Embedded ADMM-based QP Solver for MPC with polytopic constraints

Thuy V. Dang, K.V. Ling and J.M. Maciejowski

Abstract— **We propose an algorithm for solving quadratic programming (QP) problems with inequality and equality constraints arising from linear MPC. The proposed algorithm is based on the 'alternating direction method of multipliers' (ADMM), with the introduction of slack variables. In comparison with algorithms available in the literature, our proposed algorithm can handle the so-called sparse MPC formulation with general inequality constraints. Moreover, our proposed algorithm is suitable for implementation on embedded platforms where computational resources are limited. In some cases, our algorithm is division-free when certain fixed matrices are computed offline. This enables our algorithm to be implemented in fixed-point arithmetic on a FPGA. In this paper, we also propose heuristic rules to select the step size of ADMM for a good convergence rate.**

I. Introduction

Model Predictive Control (MPC) is an optimization-based control strategy which has been applied widely in industry since its appearance in the 1980s. Usually a QP problem is solved at each sampling time instant to determine the control action. In some cases, this QP problem can be solved offline by multi-parametric programming [1]. In other cases, where online solution of this QP is needed, second order methods such as Interior Point Method (IPM) and Active Set Method (ASM) are two commonly employed approaches [2], [3]. The main computational load of IPM and ASM is the solution of a set of linear equations at every iteration, and this can be the bottleneck for embedded systems with limited resources. Recently, first order QP solvers, such as gradient-based method or the 'alternating direction method of multipliers' (ADMM), have received significant interest [4]–[6] because of their simpler computational structure. Interest in ADMM for quadratic linear MPC can be found in [7]–[9].

In MPC, if the system model is linear and the cost function is quadratic, one can formulate the MPC optimisation problem as a sparse QP problem, keeping both the states and controls as decision variables. In contrast, MPC can also be formulated as a dense QP problem keeping only the controls as decision variables [10]. The sparse QP will have both equality and inequality constraints while the dense QP will have only inequality constraints. One main advantage of formulating a sparse QP is that the Hessian matrix has a banded structure and this can be exploited for computational advantage [11] whereas the Hessian in dense QP formulations does not have this property. In [12] and [13], an ADMM-based QP solver was proposed for the dense QP, whereas in [9] [14] ADMM-based QP solver relying on splitting techniques of [7] has been proposed for sparse QP problem.

The limitation of the method proposed in [9] [14] for sparse QP is that a projection onto the polytopic set that represents the inequality constraints is needed. If this set is not simple, the projection is computationally very demanding. As a consequence, the algorithm in these works are not suitable for embedded implementation for general inequality constraints. A common work-around is to restrict the inequalities to be simple box-type constraints so that this projection can be computed easily at each iteration.

In this paper, we consider solving the sparse QP problem arising from MPC. We propose an ADMM-based algorithm to solve this class of QP using the slack variables approach. ADMM with slack variables to solve dense QP has been discussed in [12]. A key motivation for introducing slack variables is to convert the polytopic set arising from general inequalities into a simpler positive orthant constraint set so that projection onto this postive orthant set can be handled even on an embedded platform where computational resources are limited. In some cases, our algorithm is division-free when some fixed matrices are computed offline. This enables our algorithm to be implemented in fixed-point arithmetic on a FPGA.

In ADMM-based algorithms, the step-size affects the convergence rate of the algorithm. A good choice of this parameter will significantly improve the convergence rate. In [12] the suggested choice of this parameter was derived for dense QP problems. For sparse QP, in [9] step-size selection is not addressed.

By writing our algorithm in a matrix recurrence form, we analyse the convergence rate and propose methods of choosing the step size parameter of our ADMM-based algorithm. Although the recurrence form of our algorithm is similar to that of [12], we cannot directly apply the step-size selection method proposed there, because the Hessian matrix arising from sparse MPC formulations is usually not positive-definite. In this paper, we derive heuristic step-size selection methods for our algorithm.

The rest of the paper is organized as follows. In Section II we summarise the method of using ADMM to solve sparse QP. Step size selection is investigated in Section III. Section IV discusses the implementation aspects with fixed-point arithmetic for embedded platform. In Section V, an example is used to illustrate our algorithm. Conclusions are given in Section VI.

