \begin{bmatrix}A' & G'\end{bmatrix} \begin{bmatrix}\lambda\\ \mu\end{bmatrix} + \begin{bmatrix}d^k\\ f^k\end{bmatrix}' \begin{bmatrix}\lambda\\ \mu\end{bmatrix} \\ \text{s.t.} \quad & \lambda \geq 0,\ \mu \text{ free} \end{aligned} \qquad (9)$$

where $\lambda \in \mathbb{R}^m$, $\mu \in \mathbb{R}^p$, and $d^k \in \mathbb{R}^m$, $f^k \in \mathbb{R}^p$ are defined as

$$\begin{bmatrix}d^k\\ f^k\end{bmatrix} \triangleq \begin{bmatrix}b\\ g\end{bmatrix} + \begin{bmatrix}A\\ G\end{bmatrix} (Q + \epsilon I)^{-1}(c - \epsilon z_k).$$

The following theorem shows how the QP problem (6) is equivalent to a least-squares problem in which some of the variables are constrained to be nonnegative, extending [11, Th. 1] to the case of equality constraints as in [15] and to scaling of d^k, f^k for improved numerical robustness.

Theorem 1: Consider the QP problem (6) with $Q = Q' \succeq 0$, and let $\epsilon > 0$. Let $L'L$ be a Cholesky factorization of $Q + \epsilon I$ and define $M \triangleq AL^{-1}$, $N \triangleq GL^{-1}$. Let γ, β be any positive scalars. Consider the Partially Nonnegative Least Squares (PNNLS) problem

$$\begin{aligned} \min_{y,\nu} \quad & \frac{1}{2}\left\| \begin{bmatrix}M'\\ \beta(d^k)'\end{bmatrix} y + \begin{bmatrix}N'\\ \beta(f^k)'\end{bmatrix} \nu + \begin{bmatrix}0\\ \beta\gamma\end{bmatrix} \right\|_2^2 \\ \text{s.t.} \quad & y \geq 0,\ \nu \text{ free} \end{aligned} \qquad (10)$$

with $y \in \mathbb{R}^m$, $\nu \in \mathbb{R}^p$, and let

$$\delta^* \triangleq \beta(\gamma + (d^k)'y^* + (f^k)'\nu^*) \qquad (11a)$$

$$r^* \triangleq \begin{bmatrix}-M'y^* - N'\nu^*\\ -\delta^*\end{bmatrix} \qquad (11b)$$

where $r^* \in \mathbb{R}^{n+1}$ is the residual obtained at the optimal solution (y^*, ν^*) of (10), $y^* \in \mathbb{R}^m$, and $\nu^* \in \mathbb{R}^p$. The following statements hold:

1) if $r^* = 0$, then QP (1) is infeasible;
2) If $r^* \neq 0$, then

$$z^* \triangleq -(Q + \epsilon I)^{-1}\left(c - \epsilon z_k + \frac{A'y^* + G'\nu^*}{\beta\delta^*}\right) \qquad (12)$$

and

$$\lambda^* \triangleq \frac{1}{\beta\delta^*} y^*,\ \mu^* \triangleq \frac{1}{\beta\delta^*}\nu^*$$

solve QP (6) and its dual (9), respectively.

Proof: See Appendix. ■

The scalar β in (10) allows weighting the terms $\|M'y + N'\nu\|$ and $\|(d^k)'y + (f^k)'\nu + \gamma\|$ differently. Contrary to the original approach of [11], in which $\beta = 1$, introducing such a parameter β largely contributes to improving the numerical robustness of the QP solution method. We will detail how to choose β and γ in Section III-B.

Next Lemma 2 characterizes the relation between the dual variable w of problem (10) and the feasibility of z with respect to the inequality constraints in (6).

Lemma 2: Let $y \in \mathbb{R}^m, \nu \in \mathbb{R}^p$ be such that $(d^k)'y + (f^k)'\nu \neq -\gamma$ and

$$-\begin{bmatrix}N & \beta f^k\end{bmatrix} \begin{bmatrix}-M'y^* - N'\nu^*\\ -\beta((d^k)'y^* + (f^k)'\nu^* + \gamma)\end{bmatrix} = 0.$$

Let $z \in \mathbb{R}^n$ and $w \in \mathbb{R}^{\mathsf{m}}$ be defined as

$$z = -\frac{1}{\beta^2(\gamma + (d^k)'y + (f^k)'\nu)} L^{-1}(M'y + N'\nu) - (Q+\epsilon I)^{-1}(c - \epsilon z_k) \tag{13a}$$

$$w = M(M'y + N'\nu) + \beta^2(\gamma + (d^k)'y + f_k'\nu)d^k. \tag{13b}$$

Then, for all $i \in \{1, \ldots, \mathsf{m}\}$ and $\sigma \geq 0$, if

$$w_i \geq -\beta^2(\gamma + (d^k)'y + (f^k)'\nu)\sigma \tag{14a}$$

then

$$A_i z - b_i \leq \sigma,\ Gz = g. \tag{14b}$$

Proof: The fact that condition (14a) implies (14b) simply follows by left multiplying z as defined in (13a) by A, subtracting b, and by recalling the definition of matrices M and N. ■

A. Infeasibility Detection

The following Corollary 2 of Theorem 1 provides a further criterion to detect infeasibility of the QP problem (1) even without solving (10) to optimality.

Corollary 2: Let $\mathcal{P} \subseteq \{1, \ldots, \mathsf{m}\}$ be a subset of constraint indices and let $A_{\mathcal{P}}$ denote the submatrix obtained by collecting the rows indexed by $\mathcal{P}$ of a matrix A. For any $\mathcal{P} \subseteq \{1, \ldots, q\}$ and $\gamma > 0$, if the solution $s_{\mathcal{P}}$ of the least squares (LS) problem

$$s_{\mathcal{P}} = \arg\min_{z_{\mathcal{P}}} \left\| \begin{bmatrix} M'_{\mathcal{P}} & N' \\ \beta(d^k_{\mathcal{P}})' & \beta(f^k)' \end{bmatrix} z_{\mathcal{P}} + \begin{bmatrix} 0 \\ \beta\gamma \end{bmatrix} \right\|_2^2 \tag{15}$$

is such that $s_{\mathcal{P}} \geq 0$ and the squared residual is zero, then the QP problem (1) is infeasible.

