Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2012, Article ID 729748, 13 pages
doi:10.1155/2012/729748

Research Article

Efficient Model Predictive Algorithms for Tracking of Periodic Signals

Yun-Chung Chu and Michael Z. Q. Chen

Department of Mechanical Engineering, The University of Hong Kong, Hong Kong

Correspondence should be addressed to Michael Z. Q. Chen, mzqchen@hku.hk

Received 30 June 2011; Accepted 31 August 2011

Academic Editor: Baocang Ding

This paper studies the design of efficient model predictive controllers for fast-sampling linear time-invariant systems subject to input constraints to track a set of periodic references. The problem is decomposed into a steady-state subproblem that determines the optimal asymptotic operating point and a transient subproblem that drives the given plant to this operating point. While the transient subproblem is a small-sized quadratic program, the steady-state subproblem can easily involve hundreds of variables and constraints. The decomposition allows these two subproblems of very different computational complexities to be solved in parallel with different sampling rates. Moreover, a receding horizon approach is adopted for the steady-state subproblem to spread the optimization over time in an efficient manner, making its solution possible for fast-sampling systems. Besides the conventional formulation based on the control inputs as variables, a parameterization using a dynamic policy on the inputs is introduced, which further reduces the online computational requirements. Both proposed algorithms possess nice convergence properties, which are also verified with computer simulations.

1. Introduction

One of the most attractive features of model predictive control (MPC) is its ability to handle constraints [1]. Many other control techniques are conservative in handling constraints, or even try to avoid activating them, thus, sacrificing the best performance that is achievable. MPC, on the contrary, tends to make the closed-loop system operate near its limits and hence produces far better performance. This property of MPC gives it the strength in practice, leading to a wide acceptance by the industry.

A very good example of system operating near its limits is a plant being driven by periodic signals to track periodic references. Under this situation, some of the system constraints will be activated repeatedly, and the optimal operating control signal is far from trivial. Just clipping the control signal to fit into the system constraints produces inferior performance typically. And the loss being considered here is not just a transient loss due to sudden disturbances, but indeed a steady-state loss due to a suboptimal operating point. Therefore, the loss is on long term and severe.

On the other hand, the successful real-life applications of MPC are mostly on systems with slower dynamics such as industrial and chemical processes [2]. The reason is simply that MPC requires a constrained optimization to be carried out online in a receding horizon fashion [3, 4]. Therefore, to apply MPC to fast-sampling systems, the computational power needed will be substantial. In any case, because of its great success in slow-sampling systems, the trend to extend MPC to fast-sampling systems is inevitable, and many recent researches have been carried out to develop efficient methods to implement MPC in such cases. While some of these works focus on unloading the computational burdens [5–9], others emphasize on code optimization [10–12] and new algorithmic paradigms [13–17].

If MPC is applied to the tracking of periodic signals in a receding horizon fashion, the horizon length will be related to the period length, and a long period will imply an online optimization problem of many variables and constraints. For a fast-sampling system, it is essentially an attempt to solve a very big computational problem within a very small time frame. In this paper, we shall analyze the structure of this

problem and then propose two efficient algorithms for the task. They aim to make the application of MPC to a fast-sampling system possible by a slight sacrifice on the transient performance, but the optimal or near-optimal steady-state performance of periodic tracking will be maintained.

In Section 2, the mathematical formation of the problem will be presented. The two algorithms, one based on the concept of receding horizon quadratic programming and the other based on the idea of dynamic MPC policy, will be presented in Sections 3 and 4, respectively. A comment on the actual implementation will be given in Section 5, followed by some computer simulations in Section 6 to illustrate several aspects of the proposed algorithms. Finally, Section 7 concludes the paper.

To avoid cumbersome notations like $u(k \mid k), u(k+1 \mid k), \ldots, u(k+N_u-1 \mid k)$, the MPC algorithms in this paper will only be presented as if the current time is $k = 0$, and we shall write $u(0), u(1), \ldots, u(N_u - 1)$ instead. The reader is asked to bear in mind that the algorithm is actually to be implemented in a receding horizon fashion.

2. Problem Formulation

Consider a linear time-invariant plant subject to a periodic disturbance:

$$x^+ = Ax + B_1 w + B_2 u, \tag{1}$$

$$y = Cx + D_1 w + D_2 u, \tag{2}$$

$$x(0) = x_0, \tag{3}$$

where the superscript + denotes the time-shift operator, that is,

$$x^+(k) = x(k+1), \tag{4}$$

and the disturbance w is measurable and periodic with period N_p. The control objective is to construct a control signal u such that the plant output y will track a specific periodic reference r of the same period N_p asymptotically with satisfactory transient performance. The control input u is also required to satisfy some linear inequality constraints (e.g., to be within certain bounds). The reference r is not necessarily fixed but may be different for different disturbance w. (For that reason, it may be more appropriate to call w an exogenous signal rather than a disturbance).