Thuy V. Dang is with the Interdisciplinary Graduate School, K.V. Ling is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798. J.M. Maciejowski is with the University of Cambridge, UK, and with the Energy Research Institute at NTU (ERI@N). (e-mail:DANG0028@e.ntu.edu.sg; EKVLING@ntu.edu.sg; jmm@eng.cam.ac.uk). The authors acknowledge the support by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

Notation

$\lambda(M)$ denotes the set of eigenvalues of M; $\lambda_i(M)$ is the i-th smallest in modulus eigenvalue and $\lambda_{max}(M)$ is the largest in modulus eigenvalue of M; $||.||$ is the norm of a vector or matrix, $|.|$ is elementwise absolute values of a vector; $I_{\mathbb{K}}(.)$ is the indicator function of $\mathbb{K}$: $I_{\mathbb{K}}(x) = 0$ if $x \in \mathbb{K}$ and $I_{\mathbb{K}}(x) = +\infty$ otherwise.

II. ADMM Algorithm for Sparse QP

In this section, we propose our algorithm for solving QP of the following form

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}x^T Q x + q^T x \tag{1}$$
$$\text{s.t.} \quad Gx \le g \tag{2}$$
$$Fx = f \tag{3}$$

where $Q \in \mathbb{R}^{n\times n}$ positive semi-definite. $F \in \mathbb{R}^{p\times n},\ G \in \mathbb{R}^{q\times n}$ and $x \in \mathbb{R}^n,\ q \in \mathbb{R}^n,\ f \in \mathbb{R}^p,\ g \in \mathbb{R}^q$ are vectors.

A. Slack variable approach

We now formulate an ADMM-based QP solver for sparse QP problems arising from MPC using the slack variable approach. The slack variable approach was described in [12] for ADMM-based QP solver for dense QP problems arising from MPC.

By introducing the slack variable z to the inequality constraint (2), we obtain the following problem:

$$\underset{x,z}{\text{minimize}} \quad \frac{1}{2}x^T Q x + q^T x + I_{\mathbb{Z}}(z) \tag{4}$$
$$s.t \quad Ax + Bz = c \tag{5}$$
$$\mathbb{Z} = \{z : z \ge 0\}) \tag{6}$$

where

$$A = \begin{bmatrix} G \\ F \end{bmatrix},\ B = \begin{bmatrix} I \\ 0 \end{bmatrix},\ c = \begin{bmatrix} g \\ f \end{bmatrix}, \tag{7}$$
$$I_{\mathbb{Z}}(z) : \text{Indicator function of } Z \tag{8}$$

The augmented Lagrangian for the ADMM iteration is defined as

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax+Bz-c) + \frac{\rho}{2}\|Ax+Bz-c\|_2^2$$

where $f(x) = \frac{1}{2}x^T Q x + q^T x,\ g(z) = I_{\mathbb{Z}}(z)$.
The selection of step-size $\rho > 0$ will be discussed later.
Define $\tau = \frac{1}{\rho}y$, the scaled dual variable. The ADMM iterations for problem (4) are

$$x_{k+1} = \arg\ \min_x\ L_\rho(x, z_k, \tau_k) \tag{9}$$
$$z_{k+1} = \arg\ \min_z\ L_\rho(x_{k+1}, z, \tau_k) \tag{10}$$
$$\tau_{k+1} = \tau_k + (Ax_{k+1} + Bz_{k+1} - c) \tag{11}$$

The sub-problem for the x-update in (9) is an unconstrained QP and has the unique solution

$$x_{k+1} = -(Q + \rho A^T A)^{-1}[q + \rho A^T(Bz_k + \tau_k - c)]$$

Here, we have assumed that
Assumption 1: *A is invertible or full column-rank*[1].
The solution of sub-problem (10) is

$$z_{k+1} = \pi_{\mathbb{Z}}(-Gx_{k+1} - \tau_k^g + g) \tag{12}$$

where $\tau_k = \begin{bmatrix} \tau_k^g \\ \tau_k^f \end{bmatrix}$, ($\tau_k^g = \tau_k(1 : q)$ (q first elements), $\tau_k^f = \tau_k(q+1 : m+p)$) (next p elements) and we have used the following notation for projection:

$$\pi_{\mathbb{K}}(z_k) = \arg\ \min_{z\in\mathbb{K}}\ \|z - z_k\|^2 \tag{13}$$

For a projection on the box type set, it reduces to a clipping operation, and (12) is simply:

$$\pi_{\mathbb{Z}}(-Gx_{k+1} - \tau_k^g + g) = max\{0, -Gx_{k+1} - \tau_k^g + g\}$$

Then, our algorithm for solving (1) is as follows:

Algorithm 1 (This paper)
```
Input: $Q, q, A, c, g$.
$x_0, z_0, \tau_0$ fixed (Cold start)
Do
    $x_{k+1} = -(Q + \rho A^T A)^{-1}[q + \rho A^T(Bz_k + \tau_k - c)]$    (14)
    $z_{k+1} = \pi_{\mathbb{Z}}(-Gx_{k+1} - \tau_k^g + g)$
    $= max\{0, -Gx_{k+1} - \tau_k^g + g\}$    (15)
    $\tau_{k+1} = \tau_k + Ax_{k+1} + Bz_{k+1} - c$    (16)
While $||r_k|| \ge \epsilon_{primal}$ or $||s_k|| \ge \epsilon_{dual}$
```

where $r_k = Ax_k + Bz_k - c$, and $s_k = \rho A^T B(z_{k+1} - z_k)$ are the primal and the dual residual, respectively.