Proof: As proved in part i) of Theorem 1, if the residual in (15) is zero then the polyhedron $\mathcal{C} \triangleq \{u \in \mathbb{R}^n : M_{\mathcal{P}}u \leq d^k_{\mathcal{P}},\ Nu = f^k\}$ is empty. Hence, also the polyhedron $\{u \in \mathbb{R}^n : Mu \leq d^k,\ Nu = f^k\} = \mathcal{C} \cap \{u \in \mathbb{R}^n : M_{\{1,\ldots,\mathsf{m}\}\setminus\mathcal{P}}u \leq d^k_{\{1,\ldots,\mathsf{m}\}\setminus\mathcal{P}}\}$ is empty, and therefore problem (1) is infeasible. ■

B. Parameter Selection

We choose the parameters $\beta, \gamma > 0$ as

$$\gamma = 1 + \|f^k\|_1 + \|d^k_{\mathcal{P}}\|_1,\ \beta = \frac{1}{\gamma} \tag{16}$$

which, according to numerical experience, provides very good robustness properties.

In addition, after computing M, N, one may prescale them by changing the constraints $A_i z \leq b_i, G_i z = g_i$ to

$$\frac{1}{\|M_i\|}A_i z \leq \frac{1}{\|M_i\|}b_i,\ \frac{1}{\|N_i\|}G_i z = \frac{1}{\|N_i\|}g_i \tag{17}$$

respectively. However, contrary to first-order methods that are very sensitive to scaling, our simulation tests did not show significant enough improvements that justify the extra computations for scaling the problem matrices as in (17).

C. Recursive LDL^T Updates

Solving each PNNLS problem (10) efficiently at a given proximal-point iteration k using the algorithm in [12] and [11] requires solving a sequence of LS problems of the form (15). This corresponds to finding a solution of the symmetric linear system

$$(W_{\mathcal{P}}W'_{\mathcal{P}} + \beta^2\theta^k_{\mathcal{P}}(\theta^k_{\mathcal{P}})')s_{\mathcal{P}} = -\beta^2\gamma\theta^k_{\mathcal{P}} \tag{18}$$

where $W_{\mathcal{P}} \triangleq \left[\begin{smallmatrix} M_{\mathcal{P}} \\ N \end{smallmatrix}\right]$ and $\theta^k_{\mathcal{P}} \triangleq \left[\begin{smallmatrix} d^k_{\mathcal{P}} \\ f^k \end{smallmatrix}\right]$. As shown in [11] and [12], the active set $\mathcal{P}$ changes from one iteration to another by adding or removing one index in the set. The symmetric linear system (18) can be solved by several methods such as conjugate gradient and batch or recursive *LDL*^{T}, QR, or Cholesky factorizations. This was analyzed in [11, Fig. 1b], where it is apparent that the recursive *LDL*^{T} factorization

$$LDL' = \begin{bmatrix} W_{\mathcal{P}} & \beta\theta^k_{\mathcal{P}} \end{bmatrix} \begin{bmatrix} W'_{\mathcal{P}} \\ \beta(\theta^k_{\mathcal{P}})' \end{bmatrix} \tag{19}$$

of the left-hand-side matrix of (18) is the fastest method. Therefore, in this paper, we only focus on such an approach and provide ways to recursively update (19) with respect to the following:

1) changes in the active set $\mathcal{P}$ (adding/removing an index);
2) changes of the scalar β during PNNLS iterations;
3) changes in the definition of vectors d^k, f^k between different proximal-point iterations.

A recursive QR approach would be also possible, but would be computationally more expensive, although more robust; however, such an extra robustness is not really required here because we can choose ϵ to make matrix $Q + \epsilon I$ arbitrarily well conditioned.

For recursive *LDL*^{T} updates due to changes in the active set $\mathcal{P}$, we recall the following results.

Result 1 (Increasing $\mathcal{P}$, see [11, Th. 2]): Let $A \in \mathbb{R}^{h\times n}$. Then, there exists a unit lower triangular matrix $L^h \in \mathbb{R}^{h\times h}$ and a diagonal matrix $D^h \in \mathbb{R}^{h\times h}$ with $D_{ii} \geq 0$, satisfying the following recursions

$$L^1 = 1,\ D^1 = \|A_1\|_2^2,\ L^{k+1} = \begin{bmatrix} L^k & 0 \\ (\ell^{k+1})' & 1 \end{bmatrix}$$

$$D^{k+1} = \begin{bmatrix} D^k & 0 \\ 0 & d^{k+1} \end{bmatrix} \quad (L^k D^k)\ell^{k+1} = A_{1:k}A'_{k+1}$$

$$d_{k+1} = \|A_{k+1}\|_2^2 - (\ell^{k+1})'D^k\ell^{k+1} \tag{20}$$

for $k = 1, \ldots, h-1$, and such that

$$A_{1:k}A'_{1:k} = L^k D^k (L^k)' \quad \forall k = 1, \ldots, h$$

where $A_{1:k}$ denotes the matrix formed by the first k rows of A and A_i the ith row of A.

