Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

A Parallel Quadratic Programming Algorithm for Model Predictive Control

Matthew Brand, Vijay Shilpiekandula[1], Chen Yao, Scott A. Bortoff

Mitsubishi Electric Research Laboratories (MERL), Cambridge MA 02139 USA.

Takehiro Nishiyama, Shoji Yoshikawa, Takashi Iwasaki

Advanced Technology R&D Center, Mitsubishi Electric Corporation, Japan.

Abstract: In this paper, an iterative multiplicative algorithm is proposed for the fast solution of quadratic programming (QP) problems that arise in the real-time implementation of Model Predictive Control (MPC). The proposed algorithm — Parallel Quadratic Programming (PQP) — is amenable to fine-grained parallelization. Conditions on the convergence of the PQP algorithm are given and proved. Due to its extreme simplicity, even serial implementations offer considerable speed advantages. To demonstrate, PQP is applied to several simulation examples, including a stand-alone QP problem and two MPC examples. When implemented in MATLAB using single-thread computations, numerical simulations of PQP demonstrate a $5-10\times$ speed-up compared to the MATLAB active-set based QP solver `quadprog`. A parallel implementation would offer a further speed-up, linear in the number of parallel processors.

Keywords: Quadratic Programming, Model Predictive Control, Reference Tracking.

1. INTRODUCTION

Model Predictive Control (MPC) is an optimization-based control strategy (Rawlings and Mayne [2009])), which has been successfully applied in a wide range of applications, such as chemical process control (Qin and Badgwell [2003]), servo motion control (Wang [2009]), automotive engine control (Bageshwar et al. [2004], Ferreau [2006]), and multi-agent control in transportation networks (Negenborn et al. [2006]). In MPC, at each sample time, an online optimization problem is solved to minimize user-specified costs over a finite horizon in forward time, and a sequence of controls is obtained (Rossiter [2003]). However, only the first control is applied and the system is driven to the next sample time, when the above procedure is repeated. The cost function at each sample time can have various structures, an example being a quadratic form, which makes the optimization problem a quadratic programming (QP) problem.

One of the main drawbacks of MPC (Camacho and Bordons [2004]) is that it requires long computation times to solve the optimization problem at each sample point, therefore, it is usually restricted to systems with slow dynamics and large sampling intervals, such as chemical processes (Qin and Badgwell [2003]).

Recently, many reports in the literature address applying MPC to control applications with short sampling intervals, by adapting fast optimization algorithms. In particular, fast QP algorithms have been proposed for the MPC problem to enable fast solution of the QP problem posed at each sample time. For instance, Rao et al. [2004], Wang and Boyd [2010] have proposed interior point methods that are specifically tailored to take advantage of the special structure of the QP problem in the MPC setting, thus achieving significant savings in the required computations. Milman and Davison [2008] modified the active set method by giving priorities on the constraints associated with

[1] Corresponding author, email: vijay.shilpiekandula@merl.com

more "recent" future. Richter et al. [2009], Zeilinger et al. [2009] adopted a fast gradient method for the MPC of *LTI* systems with input constraints.

Although significant speed-up has been reported in the above references, many of the algorithms are heuristics without any guarantees on convergence to the global minimizers. In addition, parallel implementation of these algorithms, when possible, will depend on specific problem structures and input data (Gondzio and Grothey [2007]), (Gondzio and Sarkissian [2000]).

In this paper, we develop a fast iterative algorithm to solve QP problems of MPC. The proposed new algorithm is amenable to fine-grain parallelization, hence the name Parallel Quadratic Programming (*PQP*).

The proposed algorithm uses a multiplicative fixpoint that is essentially the KKT first-order optimality conditions expressed as a ratio. Decision variables are rescaled rather than incremented as in gradient-based methods. Similar multiplicative fixpoint algorithms have been used successful in machine learning (e.g., Lee and Seung [1999], Sha et al. [2007]), tomography (e.g., Shepp and Vardi [1982]), image processing (e.g., Bertero et al. [2007]), and estimation (e.g., Eggermont [1990]), however they rely on some combination of strictly nonnegative coefficients, positive definiteness, or favorable initialization for convergence, if convergence is provable at all. *PQP* is provably convergent without such restrictions. *PQP* is also related to matrix splitting algorithms for linear complementarity problems (Luo and Tseng [1992]) and Uzawa methods (Benzi et al. [2005]) for saddle-point problems; unlike these methods, the *PQP* update is given in closed form and can be calculated independently for each variable.

Indeed, the main advantage of the *PQP* algorithm is that it is completely parallelizable for any problem data structure, and can readily exploit the full parallelism of multiprocessor machines, including multi-core, SIMD, and GPU (Kirk and Hwu [2010], Stratton et al. [2008]). Due to its extreme simplicity – two matrix-vector products and a scalar divide – the *PQP* update also offers considerable speed advantages even when implemented on serial computers. For example, it converges in half as many iterations as Sha et al. [2007], and each iteration is faster. Under favorable sparsity conditions, QP solvers based on active set methods Ferreau [2006] and Heath and Wills [2007] and precondition conjugate gradient methods Dostl [2009] can exhibit similar serial-computer time complexity, but these methods are not amenable to fine-grain parallelization.

The rest of this paper is organized as follows. In Section 2, we present the *PQP* algorithm; its convergence is proved in Section 3. In Section 4, we apply *PQP* algorithm to illustrative examples, and use simulations to benchmark the performance of the *PQP* algorithm (against MATLAB active-set solver) in serial computations. Finally, we summarize the contributions of this paper and point out future research directions.