The algorithms developed in this paper are motivated by the following situations:

(1) the period N_p is very long compared with the typical transient behaviours of the closed-loop system;

(2) the linear inequality constraints on u are persistently active, that is, for any given $\widetilde{k}$, there exists a $k > \widetilde{k}$ such that $u(k)$ will meet at least one of the associated linear *equality* constraints;

(3) there is not sufficient computational power to solve the associated quadratic program completely within one sampling interval unless both the control horizon and the prediction horizon are much shorter than N_p.

As a matter of fact, without the above considerations and restrictions, the problem is not very challenging and can be tackled with standard MPC approaches for linear systems.

The underlying idea of the approach proposed in this paper is that since the transient behaviour of the closed-loop system is expected to be much shorter than the period N_p, we should decompose the original control problem into two: one we call the steady-state subproblem and the other we call the transient subproblem. Hence, the transient subproblem can be solved with a control horizon and a prediction horizon much shorter than N_p. While the steady-state subproblem is still very computationally intensive and cannot be solved within one sampling interval, it is not really urgent compared with the transient subproblem, and its computation can be spread over several sampling intervals. Indeed, the two subproblems need not be synchronized even though the transient subproblem depends on the solution to the steady-state subproblem due to the coupled input constraints. The former will utilize the latter whenever the latter is updated and made available to the former. It is only that the transient control will try to make the plant output y track a *suboptimal* reference if the optimal steady-state control is not available in time.

Now let us present the detailed mathematical formulation of our proposed method. Naturally, since both w and r are periodic with period N_p, the solution u should also be periodic of the same period asymptotically, that is, there should exist a periodic signal $u_s(k)$ such that

$$\lim_{k\to\infty} (u(k) - u_s(k)) = 0. \tag{5}$$

Let x_s and y_s be the corresponding asymptotic solutions of x and y, and they obviously satisfy the dynamics inherited from (1) and (2):

$$\begin{aligned} x_s^+ &= Ax_s + B_1 w + B_2 u_s, \\ y_s &= Cx_s + D_1 w + D_2 u_s, \\ x_s(0) &= x_s\left(N_p\right). \end{aligned} \tag{6}$$

Ideally, we want $y_s = r$ but this might not be achievable when u_s is required to satisfy the specific linear inequality constraints. Therefore, following the spirit of MPC, we shall find u_s, such that

$$J_s = \sum_k e_s(k)^{\mathrm{T}} Q e_s(k) \tag{7}$$

is minimized for some positive definite matrix Q, where $e_s(k)$ is the asymptotic tracking error defined by

$$e_s = y_s - r, \tag{8}$$

and the summation in (7) is over one period of the signals. This is what we call the steady-state subproblem. In Sections 3 and 4 below, we shall present two different approaches to this steady-state subproblem, with an emphasis on their computational efficiencies.

Once the steady-state signals are known, the transient signals defined by

$$u_t = u - u_s, \qquad x_t = x - x_s, \qquad y_t = y - y_s, \tag{9}$$

satisfy the dynamics

$$\begin{aligned} x_t^+ &= Ax_t + B_2 u_t, \\ y_t &= Cx_t + D_2 u_t, \\ x_t(0) &= x_0 - x_s(0), \end{aligned} \tag{10}$$

derived from (1)–(3), subject to the original linear inequality constraints being applied to $u_t(k) + u_s(k)$. Since the control horizon and the prediction horizon for this transient subproblem are allowed to be much shorter than N_p, it can be tackled with existing MPC algorithms.

It is important to note that in this steady-state/transient decomposition, the steady-state control u_s is actually a *feedforward* control signal determined from w and r, whereas the transient control u_t is a *feedback* control signal depending on x. As an unstable plant can only be stabilized by feedback, but the main interest of the current paper is the computational complexity of the steady-state subproblem, we shall not discuss in depth the stabilization issue, which has been studied quite extensively in the MPC literature. Typically, stabilizability of a constrained system using MPC would be cast as the feasibility of an associated optimization problem. For the numerical example in Section 6, we shall conveniently pick a plant where A is already stable, and hence the following quadratic cost may be adopted for the transient subproblem:

$$J_t = \sum_{k=0}^{N_u-1} \left(y_t(k)^{\mathrm{T}} Q y_t(k) + u_t(k)^{\mathrm{T}} R u_t(k) \right) + x_t(N_u)^{\mathrm{T}} P_T x_t(N_u), \tag{11}$$

where N_u is the control horizon with $N_u \ll N_p$, Q and R are chosen positive definite matrices, and P_T is the (weighted) observability gramian obtained from the Lyapunov equation

$$A^{\mathrm{T}} P_T A - P_T + C^{\mathrm{T}} Q C = 0. \tag{12}$$

The minimization of J_t is simply a standard quadratic program over $u_t(0), u_t(1), \ldots, u_t(N_u - 1)$ for a given $x_t(0)$. The situation will be more complicated when A is not stable, but one well-known approach is to force the unstable modes to zero at the end of the finite horizon [18].