Result 2 (Decreasing $\mathcal{P}$, see [11, Lemma 2]): Let

$$LDL' = \begin{bmatrix} \mathcal{A}_1 \\ a' \\ \mathcal{A}_2 \end{bmatrix} \begin{bmatrix} \mathcal{A}'_1 & a & \mathcal{A}'_2 \end{bmatrix}$$

with

$$L = \begin{bmatrix} L_1 & 0 & 0 \\ \ell & 1 & 0 \\ L_3 & L_4 & L_2 \end{bmatrix},\ D = \begin{bmatrix} D_1 & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & D_2 \end{bmatrix}$$

$L \in \mathbb{R}^{h\times h}$ unit lower triangular, $D \in \mathbb{R}^{h\times h}$ diagonal, $D_{ii} \geq 0$, be a given *LDL*^{T} decomposition. Then, $\forall i = 1, \ldots, h$ an *LDL*^{T} decomposition

$$L_- D_- L'_- = \begin{bmatrix} \mathcal{A}_1 \\ \mathcal{A}_2 \end{bmatrix} \begin{bmatrix} \mathcal{A}'_1 & \mathcal{A}'_2 \end{bmatrix}$$

with $L_- \in \mathbb{R}^{(p-1)\times(h-1)}$ unit lower triangular and $D_- \in \mathbb{R}^{(h-1)\times(h-1)}$ diagonal, $D_{-ii} \geq 0$, $\forall i = 1, \ldots, h-1$, is given by

$$L_- = \begin{bmatrix} L_1 & 0 \\ L_3 & \bar{L}_2 \end{bmatrix},\ D_- = \begin{bmatrix} D_1 & 0 \\ 0 & \bar{D}_2 \end{bmatrix} \tag{21}$$

where $\bar{L}_2, \bar{D}_2$ provide the LDL^T decomposition $\bar{L}_2\bar{D}_2\bar{L}_2' = \delta L_4 L_4' + L_2 D_2 L_2'$, with $\bar{L}_2$ unit lower triangular and $\bar{D}_2$ diagonal.

Regarding recursive modifications of the LDL^T factorizations due to either changes in β or in d^k, f^k, we adopt the result of [19, Algorithm C1] for rank-one updates of LDL^T factorizations. In particular, we denote by

$$[L_1, D_1] = \mathsf{rankone}\,(L, D, \alpha, z) \tag{22}$$

the procedure to obtain, given a factorization LDL', the new factorization $L_1 D_1 L_1' = LDL' + \alpha zz'$, where $\alpha \in \mathbb{R}$ and z a vector of proper dimensions. The complexity of (22) is $n^2 + O(n)$ multiplications and $n^2 + O(n)$ additions, where n is the dimension of z.

We use (22) in two different ways. First, given the factorization $L_\mathcal{P} D_\mathcal{P} L_\mathcal{P}' = W_\mathcal{P} W_\mathcal{P}' + \bar{\beta}^2 \theta_\mathcal{P}^k (\theta_\mathcal{P}^k)'$, solving (18) for a new value β requires the following rank-one update

$$[L_\mathcal{P}^1, D_\mathcal{P}^1] = \mathsf{rankone}\,(L_\mathcal{P}, D_\mathcal{P}, \beta^2 - \bar{\beta}^2, \theta_\mathcal{P}^k). \tag{23}$$

Second, when warm starting the PPNLS problem equivalent to (6) at the optimal set $\mathcal{P}$ active at z_k, we perform the following two rank-one updates

$$\left[L_\mathcal{P}^1, D_\mathcal{P}^1\right] = \mathsf{rankone}\,(L_\mathcal{P}, D_\mathcal{P}, -1, \beta_{k-1} w_\mathcal{P}^{k-1}) \tag{24a}$$

$$\left[L_\mathcal{P}^2, D_\mathcal{P}^2\right] = \mathsf{rankone}\,(L_\mathcal{P}^1, D_\mathcal{P}^1, 1, \beta_k \theta_\mathcal{P}^k) \tag{24b}$$

where

$$w_\mathcal{P}^k = \begin{bmatrix} b_\mathcal{P} \\ g \end{bmatrix} + \begin{bmatrix} M_\mathcal{P} \\ N \end{bmatrix} L^{-T}(c - \epsilon z_k)$$

and β_{k-1}, β_k are the values for β chosen at the end of the PNNLS iterations and at the beginning of next PNNLS iterations, respectively. Moreover, since the value of β in (16) changes with the active set $\mathcal{P}$, the extra rank-one update (23) is required at each iteration for updating the LDL^T factorization.

Note that performing one or two rank-one updates has an overall complexity of $O(\ell^2)$, where $\ell = p + \#\mathcal{P}$ and $\#\mathcal{P}$ is the cardinality of $\mathcal{P}$, whereas the complexity of a new LDL^T factorization from scratch would be $O(\ell^3)$.

D. Warm Starting

When solving the QP problem (6), for efficiency one should warm start from the previous solution z_k, which becomes increasingly closer to the optimum as k increases. It is, therefore, important to use a QP solver that supports warm starting. When the equivalent PNNLS problem (10) is solved by the active-set method of [12], a complementary basic solution satisfying $y_\mathcal{P} \geq 0$ is required to warm start the algorithm for a given combination $\mathcal{P}$ of active constraints at z_k. This amounts to satisfying the following conditions:

$$w = M(M'y + N'\nu) + \beta^2(\gamma + y'd^k + \nu' f^k)d^k \tag{25a}$$

$$y'w = 0,\ y_\mathcal{P} \geq 0,\ w_\mathcal{P} = 0,\ \ y_{\{1,\ldots,m\}\setminus\mathcal{P}} = 0 \tag{25b}$$

$$\nu = -\begin{bmatrix} N' \\ \beta(f^k)' \end{bmatrix}^{\#} \left(\begin{bmatrix} M' \\ \beta(d^k)' \end{bmatrix} y + \begin{bmatrix} 0 \\ \beta\gamma \end{bmatrix} \right) \tag{25c}$$

where $^{\#}$ denotes pseudoinversion. Let $\mathcal{P}_{k-1}$ be the optimal active set of the PNNLS problem (10) at the previous step $k-1$. An initial condition (y_0^k, ν_0^k, w_0^k), an active set $\mathcal{P}_0 \subseteq \mathcal{P}$, and parameters β_0, γ_0 satisfying (16) for the new problem at step k can be obtained by executing the procedure get_feasible described in Steps 19.1-19.7 of Algorithm 1 (that is "Loop B" in [12, Algorithm (23.10)]), followed by (25a) (cf., Step 8). Starting from vectors $y_0^k = y^{k-1} \geq 0$, such a procedure replaces y with $y + \alpha(s - y)$, where s is the solution of the LS problem (15) for the current $\mathcal{P}$, γ, β, and α is the largest scalar,

Algorithm 1: Robust QP Solver Based on NNLS and Proximal-Point Iterations.