2. *PQP* ALGORITHM

In this section, we first briefly review the conventional MPC scheme (Rawlings and Mayne [2009], Rao et al. [2004]) and then present the *PQP* algorithm to solve QP problems in MPC.

2.1 MPC Scheme

In a conventional MPC scheme, at any sample time point, an optimization problem is solved, to optimize system performances forward in a finite time window. For example, if we assume a linear time invariant model and linear constraints, the following QP problem O(k) must be solved at sample time k:

$$\min_{U_k=[u_k,\ldots,u_{k+N-1}]} \quad \frac{1}{2}U_k^T \mathcal{Q} U_k + \mathcal{H}^T U_k \tag{1}$$

$$s.t. \quad VU_n \leq W \tag{2}$$

where $u_{k+i} \in \mathbb{R}^u$ represents the control input at sample time $k+i$, $\forall i \in [1, N-1]$, N is the window size for MPC scheme, and $U_k \in \mathbb{R}^{uN}$, $\mathcal{Q} \in \mathbb{R}^{uN \times uN}$, $V \in \mathbb{R}^{mN \times uN}$, mN is the number of constraints in (1), which is also the dimension of the dual variable defined later. In addition, V and W can be time-varying, i.e., functions of n, thus leading to time-varying constraints and feasible sets.

We refer to the QP problem presented above as the primal problem. As will be shown for the case of MPC (see Appendix), for the set of "stacked" control inputs $U_k = [u_k, ..., u_{k+N-1}]$ at a sample time k, for the primal problem (i) the cost function of Eq. (1) captures weighted sums of terms involving control inputs *and* those involving states (or outputs), and (ii) the constraint inequalities of Eq. (2) capture control constraints *and* state (or output) constraints.

A receding horizon implementation of MPC is as follows. After an optimal solution $U_n^* = [u_n^*, ..., u_{n+N-1}^*]$ is obtained for the primal QP problem O(k) of Eq. (1) subject to the constraints of Eq. (2), only the first control input u_n^* is applied, which drives the system to the sample time $k+1$, when another QP problem O($k+1$) is solved. This procedure is continued with the window shifted one sample at a time over the receding horizon.

2.2 Assumptions

Before we proceed to solve the QP problem, we make the following assumptions:

Assumption 1: $\mathcal{Q} \succ 0$, i.e. $\mathcal{Q}$ is positive definite in Eq. (1). This assumption holds true for a broad class of MPC problems, including those studied in this paper (see Appendix).

Assumption 2: The primal quadratic programming problem in Eq. (1) is feasible. This means there exists a solution to the QP problem that satisfies the constraints. Note that infeasible QP problems can arise from MPC problems, for example, in cases with time-varying references. Such problems can be made feasible by adding a single slack variable to dilate the feasible region. For simplicity in this paper, we assume that the problem data renders the QP problem feasible at all sample times.

2.3 Dual Problem

To facilitate better handling of the general constraints of the QP problem posed in (1), in what follows, we will propose the PQP algorithm as a multiplicative update law using the Lagrange dual form of the problem in Eq. (1).

For the primal problem of (1), consider the dual form given below:

$$\min_{y} \left\{ F(y) = \quad \frac{1}{2} y^T Q y + y^T h \right\} \tag{3}$$

$$s.t. \quad y \geq 0 \tag{4}$$

where $y \in \mathbb{R}^{N_y}$ is the dual variable, and $Q \succcurlyeq 0$ (i.e. Q is positive semi-definite) and h are obtained under Assumption 1 as

$$Q = V \mathcal{Q}^{-1} V^T \tag{5}$$

$$h = W + V \mathcal{Q}^{-1} \mathcal{H} \tag{6}$$

The optimum U^* of the primal problem in Eq. (1) can be recovered from the optimum y^* of the dual problem in Eq. (6) using the following relation:

$$U^* = -\mathcal{Q}^{-1}(\mathcal{H} + V^T y^*) \tag{7}$$

2.4 PQP Update Law

The PQP algorithm solves the dual QP problem in Eq. (3) by implementing repeated iterations of the following multiplicative update rule:

$$y_i \leftarrow y_i \left[\frac{h_i^- + (Q^- y)_i}{h_i^+ + (Q^+ y)_i} \right], \tag{8}$$

for the i^{th} element y_i of the dual variable y, starting from an initial guess $y^0 > 0$. Here $Q^+ = \max(Q, 0) + \text{diag}(r)$; $Q^- = \max(-Q, 0) + \text{diag}(r)$; $h^+ = \max(h, 0)$; $h^- = \max(-h, 0)$; $\max(a, b)$ is taken elementwise; $\text{diag}(a)$ is a diagonal matrix formed from vector a; and r is a non-negative vector specified later in Section 3.

Note that all terms in the update law of Eq. (8) are non-negative and thus all iterates remain in the non-negative cone. Furthermore, if $\forall_i Q_{ii} + r_i > 0$, where Q_{ii} denotes the i^{th} diagonal element of Q, the bracketed ratio in the update law of Eq. (8) is bounded away from zero and infinity.

Updates of this form were shown by Brand and Chen [2011] to solve very large (10^5 variables) strictly convex QPs in less than 1 second on a GPU [3]; here we develop a variant for semi-definite quadratic programs that arise in MPC duals.