B. Comparison with algorithm of [9]

In [9], the authors maintain z as a copy of x and set up the problem as follow:

$$\min \quad \frac{1}{2}x^T Q x + q^T x + I_{\mathbb{A}}(x) + I_{\mathbb{K}}(z) + \frac{1}{2}\rho||z - x||_2^2$$
$$s.t \quad x = z$$

where the constraints, including inequality and equality constraints are split into two sets: $\mathbb{A} : \{x|Fx = f\}$ and $\mathbb{K} : \{z|Gz \le g\}$.
This results in an ADMM-based algorithm as Algorithm 2 in [9]. For comparison:

- In term of flops count (floating point operations), our algorithm requires about 3 times more.
- The advantage of our algorithm is in the second step of the ADMM iterations. It is the sub-problem for the z-update (15) of our algorithm, compared with the second step of Algorithm 2 in [9], which in general requires a projection onto a polytopic set. For general constraints, this projection is very complex and not

[1]This ensures that $A^T A$ is positive definite, and hence $Q + \rho A^T A$ is positive definite.

suitable for embedded implementation. Hence most embedded implementations assume box-type constraints. In contrast, our algorithm requires only a projection onto the positive orthant.

III. Step-size Selection

The convergence rate of ADMM depends on the step size. To devise a step size selection method, [12] used the sign matrix of the variable technique of [15], to write the ADMM iterations in a matrix recurrence form. In this section, we also present our algorithm in a matrix recurrence form and propose a method to select the step-size for our algorithm.

A. Matrix Recurrence form

Recall that $\tau_k = \begin{bmatrix} \tau_k^g \\ \tau_k^f \end{bmatrix}$.The ADMM iterations (14)-(16) can be rewritten as follows:

$$x_{k+1} = -(Q+\rho A^T A)^{-1} \left[q + \rho A^T \left(\begin{bmatrix} z_k \\ 0 \end{bmatrix} + \begin{bmatrix} \tau_k^g \\ \tau_k^f \end{bmatrix} - c \right) \right] \quad (17)$$

$$z_{k+1} = max\{0, -Gx_{k+1} - \tau_k^g + g\} \quad (18)$$

$$\begin{bmatrix} \tau_{k+1}^g \\ \tau_{k+1}^f \end{bmatrix} = \begin{bmatrix} \tau_k^g \\ \tau_k^f \end{bmatrix} + \begin{bmatrix} G \\ F \end{bmatrix} x_{k+1} + \begin{bmatrix} z_{k+1} \\ 0 \end{bmatrix} - \begin{bmatrix} g \\ f \end{bmatrix} \quad (19)$$

Introduce $v_k = z_k + \tau_k^g$.
From (18) and (19), it can be seen that $v_{k+1} = |Gx_{k+1} + \tau_k^g - g| = S_{k+1}(Gx_{k+1} + \tau_k^g - g)$, and $\tau_k^g = v_k - z_k = B_k v_k$ where

$$S_{k+1} = \text{diag}(\text{sign}(Gx_{k+1} + \tau_k^g - g)), \; B_k = \frac{1}{2}(I + S_k) \quad (20)$$

Then, (17)-(19) become

$$x_{k+1} = -(Q+\rho A^T A)^{-1} \left[q + \rho A^T \left(\begin{bmatrix} v_k \\ \tau_k^f \end{bmatrix} - c \right) \right]$$
$$v_{k+1} = S_{k+1}(Gx_{k+1} + B_k v_k - g)$$

After some calculations (which we have omitted because of space constraints), we obtain the following compact iterative formulation:

$$\begin{bmatrix} S_{k+1} v_{k+1} \\ \tau_{k+1}^f \end{bmatrix} = \frac{1}{2} \begin{bmatrix} S_k v_k \\ \tau_k^f \end{bmatrix} + \left(\frac{1}{2} I - \rho A(Q+\rho A^T A)^{-1} A^T \right) \begin{bmatrix} v_k \\ \tau_k^f \end{bmatrix} + \begin{bmatrix} \bar{g} \\ \bar{f} \end{bmatrix} \quad (21)$$

where

$$\begin{aligned} \bar{g} &= -G(Q+\rho A^T A)^{-1} q + \rho G(Q+\rho A^T A)^{-1} A^T c - g \\ \bar{f} &= -F(Q+\rho A^T A)^{-1} q + \rho F(Q+\rho A^T A)^{-1} A^T c - f \end{aligned}$$