Input: QP matrices Q, c, A, b, G, g, regularization term $\epsilon > 0$, feasibility tolerance $\sigma > 0$, stop tolerance $\eta > 0$, initial guess z_0.

1. Compute inverse Cholesky factor $\mathcal{L}^{-1}$, $\mathcal{L}'\mathcal{L} = (Q + \epsilon I)$;
2. $M \leftarrow A\mathcal{L}^{-1}$, $N \leftarrow G\mathcal{L}^{-1}$; $\mathcal{P} \leftarrow \emptyset$;
3. Compute the LDL^T factorization $LDL' = NN'$;
4. $k, h \leftarrow 0$;
5. $v_k \leftarrow \mathcal{L}^{-T}(c - \epsilon z_k)$; $W_\mathcal{P} \leftarrow \begin{bmatrix} M_\mathcal{P} \\ N \end{bmatrix}$; $d^k \leftarrow b + Mv_k$; $\theta_\mathcal{P}^k \leftarrow \begin{bmatrix} d_\mathcal{P}^k \\ g + Nv_k \end{bmatrix}$; $\gamma \leftarrow \|\theta_\mathcal{P}^k\|_1$; $\beta \leftarrow \frac{1}{\gamma}$;
6. $[L, D] \leftarrow$ *rankone*$(L, D, 1, \beta\theta_\mathcal{P}^k)$;
7. $(y, \nu, \mathcal{P}, \gamma, \beta) \leftarrow$ *get_feasible*$(\mathcal{P})$; $h \leftarrow h + 1$;
8. $\delta \leftarrow \beta\left(\gamma + (\theta_\mathcal{P}^k)' \begin{bmatrix} y_\mathcal{P} \\ \nu \end{bmatrix}\right)$, $a \leftarrow W_\mathcal{P}' \begin{bmatrix} y_\mathcal{P} \\ \nu \end{bmatrix}$, $w \leftarrow Ma + \beta\delta d^k$;
9. $\rho \leftarrow a'a + \delta^2$;
10. **if** $\rho = 0$ **then** QP problem (1) is infeasible; **go to** Step 18;
11. **if** $w_i \geq -\beta\delta\sigma, \forall i \in \{1, \ldots, m\}$, **or** $\mathcal{P} = \{1, \ldots, m\}$ **then go to** Step 14;
12. $i \leftarrow \arg\min_{i \in \{1,\ldots,m\}\setminus\mathcal{P}} w_i$, $\mathcal{P} \leftarrow \mathcal{P} \cup \{i\}$, $\bar{\beta} \leftarrow \beta$, $\gamma \leftarrow \gamma + |d_i|$; $\beta \leftarrow \frac{1}{\gamma}$;
13. update LDL^T factorization as in (23), then as in (20); **go to** Step 7;
14. $k \leftarrow k + 1$;
15. $\begin{bmatrix} \lambda_k \\ \mu_k \end{bmatrix} \leftarrow \frac{1}{\delta}\begin{bmatrix} y \\ \nu \end{bmatrix}$; $u_k \leftarrow \frac{1}{\delta}a$; $z_k \leftarrow \mathcal{L}^{-1}(u_k - v_k)$;
16. **if** $\|z_k - z_{k-1}\| > \eta$ **then** $[L, D] \leftarrow$ rankone$(L, D, -1, \beta d_\mathcal{P}^k)$; **go to** Step 5;
17. $z^* \leftarrow z_k$, $\lambda^* \leftarrow \lambda_k$, $\mu^* \leftarrow \mu_k$, $\mathcal{P}^* \leftarrow \mathcal{P}$;
18. **end**.

19. **procedure**$(y, \nu, \mathcal{P}, \gamma, \beta) \leftarrow$ *get_feasible*$(\mathcal{P})$
19.1. $\begin{bmatrix} s_\mathcal{P} \\ \nu \end{bmatrix} \leftarrow$ solution of LS problem (15), $s_{\{1,\ldots,m\}\setminus\mathcal{P}} \leftarrow 0$;
19.2. **if** $s_\mathcal{P} \geq 0$ **then** $y \leftarrow s$ and **go to** Step 19.8;
19.3. $j \leftarrow \arg\min_{h \in \mathcal{P}:\ s_h \leq 0} \left\{\frac{y_h}{y_h - s_h}\right\}$;
19.4. $y \leftarrow y + \frac{y_j}{y_j - s_j}(s - y)$;
19.5. $\mathcal{I} \leftarrow \{h \in \mathcal{P} : y_h = 0\}$, $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{I}$; $\bar{\beta} = \beta$, $\gamma \leftarrow \gamma - \|d_\mathcal{I}\|_1$, $\beta = \frac{1}{\gamma}$;
19.6. update LDL^T factorization as in (21), then as in (23);
19.7. **go to** Step 19.1;
19.8. **end procedure**.

Output: Primal solution z^* solving (1), dual solution (λ^*, μ^*), optimal active set $\mathcal{P}^*$, or infeasibility status.*

$0 < \alpha \leq 1$, that keeps the new vector y nonnegative. This is executed in Step 19.4, where $\alpha = \frac{y_j}{y_j - s_j}$, and, since $y_j \geq 0$ and $s_j < 0$, $\alpha \leq 1$.