2.5 Algorithm Steps

Given the details of the PQP algorithm presented above, implementing it involves the following steps:

- **Step 1** Formulate a QP problem of the form O(k) shown in Eqs. (1) and (2).
- **Step 2** If the problem variables are constrained to be in the non-negative cone, proceed to **Step 3**. If the problem has general constraints of the form of Eq. (2), formulate the dual QP problem of the form shown in Eqs. (3) and (4).
- **Step 3** With an initial guess $y^0 > 0$, use the update rule of Eq. (8) to solve the dual problem of Eqs. (3) and (4) to a specified tolerance.
- **Step 4** Recover the primal problem solution using Eq. (7).

Step 1 can be implemented in many ways depending on the particular MPC problem. For example, refer to Section 4 and Appendix for a QP formulation for MPC servo problems involving reference tracking. While **Step 2** and **Step 4** involve well-understood matrix and matrix-vector operations, in general, it is not obvious if **Step 3** can guarantee convergence for any chosen update rule for the QP problem. In the following section, we derive a detailed proof specifying conditions that guarantee the convergence of the update rule of Eq. (8) used in **Step 3**.

[3] Current versions of MATLAB `quadprog` cannot handle such large-size QP problems.

3. PROOF OF CONVERGENCE

To prove convergence to optimum y^* of the dual problem 4, we begin with a series expansion of $F(y)$ around y:

$$F(x) = F(y) + (x-y)^T \nabla F(y) + \frac{1}{2}(x-y)^T Q(x-y)$$

For $y > 0$, we modify the last term to define an auxiliary function

$$G(x,y) = F(y) + (x-y)^T \nabla F(y) + \frac{1}{2}(x-y)^T K(y)(x-y)$$

with $K(y)$ a diagonal matrix

$$K(y) = \text{diag}(Q^+ y + h^+)\text{diag}(y)^{-1}$$

The proof argument is:

(1) $G(x,y)$ upper-bounds $F(x)$ with equality at $x = y$.
(2) The multiplicative update yields the solution $y^{k+1} = \arg\min_x G(x, y^k)$
(3) Given a positive initial guess $y^0 > 0$, the sequence of updates $y^1, y^2, y^3, \cdots$ monotonically reduces F.
(4) The sequence asymptotically converges to the minimizer of F in the non-negative cone.

It then follows from Lagrange duality that the solution of the bounded, feasible, and convex primal problem can be recovered from the minimizer of dual problem F.

The remainder of this section fleshes out this argument in lemmas:

Lemma 3.1. For some non-negative vector $r \geq 0$ that depends only on Q, G upper-bounds F, i.e.

$$\exists_{r\geq 0} \forall_{x\geq 0, y>0} F(x) \leq G(x,y).$$

Proof. It will suffice to show that $K(y) - Q \succcurlyeq 0$ (is positive semi-definite) because

$$G(x,y) - F(x) = (x-y)^T (K(y) - Q)(x-y)$$

Clearly the bound is tight at $x = y$. For $x \neq y$ we split $K(y) - Q$ into a sum of a positive semi-definite matrix and a non-negative matrix as follows:

$$\begin{aligned} K(y) - Q &= \text{diag}(Q^+ y + h^+)\text{diag}(y)^{-1} - (Q^+ - Q^-) \\ &= \{\text{diag}(Q^+ y)\text{diag}(y)^{-1} - Q^+\} \\ &\quad + \{\text{diag}(h^+)\text{diag}(y)^{-1} + Q^-\} \\ &= K_{psd} + K_{nn} \end{aligned}$$

with

$$K_{psd} = \{\text{diag}(Q^+ y)\text{diag}(y)^{-1} - Q^+\}$$

and

$$K_{nn} = \text{diag}(r) + \text{diag}(h^+)\text{diag}(y)^{-1} + \max(-Q, 0) \geq 0$$

It is well known from variation methods that $K_{psd} \succcurlyeq 0$. See, for example, the proof in Sha et al. [2007]. We now select $r \geq 0$ to make $K_{nn} \succcurlyeq 0$. For example, selecting any $r \geq \max(-Q, 0)\mathbf{1}$, where $\mathbf{1} = [1,\ 1,\ 1, ...,\ 1]^T$, makes Q^- diagonally dominant and thus positive semi-definite. Then K_{nn} and $K(y) - Q$ are positive semi-definite as well. ■

Remark: It can be shown, at greater length, that convergence can be obtained with smaller values of r, including, in many special but useful cases, $r = 0$.

Lemma 3.2. The multiplicative update rule of Eq. (8) yields the minimum of $G(x,y)$.

Proof. $G(x,y)$ is a nonconcave quadratic in x and thus has a global minimum where

$$\nabla_x G(x,y) = \nabla F(y) + K(y)(x-y) = 0\ . \qquad (9)$$

Solving for x we recover the update:

$$\begin{aligned} x &= y - K(y)^{-1}\nabla F(y) \\ &= y - K(y)^{-1}(Qy + h) \\ &= y - K(y)^{-1}(Q^+ y + h^+ - Q^- y - h^-) \\ &= y - K(y)^{-1}(Q^+ y + h^+) + K(y)^{-1}(Q^- y + h^-) \\ &= y - y + K(y)^{-1}(Q^- y + h^-) \\ &= \text{diag}(y)\text{diag}(Q^+ y + h^+)^{-1}(Q^- y + h^-) \end{aligned}$$

which is the same as the updated value of the dual variable as given in Eq. (8). ■