By defining the matrices as

$$\bar{S}_k = \begin{bmatrix} S_k & 0 \\ 0 & I_p \end{bmatrix}, \quad t_k = \begin{bmatrix} v_k \\ \tau_k^f \end{bmatrix}, \quad \gamma = \begin{bmatrix} \bar{g} \\ \bar{f} \end{bmatrix} \quad (22)$$

(21) is written as

$$\bar{S}_{k+1} t_{k+1} = \frac{1}{2} \bar{S}_k t_k + \left(\frac{1}{2} I - \rho A(Q+\rho A^T A)^{-1} A^T \right) t_k + \gamma \quad (23)$$

Equation (23) is the matrix recurrence form of Algorithm 1. This is the same as equation (22) of [12]. However, the step-selection method proposed in [12] requires the existence of Q^{-1}, but in sparse QP formulation arising from MPC problem Q is usually only semi-definite. Hence we propose heuristic rules to select the step-size for our proposed algorithm based on the following analysis.

B. Step-size selection

From (23) we have

$$\bar{S}_{k+1} t_{k+1} - \bar{S}_k t_k = \frac{1}{2}(\bar{S}_k t_k - \bar{S}_{k-1} t_{k-1}) + \left(\frac{1}{2} I - \rho A(Q+\rho A^T A)^{-1} A^T \right)(t_k - t_{k-1}) \quad (24)$$

By taking norm [2] (which will be chosen later) of both sides of (24), and since $||t_k - t_{k-1}|| \leq ||\bar{S}_k t_k - \bar{S}_{k-1} t_{k-1}||$, we get:

$$\xi_{k+1} \leq \left(\frac{1}{2} + \left\| \frac{1}{2} I - M \right\| \right) \xi_k = \eta_\rho \xi_k \quad (25)$$

where
$\xi_k = ||(\bar{S}_k t_k - \bar{S}_{k-1} t_{k-1}||, \; M = \rho A(Q+\rho A^T A)^{-1} A^T$ and

$$\eta_\rho = \frac{1}{2} + \left\| \frac{1}{2} I - M \right\| = \frac{1}{2} + \left\| \frac{1}{2} I - \rho A(Q+\rho A^T A)^{-1} A^T \right\| \quad (26)$$

Therefore, the optimal ρ can be obtained by minimising η_ρ over ρ. Different norms can be used, and in this paper, we investigate using (a) the 2-norm and (b) the max-norm to select the step-size parameter.

B.1. Step-size selection with respect to 2-norm

M is symmetric, since Q is symmetric. As the 2-norm of a real symmetric matrix is the same as the spectral radius, we will analyse the eigenvalues of the matrices appearing in the expression for η_ρ.
Defining $T = \frac{1}{2}I - M$, we see that T always has maximum eigenvalues at $\frac{1}{2}$ and minimum eigenvalues equal or greater than $-\frac{1}{2}$ because M always has minimum eigenvalue at 0 independent of ρ. If Q is only (positive) semidefinite, M will have eigenvalues at 1, hence, $\eta_\rho = 1$ regardless of the choice of ρ. However, we observed that if we try to shift the other eigenvalues of T as close as possible to 0, or equivalently, move the eigenvalues of M as close as possible to $\frac{1}{2}$, the convergence speed becomes better.
Hence, as a heuristic, we solve the following problem to select ρ:
Problem 1: Find ρ such that the eigenvalues of $M = \rho A(Q+\rho A^T A)^{-1} A^T$ are as close as possible to $\frac{1}{2}$.
In order to solve Problem 1, we can express the eigenvalues of matrix M explicitly as a function of ρ. This can be achieved as follows: Since $Q \succeq 0$, there exists $P \succeq 0$

[2]In order for (25) to hold, we need the chosen norm to have sub-multiplicative property.

such that $Q = P^T P$ ([16] p8.3). Using the matrix inversion lemma [17], M can be written as

$$M = \rho A(Q + \rho A^T A)^{-1} A^T = A(\rho^{-1} P^T P + A^T A)^{-1} A^T$$
$$= A((A^T A)^{-1} - \quad (27)$$
$$(A^T A)^{-1} \rho^{-1} P^T (I + \rho^{-1} P (A^T A)^{-1} P^T)^{-1} P (A^T A)^{-1}) A^T$$