In many applications, such as in model predictive control (MPC) based on linear time-invariant prediction models, one needs to solve several QPs of the same size in which only c, b, g change value from one problem to another. In such a circumstance, given the LDL^T factorization of $M_\mathcal{P} M_\mathcal{P}' + \beta^2 d_\mathcal{P} d_\mathcal{P}'$ for a given problem and a combination $\mathcal{P}$ of active constraints, the factorization of $M_\mathcal{P} M_\mathcal{P}' + \tilde{\beta}^2 \tilde{d}_\mathcal{P} \tilde{d}_\mathcal{P}'$ for the new problem is obtained by subtracting $\beta^2 d_\mathcal{P} d_\mathcal{P}'$ and adding $\tilde{\beta}^2 \tilde{d}_\mathcal{P} \tilde{d}_\mathcal{P}'$, that is two rank-one updates similar to (24). In addition, in the proximal-point iterations (6) vector z_0 can be set to an arbitrary initial guess, such as the shifted previous optimal sequence of inputs in MPC problems.

* From (12) we have (Q+eps*I)z_k = -c + epsil z_{k-1}+A' lambda_k+ G' mu_k and hence ||Qz_k+c+A'lambda_k+G'mu_k|| = epsil*||z_k-z_{k-1}|| <= epsil*eta when Algorithm 1 stops

IV. QP Solution Algorithm

Algorithm 1 describes the overall approach to solve QP (1) by combining the proximal-point algorithm iterations (6) with the active-set method for strictly convex QPs that derives from Theorem 1. The latter is efficiently implemented by using recursive LDL^T factorizations, as described in Section III-C.

Steps 1–3 initialize all quantities, including the LDL^T factorization of matrix NN' in case equality constraints are present. Such a factorization in Step 3 is equivalent to executing (24a).

Active-set iterations begin at Step 5. The execution of Step 6 corresponds to (24b). Steps 7–8 compute a feasible triplet (y, ν, w) satisfying (25) and the values of γ, β for the given active set $\mathcal{P}$. In particular, the least-squares problem (15), (18) is solved using the available LDL^T factorization (19); in case the solution does not satisfy the nonnegative constraints, the active set $\mathcal{P}$ is possibly reduced inside the get_feasible procedure at Step 24 and, consistently, the LDL^T factorization updated at Step 25.

Step 10 computes the corresponding squared norm ρ of the residual and in case $\rho = 0$, in accordance with Corollary 2, stops the algorithm for infeasibility of QP (1). If at Step 11 vector w is considered feasible (within tolerance), an hence an optimal triple (y, ν, w) has been found, or if all inequality constraints have entered the active set, the inner active-set iterations are stopped and the new solution z_k is computed at Step 14. Otherwise inner active set iterations continue from Step 5. Step 16 continues executing proximal-point iterations if the primal solution is still changing by at least η, otherwise the algorithm computes the optimal solution at Step 17 and stops.

The index k counts the number of proximal-point iterations, the index h the number of active-set iterations. In the practical implementation of Algorithm 1, a maximum number $h_{\max}$ of active-set iterations should be imposed, so that the algorithm is possibly stopped after Step 7 if $h > h_{\max}$. Note that if $h_{\max}$ is reached when $k \geq 1$, the current iterate z_k is feasible, although it may not be optimal.

V. Numerical Results

A. Random QPs

We compare the performance of the proposed QP solver (labeled as QPNNLS PROX), with $\epsilon = 0.1$ and $\sigma = \eta = 3.45 \cdot 10^{-5}$ (that is the square root of the machine precision used for the tests in single-precision arithmetic), against other solution methods: the QP solver based on NNLS proposed in [11] using recursive LDL^T updates (QPNNLS); the accelerated gradient projection method of [7] (GPAD) applied to the dual of QP (1) with diagonal preconditioning [20, Sec. 2.3.1]; the alternative direction method of multipliers [9] (ADMM), run with $\rho = 0.2$ for a fixed number of iterations (1000 or 3000) in all instances, with feasibility/optimality criteria conditions disabled and no preconditioning. The numbers of ADMM iterations are chosen to achieve good speed of execution without compromising the quality of the solution. All algorithms are implemented in Embedded MATLAB code, using single-precision arithmetic to focus our study on numerical robustness. The numerical experiments were obtained on a Macbook Pro 3GHz Intel Core i7 with 16GB RAM running MATLAB R2016b.

Figs. 1 and 2 show the results obtained on random feasible QP problems with $n = 30$ variables, $m = 100$ inequality constraints, $p = 0$ equality constraints. The condition number κ of the primal Hessian Q is varied between 10 and 10^8. For each κ, we report the CPU time spent in the worst case over 100 QP instances, as this is the key figure for real-time applications. In evaluating the CPU time, we also consider the time spent on preprocessing the matrices of the problem (factorizations, preconditioning, etc.). Such extra computations could be moved off-line when the QP solver is used for real-time implementation of an MPC controller based on a linear time-invariant prediction model, as described in Section V-B. The distance of the solution from optimality is computed by solving each QP also in double precision using MATLAB's QUADPROG solver.

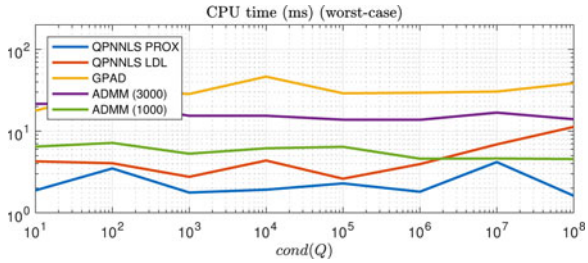

Fig. 1. Worst-case CPU time spent on random QP problems.