Lemma 3.3. For any non-optimal y_k, if $\exists i | y_i^k (Q^+ y^k + h^+)_i \neq 0$, then the update reduces the objective:

$$F(y^{k+1}) \leq G(y^{k+1}, y^k) < G(y^k, y^k) = F(y^k)\ .$$

Proof. Lemma 3.1 gives the first inequality; here we prove the second, strict inequality. $G(y^{k+1}, y^k)$ is nonconcave in y^{k+1} by construction, and strictly convex w.r.t. any variable y_i^{k+1} satisfying $y_i^k (Q^+ y^k + h^+)_i \neq 0$, because

$$\partial_{y_i^{k+1} y_i^{k+1}} G(y^{k+1}, y^k) = K(y^k)_{ii} = (Q^+ y^k + h^+)_i / y_i^k > 0\ .$$

Since $K(y^k)_{ii} > 0$ and $y^k \neq y^*$, it follows that $y_i^{k+1} = \{\arg\min_x G(x, y^k)\}_i \neq y_i^k$, because

$$\Delta y_i^k = y_i^{k+1} - y_i^k = -\{K(y^k)^{-1}\nabla F(y^k)\}_i \neq 0\ ; \qquad (10)$$

Together these facts imply that y^k is not a minimizer of $G(x, y^k)$, thus $G(y^{k+1}, y^k) < G(y^k, y^k)$. ■

Theorem 1. Given $\forall_i \quad Q_{ii} + r_i > 0$, $Q \succcurlyeq 0$, and r chosen as per lemma 3.1, the update converges monotonically and asymptotically from any positive $y^0 > 0$ to the minimum of F in the non-negative cone.

Proof. The condition $\forall_i \quad Q_{ii} + r_i > 0$ satisfies the requirement of lemma 3.3 for monotonic descent in F within the positive cone. From Eq. (10) we know that y is a stationary point iff $\forall_i y_i \partial_{y_i} F(y) = 0$. Note that this is the

also the KKT condition at a QP solution y^*, so any y^* is a stationary point of the update. It remains to show that the update has no other fixpoints in the positive cone. Since $\forall_i y_i > 0$ (albeit possibly infinitesimally so), such fixpoints would require $\nabla F = 0$; due to convexity of F, this can only occur at y^* and only if $\forall_i y_i^* > 0$. Consequently any fixpoint of the update is a solution of the QP. ■

Remark: Although all iterates remain in the positive cone, elements of y that correspond to inactive constraints in the primal QP are seen to rapidly (albeit asymptotically) decay to zero.

4. CASE STUDY

In this section, we apply the proposed *PQP* algorithm to three simulation examples – a stand-alone QP problem and two MPC problems – and compare its performance with the MATLAB QP solver `quadprog`. In our simulations, both the *PQP* algorithm and `quadprog` are implemented on MATLAB running on a 2.4 GHz Intel Core$^{\text{TM}}$2 CPU. For benchmarking purposes, MATLAB is restricted to handle only single-thread computations, thus ensuring a single core is used.

4.1 Example I: Stand-alone QP

The first simulation example is a two-dimensional QP problem, as formulated below:

$$\min_{X\in\mathbb{R}^2} \quad \frac{1}{2}X^T RX + h^T X \tag{11}$$

$$s.t. \quad 0 \le l_1 \le X(1) \le u_1$$

$$0 \le l_2 \le X(2) \le u_2 \tag{12}$$

where the state X is restricted in the first quadrant and subjected to box constraints. $R \in \mathbb{R}^2$ is chosen as a positive definite matrix, which satisfies the convergence requirements of the *PQP* algorithm. Comparing Eqs. (11)-(12) with Eqs. (3)-(4), we can see that this problem is specifically formulated for the use of *PQP*. It does not need to be transformed back and forth between primal and dual space, which makes *PQP* algorithm particularly efficient. Simulation experiments were performed using a wide range of parameter values ($R, h, u_1, u_2, l_1, and\ l_2$), and initial conditions (inside and outside the box in the non-negative cone) using both *PQP* algorithm and MATLAB `quadprog`. An average of speed-up of more than $10\times$ was achieved with *PQP* algorithm when compared with MATLAB `quadprog`.

4.2 Example II: MPC for Stochastic Control of a LTI System

In this example, we integrate *PQP* into an MPC scheme for a linear time-invariant (*LTI*) system (Wang and Boyd [2010])

$$\mathbf{X}_{k+1} = A\mathbf{X}_k + B\mathbf{U}_K + \omega_k \tag{13}$$

where $\mathbf{X}_k \in \mathbb{R}^{12}$,$\mathbf{U}_K \in \mathbb{R}^3$, and ω_k is an i.i.d random noise every element of which is uniformly distributed between $[-0.5, 0.5]$. The MPC solves a QP problem at each sample time, e.g., at sample time k:

$$\min_{\{\mathbf{X}_j, \mathbf{U}_{j-1}\}} \frac{1}{2}\sum_{j=k+1}^{k+N} \begin{pmatrix} \mathbf{X}_j^T R_x \mathbf{X}_j + H_x^T \mathbf{X}_j + \\ \mathbf{U}_{j-1}^T R_u \mathbf{U}_{j-1} + H_u^T \mathbf{U}_{j-1} \end{pmatrix} \tag{14}$$

$$s.t.\ \mathbf{X}_{j+1} = A\mathbf{X}_j + B\mathbf{U}_j$$

$$\mathbf{X}_{\min} \le \mathbf{X}_j \le \mathbf{X}_{\max}$$

$$\mathbf{U}_{\min} \le \mathbf{X}_j \le \mathbf{U}_{\max} \tag{15}$$

where $N = 25$ is the window size, and $\mathbf{X}_{\min} = -\mathbf{X}_{\max} = 2$, $\mathbf{U}_{\min} = -\mathbf{U}_{\max} = 1$. The cost parameters are chosen as $R_x = R_u = I, H_x = H_u = 0$, and it can be verified that the chosen parameters satisfy the conditions of convergence of Section 3, hence the *PQP* algorithm will converge to the optimal solution of the posed QP problem.