After some algebraic manipulations, M can be expressed as

$$\begin{aligned} M =& U(SS_d(I - \rho^{-1} P_d + \\ & \rho^{-2} P_d (I + \rho^{-1} P_d)^{-1} \ P_d) S_d^T S^T) U^T \\ =& U \left(SS_d \Psi_\rho S_d^T S^T \right) U^T \end{aligned} \quad (28)$$

where $A = USV^T$, the singular value decomposition of A. Define $S_d = (S^T S)^{-\frac{1}{2}}$[3], then

$$P_d = S_d^T V^T P^T P V S_d = S_d^T V^T Q V S_d \quad (29)$$

$$\Psi_\rho = I - \rho^{-1} P_d + \rho^{-2} P_d (I + \rho^{-1} P_d)^{-1} \ P_d \quad (30)$$

Since $U^T = U^{-1}$, and the fact that the eigenvalues of two similar matrices are the same, we obtain $\lambda(M) = \lambda(SS_d \Psi_\rho S_d^T S^T) = \lambda(\bar{\Psi}_\rho)$,where $\bar{\Psi}_\rho = SS_d \Psi_\rho S_d^T S^T$.
As S is diagonal and singular, and S_d is diagonal, it can be shown that $\bar{\Psi}_\rho = \begin{bmatrix} \Psi_\rho & 0 \\ 0 & 0 \end{bmatrix}$. Hence

$$\lambda(M) = \lambda(\bar{\Psi}_\rho) = \lambda(\Psi_\rho) \cup \{0\} \quad (31)$$

Then problem 1 is equivalent to the following problem:
Problem 2: Find ρ such that the elements of $\lambda(\Psi_\rho)$ are as near as possible to 0.5.
One way to solve Problem 2 is by solving:

$$\min_\rho \sum_{\lambda_i(\Psi_\rho) \neq 0} (\lambda_i(\Psi_\rho) - 0.5)^2 \quad (32)$$

Define

$$f(t) \quad = \quad 1 - t + t(1+t)^{-1} t = \frac{1}{1+t}$$

We have, based on [12], the eigenvalues of Ψ_ρ related to the eigenvalues of P_d as:

$$\lambda(\Psi_\rho) = f(\lambda(\rho^{-1} P_d)) = \frac{1}{1 + \rho^{-1} \lambda(P_d)} \quad (33)$$

Hence, (32) is equivalent to:

$$\min_\rho \sum_{\lambda_i(P_d) \neq 0} \left(\frac{1}{1 + \rho^{-1} \lambda_i(P_d)} - 0.5 \right)^2 \quad (34)$$

It is not easy to solve (34). Since $\frac{1}{1+\rho^{-1}\lambda_1(P_d)}$ and $\frac{1}{1+\rho^{-1}\lambda_{max}(P_d)}$ are the maximum and minimum in the sequence $\frac{1}{1+\rho^{-1}\lambda_i(P_d)}$, we solve the following problem instead:

$$\min_\rho \left(\frac{1}{1 + \rho^{-1} \lambda_1(P_d)} - \frac{1}{2} \right)^2 + \left(\frac{1}{1 + \rho^{-1} \lambda_{max}(P_d)} - \frac{1}{2} \right)^2 \quad (35)$$

[3]S_d is diagonal matrix, since $S^T S$ is diagonal and positive definite (Assumption 1)

Solving $\partial(.)/\partial\rho = 0$ leads to the solution of the following polynomial equation:

$$\begin{aligned} &\beta_1 \beta_2 (\beta_1^2 + \beta_2^2) \rho^4 - (\beta_1^3 + \beta_2^3 - 3\beta_1 \beta_2 (\beta_1 + \beta_2)) \rho^3 \\ &- (\beta_1 - \beta_2)^2 \rho^2 - 2(\beta_1 + \beta_2) \rho - 2 = 0 \end{aligned} \quad (36)$$

where $\beta_1 = (\lambda_1(P_d))^{-1}, \ \beta_2 = (\lambda_{max}(P_d))^{-1}$. In principle, one can obtain an analytical expression for the roots ρ of this 4th-degree equation. More practically, we obtain a numerical solution, selecting the positive root. We outline the procedure for determining ρ offline since A and Q (see (1) and (7)) do not change for MPC problems with linear time invariant models and fixed costs:

1) Given A and Q
2) Calculate P_d as in (29), calculate non-zero eigenvalues $\lambda_1(P_d), \lambda_{max}(P_d)$, $\beta_1 = (\lambda_1(P_d))^{-1}$, $\beta_2 = (\lambda_{max}(P_d))^{-1}$
3) Solve (36) for ρ

B.2. Step-size selection with respect to max-norm

Since we can determine the step-size parameter offline, we perform a simple line search over ρ to minimise η_ρ directly, using the max-norm[4]. The advantage of using the max-norm is that it may handle general A and Q in the iterative form (25). We believe that this max-norm approach may be equally applicable to the method proposed in [12]. A more detailed analysis is the topic of current research. In section V, our experiments suggest that the max-norm and the 2-norm criteria are both effective.