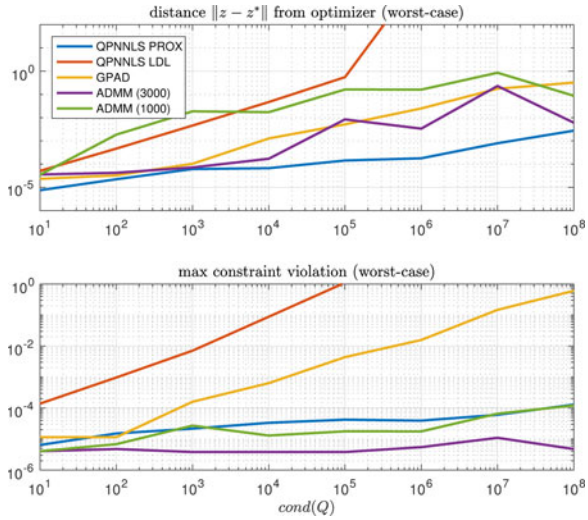

Fig. 2. Distance from optimizer and maximum violation of the inequality constraints on random QP problems.

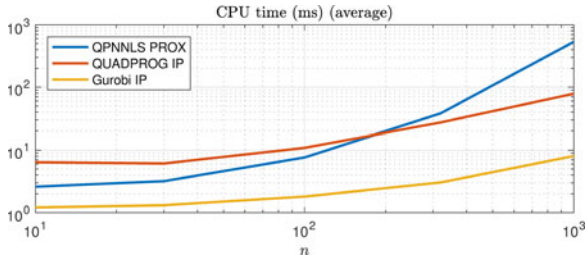

Fig. 3. Random sparse QP problems with n variables and $3n$ inequality constraints. Algorithm 1 is implemented in interpreted MATLAB code using sparse linear algebra.

In order to test our QP solver when dealing with sparse problems, we run it on problems with n variables and $m = 3n$ inequality constraints, matrix Q with $n/2$ nonzero entries and $\text{cond}(Q) = 10^4$, matrix A with $m/2$ nonzero entries, $\epsilon = 10^{-6}$, $\eta = 10^{-4}$, in interpreted MATLAB code using sparse linear algebra in double precision arithmetic. A saturation on γ at 10^4 is enforced due to the large number of constraints. MATLAB's backslash function is used to solve the least-squares problem at Step 20. Fig. 3 shows the performance obtained by averaging the results of 100 random QP instances for each n and compared to the interior-point methods of QUADPROG and GUROBI. As expected, being Algorithm 1 an active-set method, its performance gets worse with

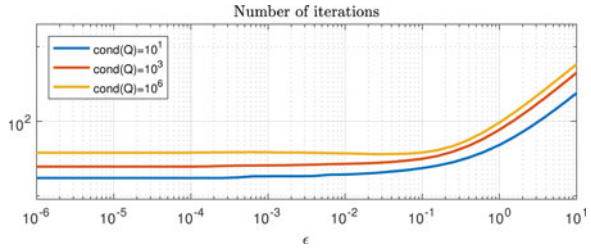

Fig. 4. Average number of iterations as a function of ϵ [500 random QP problems with $n = 30$ variables and $m = 100$ inequality constraints solved for each cond(Q)].

respect to interior point methods, as the number of active-set iterations increases with m. The maximum constraint violation is about 10^{-10}.

Finally, we test the sensitivity of the algorithm with respect to the regularization term ϵ. The results, obtained in double precision by averaging the number of iterations on 500 random QP problems with $n = 30$ variables, $m = 100$ inequality constraints, $\sigma = 10^{-5}$, $\eta = 10^{-6}$, for cond$(Q) = 10, 10^3, 10^6$, are plotted in Fig. 4. It is apparent that Algorithm 1 is not very sensitive w.r.t. ϵ up to about 0.1, for which the proximal iterations start being the dominant factor. Note that the choice of ϵ affects the practical convergence rate of the algorithm, from the one of an active-set method (ϵ small) to the one of a proximal-point method (ϵ large). We expect that a thorough theoretical analysis of the asymptotic convergence of Algorithm 1 would lead to very conservative results, due to the worst-case exponential complexity of active-set iterations.

B. Model Predictive Control

We consider the AFTI-F16 aircraft control example of [21] under the settings of the demo `aft16.m` in the Hybrid Toolbox for MATLAB [22], with a prediction and control horizon of N steps and hard input and soft constraints enforced over the prediction horizon as described in [11, Sec. IV-B]. This MPC formulation leads to a QP problem (1) with $n = 2N + 1$ optimization variables (the extra variable is needed to soften output constraints, that is weighted with a penalty of 10^4) and $q = 4N + 2(N - 1)$ constraints. The reference trajectory is 0 for the constrained output, and switches between $\pm 10^\circ$ for the second output, over a 12 s simulation interval.

We compare Algorithm 1 with $\epsilon = 10^{-6}$, $\sigma = \eta = 1.5 \cdot 10^{-8}$ (that is the square root of the machine precision used in double precision arithmetic) against: Dantzig's active set algorithm [1, Sec. 24-4] applied to solve the dual problem of (1) (coded in C); ADMM with 3000 iterations and parameter $\rho = 1$ (with a lower number of iterations ADMM provide solutions of excessively low quality); the interior-point method of GUROBI [23]; QPOASES [3] (coded in C). For a homogeneous comparison, all computations are done in double-precision arithmetic. Warm starting from the shifted previous optimal solution is used. In case of Algorithm 1, warm starting happens in two ways: the shifted previous optimal sequence of moves is used as the initial guess z_0 in (6), and the previous optimal active set $\mathcal{P}^*$ is used as initial guess for the new active set, by computing the initial LDL^T factorization via two rank-one updates as described in Section III-D.

The worst-case CPU time encountered during the simulation for prediction horizons between 2 and 30 is reported in Fig. 5. Such a CPU time excludes the time spent at the initial step $t = 0$ of the simulation, that does not benefit from warm starting. It is apparent that Algorithm 1 performs similarly to the original algorithm in [11], but is much more robust as reported in Section V-A.

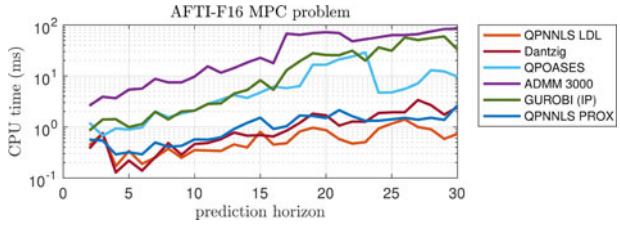

Fig. 5. Worst-case CPU time (ms) for the AFTI-F16 MPC problem as a function of the prediction horizon.