Eqs. (14)-(15) can be transformed into the form of Eqs. (1)-(2), where the state and constraint dimensions are 300 and 750 respectively. The MPC scheme was run for 100 sample points, solving Eq. (14) at each sample point using both *PQP* algorithm and the active-set based MATLAB function `quadprog`. Over multiple (> 20) simulation runs, the average *PQP* computation time is about 103.5 ms, while that of `quadprog` is about 531.8 ms per MPC window, indicating a speed-up on the order of $5\times$ for *PQP*.

The speed-up is not as significant as in the first example, because additional computation is needed for transformation between primal space and dual space. Further, since the number of constraints (750) is much larger than that of states (300), and *PQP* algorithm is applied in the dual space, where problem dimension actually increases to 750/750 from 300/750 in the primal problem.

4.3 Example III: MPC for Time-varying Reference Tracking of a LTI System

Finally, we apply *PQP* to an MPC scheme for solving a time-varying reference tracking problem. Consider a servo tracking problem for an LTI system with the following state and output equations at any time sample k:

$$x_{k+1} = Ax_k + Bu_k \tag{16}$$

$$y_k = Cx_k \tag{17}$$

where with the state vector $x_k \in \mathbb{R}^n$, the output vector $y_k \in \mathbb{R}^m$, the control vector $u_k \in \mathbb{R}^u$, and the system matrices A, B, and C are selected accordingly. Further,

without any loss of generality, we focus here on the tracking problem for this LTI system under the commonly used assumptions of detectability and reachability (Wang [2009]).

Our objective is to minimize within each MPC horizon (i) the tracking error between the output y_k (= position $x_{p,k}$) and the reference signal r_k, and also (ii) the control energy characterized by the magnitude of u_k. The constraints in the problem are chosen as (i) the state constraints: (a) the output positions $x_{p,k}$ are required to be always within a tolerance band $\Delta_{\max}$ around the reference, which we refer to as a *tube* constraint, (b) the velocities $x_{v,k}$ are bounded and (ii) the control constraints: the control u_k at each sample point must be within the actuator saturation limits $u_{\min}$ and $u_{\max}$.

An MPC scheme is proposed for this problem to solve the QP problem at any sample time k as

$$\min_{U_k=[u_k,\ldots u_{k+N-1}]} J(U_k, x_k) \tag{18}$$

where

$$\begin{aligned} J(U_k, x_k) = \sum_{i=1}^{N-1} \{ &(y_{k+i} - r_{k+i})^T S (y_{k+i} - r_{k+i}) + \\ &+ u_{k+i}^T R u_{k+i} \} \\ &+ (x_{k+N} - x_f)^T P (x_{k+N} - x_f) \end{aligned} \tag{19}$$

s.t. $\forall i \in [1, N-1]$

$$x_{k+i} = A x_{k+i-1} + B u_{k+i-1} \tag{20}$$

$$r_{k+i} - \triangle_{\max} \leq x_{p,k+i} \leq r_{k+i} + \triangle_{\max} \tag{21}$$

$$v_{\min} \leq x_{v,k+i} \leq v_{\max} \tag{22}$$

$$u_{\min} \leq u_{k+i} \leq u_{\max} \tag{23}$$

where $S \succ 0$, $R \succ 0$, are weights on the tracking error cost and the control cost, respectively, in the bracketed term of the horizon cost function, and the last term in the horizon cost function is a terminal cost representing a weighted deviation of the terminal state x_{n+N} in the window from a desired terminal state x_f. As is well-known and established, the terminal cost is added to ensure asymptotic stability of the unconstrained closed-loop system (Wang [2009], Borrelli [2003]).

The above MPC problem can be transformed to the general primal QP problem of the form given in Eqs. (1)-(2) (see Appendix for details), allowing us to apply the proposed PQP algorithm to solve it.

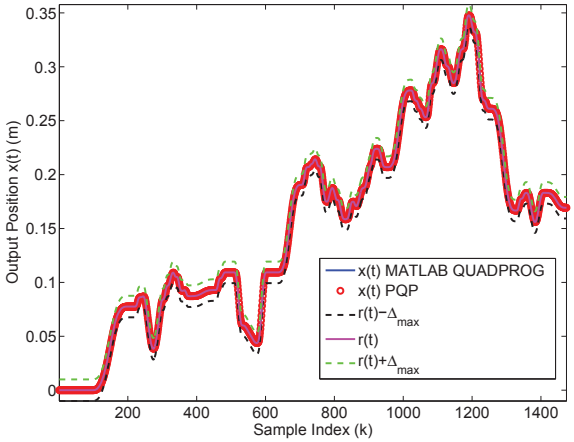

Fig. 1. Comparison of position signal $x_{p,k}$ generated by PQP when compared with that by `quadprog` command. A window size $N = 25$ was used. The PQP computation time for this simulation is about 12.1 ms, while that for MATLAB `quadprog` is 703.1 ms, indicating a speed-up of about 60× while still maintaining the required precision.