IV. Implementation on an FPGA

Field Programmable Gate Arrays (FPGA) have emerged as popular platforms for real-time embedded MPC applications [18]–[20]. The simplicity of ADMM, and the parallel processing capability of FPGA, offer a promising combination for high speed embedded MPC. In this section, we discuss some implementation aspects of our proposed algorithm on an FPGA.

A. Offline and online computations

For MPC based on a linear time invariant model and with fixed costs and constraints, certain matrices can be computed offline, such as $M = -(Q + \rho A^T A)^{-1}$ and $\bar{M} = \rho M A^T$.

Variables which depend on measurements must of course be computed online, but the computations are relatively fast if matrices have been pre-computed. In particular, for the following algorithm $\bar{q} = Mq - \bar{M}c$ must be computed once each sampling time, when a new state measurement becomes available. Note also that $Bz = [z; 0]$.

Algorithm 1.1 Customized for embedded implementation

$$x_{k+1} \quad = \quad \bar{q} + \bar{M}([z_k; 0] + \tau_k) \quad (37)$$
$$z_{k+1} \quad = \quad max\{0, -Gx_{k+1} - \tau_k^g + g\} \quad (38)$$
$$\tau_{k+1} \quad = \quad \tau_k + Ax_{k+1} + [z_{k+1}; 0] - c \quad (39)$$

[4]The max-norm of a matrix is the largest absolute value of any element of the matrix. To be precise we should use $(p+q) *$ max-norm so that the sub-multiplicative property is satisfied. However, since $p+q$ is fixed, for simplicity, we just say max-norm.

Remark 1: Equation (37) is an option for carrying the calculation step (14) which is the most computational step of Algorithm 1. This approach will result a matrix-vector product, hence favourable for efficient parallelism computation on FPGA. However, the sparsity of the matrices will be destroyed. An alternative option can be offline computation of the Cholesky factorisation of the banded matrix $Q+\rho A^T A$ and then (14) can be carried out by back-substitution. This approach will reduce the computational cost and memory cost since the sparsity will be preserved.

B. Fixed-point Arithmetic Implementation

Since Algorithm 1.1 is division-free, it can be implemented in fixed-point arithmetic on a FPGA. Compared with floating point implementation, a fixed point implementation not only runs faster but also requires fewer hardware resources. In order to implement the algorithm using fixed-point arithmetic, we must establish the range of variables and the accuracy requirements, in order to decide on the word length and number of fractional bits.
We establish an analytical bound of the variables as following: From equation (25), we have:

$$\begin{aligned}
&||t_k||_2 = ||\bar{S}_k t_k||_2 = ||\bar{S}_0 t_0 + \sum_{j=1}^{k}(\bar{S}_j t_j - \bar{S}_{j-1} t_{j-1})||_2 \\
&\leq ||\bar{S}_0 t_0||_2 + \sum_{j=1}^{k}||\bar{S}_j t_j - \bar{S}_{j-1} t_{j-1}||_2 = ||\bar{S}_0 t_0||_2 + \sum_{j=1}^{k} \xi_j \\
&\leq ||\bar{S}_0 t_0||_2 + K_{max}\xi_1 = K_{max}||\gamma||_2 \qquad (40)
\end{aligned}$$

where γ is defined as equation (22), and K_{max} is the maximum number of iterations, a design choice.
We can have the upper bound for $||\gamma||_2$ for a particular system. Once we have a bound for $||t_k||_2$, we will have a bound for $||t_k||_\infty$, i.e $||t_k||_\infty \leq \bar{t}$ since $||t_k||_\infty \leq ||t_k||_2$. Based on $\bar{t}$, we can obtain the range of all variables as well as intermediate variables involved when implementing the algorithm. Due to space constraints, we do not give the details in this paper.