VI. Conclusion

This paper has proposed a novel QP solution method tailored to embedded optimization applications, in which speed of execution and numerical robustness are both main concerns. The solver accepts positive semidefinite Hessian matrices, arbitrary linear inequality and equality constraints, and warm starting. The performance of the solver competes with other state-of-the-art methods. In particular, it inherits the fast speed of execution of the solver in [11], while gaining a lot in numerical robustness, thanks to the proximal-point iterations that allow regularizing the Hessian matrix without altering the optimality of the solution. Future research will explore the use of different NNLS methods (e.g., [13], [24]) in alternative to the basic active set method of [12], especially for dealing with large-scale problems.

Appendix
Proof of Theorem 1

The proof extends the proof reported in [11] to the case of equality constraints. First, by defining the change of coordinates

$$u \triangleq \mathcal{L}z + \mathcal{L}^{-T}(c - \epsilon z_k) \tag{26}$$

we complete the squares in (1) by substituting $z = \mathcal{L}^{-1}u - (Q + \epsilon I)^{-1}(c - \epsilon z_k)$ and recast (1) into the equivalent constrained LDP

$$\min_u \quad \frac{1}{2}\|u\|^2 \tag{27a}$$
$$\text{s.t.} \quad Mu \le d^k \tag{27b}$$
$$Nu = f^k. \tag{27c}$$

1) Assume the optimal residual $r^* = 0$ in (11b). Let $\nu_+^* \triangleq \max\{\nu^*, 0\}$, $\nu_-^* \triangleq \max\{-\nu^*, 0\}$ be the positive and negative parts of ν^*, $\nu^* = \nu_+^* - \nu_-^*$, $\nu_+^*, \nu_-^* \ge 0$. Then, by (11b) we get

$$\begin{aligned} M'y^* + N'\nu_+^* - N'\nu_-^* &= 0 \\ (d^k)'y^* + (f^k)'\nu_+^* - (f^k)'\nu_-^* &= -\gamma \\ y^*, \nu_+^*, \nu_-^* &\ge 0. \end{aligned} \tag{28}$$

By Farkas's Lemma[25, p. 201], for any $\gamma > 0$ (28) is equivalent to infeasibility of

$$Mu \le d^k,\ Nu \le f^k,\ -Nu \le -f^k$$

which is obviously equivalent to (27b) and (27c). Therefore, the LDP problem (27) does not admit a solution, and consequently (1).

2) Consider the Karush-Kuhn-Tucker (KKT) conditions for problem (10)

$$-\begin{bmatrix} M & \beta d^k \\ N & \beta f^k \end{bmatrix} \begin{bmatrix} -M'y^* - N'\nu^* \\ -\beta((d^k)'y^* + (f^k)'\nu^* + \gamma) \end{bmatrix} - \begin{bmatrix} I \\ 0 \end{bmatrix} w^* = 0 \tag{29a}$$

$$(y^*)'w^* = 0 \tag{29b}$$

$$\nu^* \text{ free},\ w^* \ge 0, y^* \ge 0 \tag{29c}$$

where w^* is the optimal dual vector for problem (10). From (29a) we get

$$-\left[\begin{array}{cc} M & \beta d^k \end{array}\right] r^* - w^* = 0 \tag{30a}$$

$$-\left[\begin{array}{cc} N & \beta f^k \end{array}\right] r^* = 0 \tag{30b}$$

and, hence, the condition $r^* \neq 0$, (29a), (29b), and (30) imply that $0 < (r^*)'r^* = (r^*)'\left[\begin{smallmatrix} -M' \\ -\beta(d^k)' \end{smallmatrix}\right] y^* + (r^*)'\left[\begin{smallmatrix} N' \\ \beta(f^k)' \end{smallmatrix}\right]\nu^* - \gamma\beta r^*_{n+1} = (w^*)'y^* - \gamma\beta r^*_{n+1} = -\gamma\beta r^*_{n+1}$, i.e., $r^*_{n+1} = -\beta((d^k)'y^* + (f^k)'\nu^* + \gamma) < 0$. By letting

$$u^* \triangleq -\frac{1}{\beta r^*_{n+1}} r^*_{\{1,\dots,n\}} = -\frac{M'y^* + N'\nu^*}{\beta^2(\gamma + (d^k)'y^* + (f^k)'\nu^*)} \tag{31}$$

from (29c) and (30a) we obtain

$$0 \leq w^* = -\left[\begin{array}{cc} M & \beta d^k \end{array}\right] r^* = -r^*_{n+1}\left[\begin{array}{cc} \beta M & \beta d^k \end{array}\right]\left[\begin{array}{c} \frac{r^*_{\{1,\dots,n\}}}{\beta r^*_{n+1}} \\ 1 \end{array}\right]$$

and, hence, $-Mu^* + d^k \geq 0$, or equivalently u^* satisfies (27b). Moreover, $Nu^* - f^k = -N\frac{M'y^* + N'\nu^*}{\beta^2(\gamma + (d^k)'y^* + (f^k)'\nu^*)} - f^k = 0$ iff $0 = NM'y^* + NN'\nu^* + \beta^2(\gamma f^k + f^k(d^k)'y^* + f^k(f^k)'\nu^*) = (NN' + \beta^2 f^k(f^k)')\nu^* + (NM' + \beta^2 f^k(d^k)')y^* + \beta^2\gamma f^k$, or equivalently iff

$$\nu^* = -\left[\begin{array}{c} N' \\ \beta(f^k)' \end{array}\right]^{\#}\left(\left[\begin{array}{c} M' \\ \beta(d^k)' \end{array}\right] y^* + \left[\begin{array}{c} 0 \\ \beta\gamma \end{array}\right]\right). \tag{32}$$