In what follows, we will have one simulation example of this problem. In this example, the system matrices in a continuous-time representation are selected as:

$$A_c = \begin{bmatrix} 0 & 0 \\ 0 & -0.1 \end{bmatrix}, \; B_c = \begin{bmatrix} 0 \\ 0.02 \end{bmatrix} \tag{24}$$
$$C_c = [\, 1 \;\; 0 \,]$$

where all numerical values are in SI units, and the system discretization is performed at a sampling frequency of 1.125 kHz to generate the discrete-time system matrices A, B, C of Eq. (16).

The MPC problem was solved for 1500 time samples, with different sizes for the horizon window N. An example set of window sizes N is $[25, 50, 75, 100]$. A relative tolerance of 10^{-6} for the computed cost function was used as a termination condition for the `quadprog` algorithm. To match the precision of the tracked output computed from both the algorithms, a relative tolerance of 10^{-6} was found to be adequate as a termination condition on the cost function for the PQP algorithm.

An illustrative sample subset of results obtained with either algorithm is shown in Figs. 1 and 2. Fig. 1 shows the tracked output of the system, $x_{p,k}$. As seen in the zoomed version showed in Fig. 2, the proposed MPC scheme is indeed feasible, as there always exists admissible controls that maintain system position $x_{p,k}$ within the tolerance

Window Size N	QP Size (States/ Constraints)	Avg. time (ms) PQP	Avg. time (ms) `quadprog`
25	50/150	11.88	697.69
50	100/300	25.02	782.14
75	150/450	45.96	830.13
100	200/600	75.00	858.80

Table 1. Comparison of computation times between the proposed PQP algorithm and `quadprog` for different window sizes

tube around the given reference signal. Further, the MPC solutions obtained from PQP and `quadprog` agree well within the specified relative tolerance. A detailed error analysis, exploring the effects of MPC parameters on the performance of PQP in terms of the precision and rate of convergence is currently under study. For proof of linear convergence rate and discussion, please refer to Brand and Chen [2011].

Table 1 shows the average computation time per MPC window for PQP and `quadprog` for different window sizes. The average computation time was computed over the total horizon length of 1500 samples. On average, PQP shows about more than $10\times$ speed-up over `quadprog`.

All three simulation examples detailed above show significant ($5\times$ to $10\times$) speed-ups of the proposed PQP algorithm over existing MATLAB QP solver `quadprog` even when it is implemented sequentially on a 2.4 GHz Intel Core^{TM}2 CPU (and slowed by Matlab's cumbersome `for` loops). Further speed-up is expected from a parallel implementation; Brand and Chen [2011] reports that GPU implementations further multiply the speed-up by a factor of $20\times - 80\times$. SIMD machines, which are perfectly suited to the data-flow of the update, may offer significantly greater speed-ups.

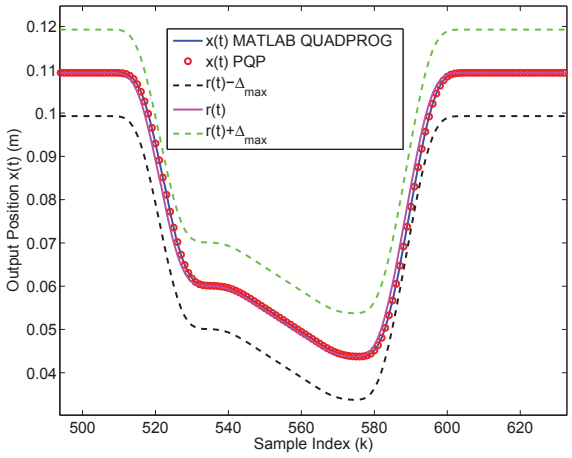

Fig. 2. Zoomed image showing tracking plots of Fig. 1 in detail, with the PQP solution closely overlapping with the `quadprog` solution. A tolerance band of $\Delta_{max} = 0.01\ m$ was used.

It must be noted that the presented version of the PQP algorithm has not taken advantage of special structures in the MPC framework, which are the main sources of speed-ups reported in the literature (see Rao et al. [2004],Wang and Boyd [2010]). New variations of the proposed PQP algorithm, exploiting special structures of the MPC framework, have resulted in additional $5\times$ to $10\times$ speed-ups. These PQP algorithm variations and their effects will be covered in a future paper from our group.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a simple, easy-to-use algorithm with multiplicative updates for solving QP problems. The algorithm is parallelizable and has been demonstrated to achieve speed-ups for MPC schemes, while still maintaining the required computational precision. We established and proved the convergence conditions of the PQP algorithm. Simulation results illustrate significant speed-up of the proposed algorithm over existing QP solvers, even when it is implemented in sequential fashion. Future work will focus on exploiting primal problem sparsity, and implementing the PQP algorithm using multi-processor parallel computing devices, thus exploiting its full parallelizability.