V. Case study

A. Illustrative Example

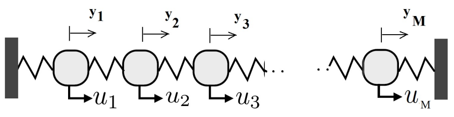

Fig. 1. Spring-mass System

Consider the spring-mass system [21] shown in Fig. 1 with $M = 8$, , spring constant $k = 1\ Nm^{-1}$ and equal masses of $m = 1kg$ each. The states are the position and velocity of each mass, and the inputs are the forces on each mass. Hence the system has 16 states and 8 inputs. The output $y = [y_1; y_2; ...; y_8]$ contains the positions of the 8 masses.

We designed an MPC controller with the following cost function and parameters:

$$min \sum_{k=0}^{N-1} \left(x_k^T Q_x x_k + u_k^T R u_k\right) + x_N^T Q_N x_N \qquad (41)$$

$$\text{s.t. } |y_i| \leq 4, |y_1 + y_2| \leq 6,\ |u_i| \leq 1 \qquad (42)$$

We added the constraint $|y_1 + y_2| \leq 6$ to demonstrate the advantage of our algorithm in handling non-box type constraints. In addition, to ensure Assumption 1 holds, we set a constraint on the velocities to a large value: $|\dot{y_i}| \leq 100$.
$Q_x = C_d^T C_d, R = I$ and terminal matrix Q_N is the solution of the discrete algebraic Riccati equation with parameters A_d, B_d, Q_x, R where A_d, B_d, C_d are the state-space matrices of the discrete-time system obtained by sampling the continuous system using a zero-order hold model with a sampling interval $T_s = 0.5$.
With horizon $N = 5$, the sparse QP formulation has 120 decision variables, 80 equality constraints, and 250 inequality constraints. Algorithm 1 was used to solve the QP problem online.

B. Result and Discussion

Figure 2 shows the output and control on the fifth mass. The solution obtained by Algorithm 1 (red-dash square, implemented in MATLAB m-file), agreed with the solution obtained by calling the *quadprog* function in MATLAB (green line). We also implemented Algorithm 1 on the Zynq ZC702 FPGA ((blue-dot circle) in fixed-point arithmetic (32 bit word, 17 fractional bit type 1, and 25 bit word 17 fractional bit type 2). The result produced by the FPGA implementation is almost the same, with relative errors of about 1.56% compared with the solution obtained on a PC. On average, it took about 80 ADMM iterations[6] (or

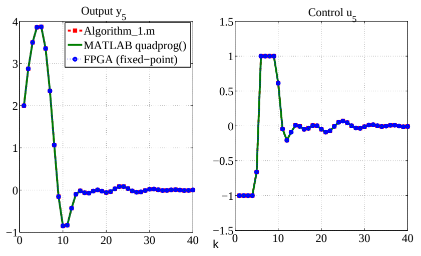

Fig. 2. MPC performance in Spring-Mass system

215 ms) to solve one QP on the FPGA [5], when the step-size $\rho = 2.0815$ was chosen based on the 2-norm method proposed in Section III.

- Figure 3(a) plots, as a function of ρ, the average number of iterations needed to solve QP problems arising from the first 10 sampling instants [6].

[6]Cold start, stopping criteria $\epsilon_{primal} = \epsilon_{dual} = 10^{-4}$ for each sampling instant

[5]Computations are carried out sequentially. There is room for timing improvement if paralleling processing is employed, since Algorithm 1.1 mainly consists of dot-product computations.

- Figure 3(b) shows that the max-norm of matrix T (Section III.B.1) is a good indicator for the number of ADMM iterations.

Although our heuristics did not give the optimal ρ, they nevertheless gave values that are close to the optimal ρ.

C. Comparison between 2-norm and max-norm

In Figure 4 we generated additional QP problems with randomly selected MPC parameters. The results showed that although different values of ρ were selected depending on the different norm used, the resulting number of iterations did not differ much for a wide range of ρ values. It appears that the max-norm could be a better indicator for selecting the step size parameter ρ, although experience needs to be gained with a greater variety of examples.

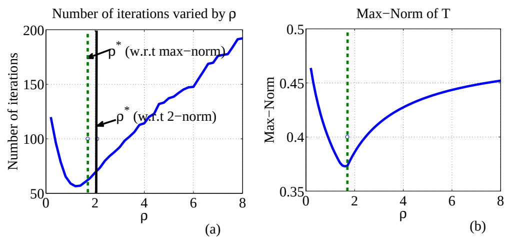

Fig. 3. Selection of ρ based on the 2-norm or max-norm criteria. (MPC parameters: $Q_x = C_d^T C_d$, $R = I$)