Since by [26, Lemma 1] condition (32) is always satisfied at optimality of (10), we have proved that u^* also satisfies (27c) and therefore is a feasible candidate to solve (27). It remains to prove that u^* is also optimal for (27). To this end, consider the remaining KKT conditions of optimality for problem (27)

$$u^* + M'\lambda^* + N'\mu^* = 0 \tag{33a}$$

$$(\lambda^*)'(Mu^* - d^k) = 0 \tag{33b}$$

$$\lambda^* \geq 0,\ \nu^* \text{ free.} \tag{33c}$$

Let

$$\lambda^* \triangleq -\frac{1}{\beta r^*_{n+1}} y^*,\ \mu^* \triangleq -\frac{1}{\beta r^*_{n+1}}\nu^*.$$

By negativity of r^*_{n+1} and nonnegativity of y^* we get $\lambda^* \geq 0$. Moreover, by recalling (31), we get

$$u^* = \frac{1}{\beta r^*_{n+1}}(M'y^* + N'\nu^*) = -M'\lambda^* - N'\nu^*$$

so that also (33a) is satisfied. To prove (33b), we observe that $(\lambda^*)'(Mu^* - d^k) = -\frac{1}{\beta r^*_{n+1}}(\lambda^*)'(Mr^*_{\{1,\dots,n\}} + \beta d^k r^*_{n+1}) = \frac{1}{\beta^2(r^*_{n+1})^2}(y^*)'\left[\begin{array}{cc} M & \beta d^k \end{array}\right] r^* = -\frac{1}{\beta^2(r^*_{n+1})^2}(y^*)'w^* = 0$ because of (30a) and (29b). In conclusion, u^* is the optimal solution of problem (27), and hence vector z^* defined in (12) solves (6). As problems (27) and (6) only differ for the transformation (26) of primal variables, by (33) (λ^*, μ^*) is also the optimal dual pair for (6). This can be also directly inspected by substituting $u^* = \mathcal{L}z^* + \mathcal{L}^{-T}(c - \epsilon z_k)$ in (33a) and left-multiplying by $\mathcal{L}'$, and in (33b). ■

ACKNOWLEDGMENT

The author would like thank Prof. S. Boyd for a very inspiring discussion on the use of iterative refinement in solving linear systems and its relation with proximal methods.

REFERENCES

[1] G. Dantzig, *Linear Programming and Extensions*. Princeton, NJ, USA: Princeton Univ. press, 1998.
[2] N. Ricker, "Use of quadratic programming for constrained internal model control," *Ind. Eng. Chem. Process Des. Dev.*, vol. 24, no. 4, pp. 925–936, 1985.
[3] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.
[4] C. Schmid and L. Biegler, "Quadratic programming methods for reduced Hessian SQP," *Comput. Chem. Eng.*, vol. 18, no. 9, pp. 817–832, 1994.
[5] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optim. Eng.*, vol. 13, pp. 1–27, 2010.
[6] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.
[7] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Trans. Autom. Control*, vol. 59, no. 1, pp. 18–33, Jan. 2014.
[8] S. Richter, C. Jones, and M. Morari, "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method," *IEEE Trans. Autom. Control*, vol. 57, no. 6, pp. 1391–1403, Jun. 2012.
[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[10] G. Banjac, B. Stellato, N. Moehle, P. Goulart, A. Bemporad, and S. Boyd, "Embedded code generation using the OSQP solver," in *Proc. 56th IEEE Conf. Decis. Control*, Melbourne, Australia, 2017. [Online]. Available: https://github.com/oxfordcontrol/osqp
[11] A. Bemporad, "A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control," *IEEE Trans. Autom. Control*, vol. 61, no. 4, pp. 1111–1116, Apr. 2016.
[12] C. Lawson and R. Hanson, *Solving Least Squares Problems*, vol. 161. Philadelphia, PA, USA: SIAM, 1974.
[13] L. Portugal, J. Júdice, and L. Vicente, "A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables," *Math. Comput.*, vol. 63, no. 208, pp. 625–643, 1994.
[14] S. Bellavia, M. Macconi, and B. Morini, "An interior point newton-like method for non-negative least-squares problems with degenerate solution," *Numer. Linear Algebra Appl.*, vol. 13, no. 10, pp. 825–846, 2006.
[15] A. Bemporad, "Solving mixed-integer quadratic programs via nonnegative least squares," in *Proc. 5th IFAC Conf. Nonlinear Model Predictive Control*, Sevilla, Spain, 2015, pp. 73–79.
[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, 2004. [Online]. Available: http://www.stanford.edu/ boyd/cvxbook.html
[17] H. Bauschke and P. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York, NY, USA: Springer Science & Business Media, 2011.
[18] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014.
[19] P. Gill, G. G. H. Golub, W. Murray, and M. Saunders, "Methods for modifying matrix factorizations," *Math. Comput.*, vol. 28, no. 126, pp. 505–535, Apr. 1974.
[20] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA, USA: Athena Scientific, 1999.
[21] A. Bemporad, A. Casavola, and E. Mosca, "Nonlinear control of constrained linear systems via predictive reference management," *IEEE Trans. Autom. Control*, vol. AC-42, no. 3, pp. 340–349, Mar. 1997.
[22] A. Bemporad, *Hybrid Toolbox—User's Guide*, Dec. 2003. [Online]. Available: http://cse.lab.imtlucca.it/ bemporad/hybrid/toolbox
[23] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*, 2016. [Online]. Available: http://www.gurobi.com
[24] J. Kim and H. Park, "Fast nonnegative matrix factorization: An active-set-like method and comparisons," *SIAM J. Sci. Comput.*, vol. 33, no. 6, pp. 3261–3281, 2011.
[25] R. Rockafellar, *Convex Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 1970.
[26] A. Bemporad, "A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares," *IEEE Trans. Autom. Control*, vol. 60, no. 11, pp. 2892–2903, Nov. 2015.