6. APPENDIX

Derivation of Eq. (1) for MPC reference-tracking problem:
For the system defined in Eq. (20), consider the problem of tracking a reference vector Γ_k given as:

$$\Gamma_k = [r_k, r_{k+1}, ..., r_{k+N}]^T \tag{25}$$

Over the horizon of window length N, the following vectors are defined:
(i) a stacked state vector $\mathbf{X}_k = [x_{k+1}, ..., x_{k+N}]^T$
(ii) a stacked output vector $\mathbf{Y}_k = [y_{k+1}, ..., y_{k+N}]^T$, and
(iii) a stacked control vector $\mathbf{U}_k = [u_k, ...u_{k+N-1}]^T$.

Using Eq. (16), we have the set of state equations stacked over the horizon window N as follows:

$$\mathbf{X}_k = \Psi x_k + \Omega \mathbf{U}_k \tag{26}$$

where,

$$\Psi = \left[A,\ A^2,\ ...,\ A^N\right]^T \in \mathbb{R}^{nN\times n}$$

$$\Omega = \begin{bmatrix} B & 0 & \cdot & \cdot & \cdot & 0 \\ AB & B & 0 & \cdot & \cdot & \cdot \\ A^2B & AB & B & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & B & 0 \\ A^{N-1}B & A^{N-2}B & \cdot & \cdot & AB & B \end{bmatrix} \in \mathbb{R}^{nN\times uN}$$

Similarly, we have the output stacked as follows:

$$\mathbf{Y}_k = \Phi \mathbf{X}_k$$

where

$$\Phi = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & C \end{bmatrix} \in \mathbb{R}^{mN\times nN}$$

In implementing the MPC formulation of Eq. (18), the tuning parameters include (i) weight S on the tracking error cost, (ii) weight R on the control cost, (iii) weight P on the error cost between terminal state and (iv) the desired terminal state x_f. Specifically, it can be shown that asymptotic stability of the unconstrained closed loop system results from using $P = P^T \succ 0$ as the solution of the discrete-time algebraic Riccati equation (Borrelli et al. [2010]), given as follows:

$$\begin{aligned} P &= A^T P A + \overline{S} - K^T (R + B^T P B) K \\ K &= (R + B^T P B)^{-1} B^T P A \end{aligned} \tag{27}$$

where $\overline{S} = C^T S C;\ S \succ 0$ and $R \succ 0$ are the user-defined tuning penalties on the states and controls, respectively, in the infinite-horizon optimal control formulation solved by the algebraic Riccati equation.

For the above-defined P, Eq. (26) can be used to reduce the primal QP problem posed in Eq. (19) to the general form of Eq. (1)

$$\min_{U_n = \left[u_n, \ldots u_{n+N-1}\right]^T} \frac{1}{2} U_n^T \mathcal{Q} U_n + \mathcal{H}^T U_n$$
$$V U_n \leq W$$

with $\mathcal{Q} \in \mathbb{R}^{uN\times uN}$ and $\mathcal{H} \in \mathbb{R}^{uN\times 1}$, respectively, appearing in the primal cost function, given as follows:

$$\mathcal{Q} = 2L_1 + 2\Omega^T L_2 \Omega \tag{28}$$

$$\mathcal{H} = 2x_n^T \Psi^T L_2 \Omega - 2\Gamma_k^T L_3 \Phi \Omega - 2x_f^T P L_4 \Omega \tag{29}$$

where the matrices $L_1 \in \mathbb{R}^{uN\times uN}$, $L_2 \in \mathbb{R}^{nN\times nN}$, $L_3 \in \mathbb{R}^{mN\times mN}$, and $L_4 \in \mathbb{R}^{n\times nN}$ are defined as follows:

$$L_1 = \begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & R \end{bmatrix};\quad L_2 = \begin{bmatrix} \overline{S} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \overline{S} & 0 \\ 0 & \ldots & 0 & P \end{bmatrix};$$

$$L_3 = \begin{bmatrix} S & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & S & 0 \\ 0 & \ldots & 0 & 0 \end{bmatrix};\quad L_4 = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & I_{n\times n} \end{bmatrix};$$

and, $V \in \mathbb{R}^{uN\times uN}$ and $W \in \mathbb{R}^{uN\times uN}$, respectively, appearing in the primal constraints, given as follows:

$$\begin{aligned} V &= [\Omega,\ -\Omega,\ I,\ -I]^T \\ W &= [\mathbf{X}_{k\,\max} - \Psi x_k,\ -\mathbf{X}_{k\,\min} + \Psi x_k,\ \mathbf{U}_{k\,\max},\ -\mathbf{U}_{k\,\min}]^T \end{aligned} \tag{30}$$

For our problem of reference tracking with *tube* constraints imposed on position outputs, velocity constraints, and control constraints, the constraint limits appearing in Eq. (30) above are given as follows:

$$\mathbf{X}_{k\,\max} = \begin{bmatrix} r_{k+1} + \triangle_{\max}, v_{\max}, ..., \\ r_{k+N} + \triangle_{\max}, v_{\max} \end{bmatrix}^T$$

$$\mathbf{X}_{k\,\min} = \begin{bmatrix} r_{k+1} - \triangle_{\max}, v_{\min} ..., \\ r_{k+N} - \triangle_{\max}, v_{\min} \end{bmatrix}^T$$

$$\mathbf{U}_{k\,\max} = [u_{\max}, ..., u_{\max}]^T;\qquad \mathbf{U}_{k\,\min} = [u_{\min}, ..., u_{\min}]^T$$

This completes our derivation of the QP formulation for the MPC servo problem of reference tracking.■

It should be noted here that for $P, R, S \succ 0$, we have $L_1 \succ 0$ and $L_2 \succcurlyeq 0$. These relations, along with Eq. (28), imply that $\mathcal{Q} \succ 0$, i.e. $\mathcal{Q}$ is positive definite, which validates Assumption 1 used in Section 2 in our *PQP* formulation.