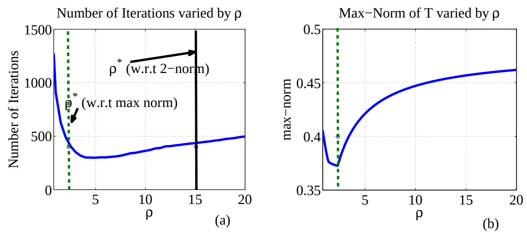

Fig. 4. Selection of ρ based on the 2-norm or max-norm criteria. (Random MPC parameters: $Q_x \succeq 0,\ R \succ 0$

VI. Conclusion

An ADMM-based algorithm for QP with inequality and equality constraints arising from MPC problems was proposed in this paper. By introducing slack variables, the algorithm is greatly simplified because the projection is now onto a positive orthant rather than a general polytopic set. Hence our algorithm can handle general inequality constraints without the complexity of polytope projection. We also proposed heuristic methods based on 2-norm and max-norm criteria to select the step-size of the ADMM iteration for good convergence rates. Simulations confirmed that our heuristics are effective and that the max-norm approach may be more successful. Since the algorithm has simpler computational structure, as well as being division-free if some offline computations are performed, it allows an efficient implementation, with fixed-point arithmetic, on embedded platforms such as an FPGA.

References

[1] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[2] Stephen J. Wright. *Primal-dual interior-point methods*. Siam, 1997.

[3] R Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, 1987.

[4] Stefan Richter, Colin Neil Jones, and Manfred Morari. Real-time input-constrained MPC using fast gradient methods. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 7387–7393. IEEE, 2009.

[5] Panagiotis Patrinos and Alberto Bemporad. An accelerated dual gradient-projection algorithm for linear model predictive control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 662–667. IEEE, 2012.

[6] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[7] Brendan O'Donoghue, Giorgos Stathopoulos, and Stephen P Boyd. A splitting method for optimal control. *IEEE Trans. Contr. Sys. Techn.*, 21(6):2432–2442, 2013.

[8] Markus J Koegel and Rolf Findeisen. Parallel solutions of model predictive control using the alternating direction method of multipliers. In *Nonlinear Model Predictive Control*, volume 4, pages 369–374, 2012.

[9] J.L. Jerez, P.J. Goulart, S. Richter, G.A. Constantinides, E.C. Kerrigan, and M. Morari. Embedded online optimization for model predictive control at megahertz rates. *Automatic Control, IEEE Transactions on*, 59(12):3238–3251, Dec 2014.

[10] J. M. Maciejowski. *Predictive control: with constraints*. Prentice Hall, 2002.

[11] Stephen J Wright. Applying new optimization algorithms to model predictive control. In *AIChE Symposium Series*, volume 93, pages 147–155. Citeseer, 1997.

[12] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): Quadratic problems. *Automatic Control, IEEE Transactions on*, 60(3):644–658, March 2015.

[13] Arvind U Raghunathan and Stefano Di Cairano. Alternating direction method of multipliers for strictly convex quadratic programs: Optimal parameter selection. In *American Control Conference (ACC), 2014*, pages 4324–4329. IEEE, 2014.

[14] Arvind U Raghunathan and Stefano Di Cairano. Optimal step-size selection in alternating direction method of multipliers for convex quadratic programs and model predictive control,. In *Proceedings of Symposium on Mathematical Theory of Networks and Systems*, pages 807–814, 2014.

[15] Daniel Boley. Local linear convergence of the alternating direction method of multipliers on quadratic or linear programs. *SIAM Journal on Optimization*, 23(4):2183–2207, 2013.

[16] Jack J Dongarra, James R Bunch, Cleve B Moler, and Gilbert W Stewart. *LINPACK users' guide*, volume 8. Siam, 1979.

[17] Max A. Woodbury. Inverting modified matrices. *Memorandum Rept. 42, Statistical Research Group, Princeton University, Princeton, NJ*, 1950.

[18] K.V. Ling S.P. Yue and J.M. Maciejowski. A FPGA implementation of model predictive control. In *Proc. American Control Conference (ACC)*, pages 1930–1935. Minneapolis, MN, IEEE, 2006.

[19] E.N. Hartley, J.L. Jerez, A. Suardi, J.M. Maciejowski, E.C. Kerrigan, and G.A. Constantinides. Predictive control using an fpga with application to aircraft control. *Control Systems Technology, IEEE Transactions on*, 22(3):1006–1017, May 2014.

[20] E.N. Hartley and J.M. Maciejowski. Graphical FPGA design for a predictive controller with application to spacecraft rendezvous. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 1971–1976. IEEE, 2013.

[21] Markus J Koegel and Rolf Findeisen. A fast gradient method for embedded linear predictive control. In *IFAC World Congress, Milan, 2011*, pages 1362–1367.