Further, from Eqs. (28)-(29), it is clear that the only term in the cost function of Eq. (1) that depends explicitly on the reference is the linear term containing $\mathcal{H}$. However, $\mathcal{Q}$ does not have an explicit dependence on the time-varying reference; $\mathcal{Q}$ is predetermined by the physical properties of the system (i.e. the state matrices) and penalties selected for the terminal cost function. More speed-up of the current algorithm can be achieved by exploiting this feature of the MPC formulation.

As for the constraints in the primal problem, since we have *tube* constraints on the position output of the system, the state constraints also change with changes in the reference.

The dual formulation used in Section 2 conveniently allows handling these general primal constraints by mapping them into dual variables in the dual space.

REFERENCES

Bageshwar, V., Garrard, W., and Rajamani, R. (2004). Model predictive control of transitional maneuvers for adaptive cruise control vehicles. *IEEE Transactions on Vehicular Technology*, 53 (5), 1573–1585.

Benzi, M., Golub, G.H., and Liesen, J. (2005). Numerical solution of saddle point problems. *Acta Numerica*, 1, 1–137.

Bertero, M., Lanteri, H., and Zanni, L. (2007). Iterative image reconstruction: A point of view. In *Proceedings of the Interdisciplinary Workshop on Mathematical Methods in Biomedical Imaging and Intensity-Modulated Radiation Therapy (IMRT)*.

Borrelli, F. (2003). *Constrained Optimal Control of Linear and Hybrid Systems*. Sringer, Lecture Notes in Control and Information Sciences.

Borrelli, F., Bemporad, A., and Morari., M. (2010). Predictive control. *Springer, In Press. http://www.mpc.berkeley.edu*.

Brand, M. and Chen, D. (2011). Parallel quadratic programming for image processing. *In Reviews*.

Camacho, E.F. and Bordons, C. (2004). *Model Predictive Control*. Springer.

Dostl, Z. (2009). *Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities*. Springer Publishing Company, Incorporated.

Eggermont, P. (1990). Multiplicative iterative algorithms for convex programming. *J. Linear Algebra and Applications*, 130, 25–42.

Ferreau, H. (2006). An online active set strategy for fast solution of parametric quadratic programs with applications to predictive engine control. Master's Thesis, University of Heidelberg.

Gondzio, J. and Grothey, A. (2007). Parallel interior-point solver for structured quadratic programs: Application to financial planning problems. *Annals of Operations Research*, 152, 319–339.

Gondzio, J. and Sarkissian, R. (2000). Parallel interior-point solver for structured linear programs. *Mathematical Programming*, 96, 561–584.

Heath, W. and Wills, A. (2007). Zames-falb multipliers for quadratic programming. *IEEE Transactions on Automatic Control*, 52, 1948–1951.

Kirk, D. and Hwu, W. (2010). *Programming Massively Parallel Processing*. Morgan Kaufmann.

Lee, D.D. and Seung, H.S. (1999). Learning the parts of objects with nonnegative matrix factorization. *Nature*, 401, 788–791.

Luo, Z.Q. and Tseng, P. (1992). On the linear convergence of descent methods for convex essentially smooth minimization. *SIAM J. Control and Optimization*, 30(2), 408–425.

Milman, R. and Davison, E.J. (2008). A fast mpc algorithm using nonfeasible active set methods. *Journal of optimization theory and applications*, 139, 591–616.

Negenborn, R.R., Schutter, B.D., and Hellendoorn, H. (2006). Multi-agent model predictive control for transportation networks: Serial versus parallel schemes. *Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'2006)*, 339–344.

Qin, S.J. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7), 733–764.

Rao, C.V., Wright, S.J., and Rawlings, J.B. (2004). Application of interior point methods to model predictive control. *Journal of optimization theory and applications*, 90(3), 723–757.

Rawlings, J.B. and Mayne, D.Q. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Publishing, LLC.

Richter, S., Jones, C., and Morari, M. (2009). Real-time input-constrained mpc using fast gradient methods. In *Proc. IEEE Conference on Decision and Control*, 7387–7393. Shanghai, China.

Rossiter, J.A. (2003). *Model-based Predictive Control: A Practical Approach*. Prentice Hall International.

Sha, F. et al. (2007). Multiplicative updates for nonnegative quadratic programming. *Neural Computation*, 19(8), 2004–2031.

Shepp, L. and Vardi, Y. (1982). Maximum likelihood reconstruction for emission tomography. *IEEE Trans. Medical Imaging 1*, 1, 113–122.

Stratton, J., Stone, S., and Hwu, W. (2008). Mcudal an efficient implementation of cuda kernels for multi-core cpus. In *Proc. 21st International Workshop on Languages and Compilers for Parallel Computing*, 16–30.

Wang, L. (2009). *Model Predictive Control System Design and Implementation using MATLAB*. Springer-Verlag.

Wang, Y. and Boyd, S. (2010). Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2), 267–278.

Zeilinger, M.N., Jones, C., and Morari, M. (2009). Real-time mpc-stability through robust design. In *Proc. IEEE Conference on Decision and Control*, 3980–3986. Shanghai, China.