Remark 1. Essentially, the choice of the cost function (11) with P_T from (12) for a stable A means that the projected control action after the finite horizon is set to zero, that is, $u_t(k) = 0$ for $k \geq N_u$ since the "tail" of the quadratic cost is then given by

$$\begin{aligned} \sum_{k=N_u}^{\infty} y_t(k)^{\mathrm{T}} Q y_t(k) &= \sum_{k=N_u}^{\infty} x_t(k)^{\mathrm{T}} C^{\mathrm{T}} Q C x_t(k) \\ &= x_t(N_u)^{\mathrm{T}} P_T x_t(N_u). \end{aligned} \tag{13}$$

This terminal policy is valid because the steady-state subproblem has already required that u_s satisfies the linear inequality constraints imposed on u. Hence, J_t is obviously a Lyapunov function which will be further decreased by the receding horizon implementation when the future control $u_t(N_u)$ turns into an optimization variable from zero.

Remark 2. We have deliberately omitted the R-matrix in the steady-state cost J_s in (7). The reason is simply that we want to recover the standard linear solution (for perfect asymptotic tracking) as long as u_s does not hit any constraint.

3. Steady-State Subproblem: A Receding Horizon Quadratic Programming Approach

When the periodic disturbance w is perfectly known, the steady-state subproblem is also a conceptually simple (but computationally high-dimensional) quadratic program. One way to know w is simply to monitor and record it over one full period. This, however, does not work well if w is subject to sudden changes. For example, the plant to be considered in our simulations in Section 6 is a power quality device called Unified Power Quality Conditioner [19], where w consists of the supply voltage and the load current of the power system, and both may change abruptly if there are supply voltage sags/swells and variations in the load demands. Indeed, the main motivation of the receding horizon approach in MPC is that things never turn out as expected and the control signal should adapt in an autonomous manner. If the suddenly changed disturbance w can be known precisely only after one full period of observation, the transient response of the steady-state subproblem (not to be confused with the transient subproblem described in Section 2) will be unsatisfactory.

One way to overcome this is to introduce an exogenous model for the signals w and r, as adopted in [19]. Specifically, we construct a state-space model:

$$v^+ = A_v v, \tag{14}$$

$$w = C_w v, \tag{15}$$

$$r = C_r v, \tag{16}$$

and assume that both w and r are generated from this model. Since w and r are periodic with period N_p, we have

$$A_v^{N_p} = I. \tag{17}$$

One simple (but not the only) way to construct A_v, as demonstrated similarly in [19] in the continuous-time case, is to make A_v a block-diagonal matrix with each block taking the form:

$$\begin{bmatrix} \cos(n\omega T_s) & -\sin(n\omega T_s) \\ \sin(n\omega T_s) & \cos(n\omega T_s) \end{bmatrix}, \tag{18}$$

where n is an integer, T_s is the sampling time and $\omega T_s \times N_p = 2\pi$. Then the matrices C_w and C_r are just to sum up their respective odd components of v. This essentially performs a Fourier decomposition of the signals w and r, and hence their approximations by $C_w v$ and $C_r v$ will be arbitrarily good when more and more harmonics are included in the model.

Based on the exogenous model (14)–(17), an observer can be easily constructed to generate (an estimate of) v from the measurements of w and r. From $v(0)$, the model (14)–(17) can then generate predictions of $w(0), w(1), \ldots, w(N_p - 1)$ and $r(0), r(1), \ldots, r(N_p - 1)$, and these can be used to find $u_s(0), u_s(1), \ldots, u_s(N_p - 1)$ by the quadratic program. The use of the exogenous model (14)–(17) typically allows the changed w and r to be identified much sooner than the end of one full period.

The quadratic program for the steady-state subproblem can be written as follows:

$$\min_{\mathbf{u}_s(0)} J_s,$$

$$J_s := \begin{bmatrix} \mathbf{u}_s(0) \\ v(0) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{H} & \mathbf{F} \\ \mathbf{F}^{\mathrm{T}} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \mathbf{u}_s(0) \\ v(0) \end{bmatrix},$$

$$\mathbf{u}_s(0) := \begin{bmatrix} u_s(0) \\ u_s(1) \\ \vdots \\ u_s\left(N_p - 1\right) \end{bmatrix}, \tag{19}$$

$$\text{subject to} \quad \mathbf{a}_j^{\mathrm{T}} \mathbf{u}_s(0) \le \mathbf{b}_j, \qquad j = 1, 2, \ldots, N_m, \tag{20}$$

where N_m is the total number of linear inequality constraints. Note that since we assume only input but not state constraints for our original problem, (20) does not depend on $v(0)$ and, hence, the feasibility of any $\mathbf{u}_s(0)$ remains the same even if there is an abrupt change in $v(0)$ (i.e., if $v(0)$ is different from the predicted value from (14) and $v(-1)$). Furthermore, the active set of constraints remains the same.

Definition 3. The active set $\mathcal{A}(\mathbf{u}_s(0))$ of any feasible $\mathbf{u}_s(0)$ satisfying (20) is the subset of $\{1, 2, \ldots, N_m\}$ such that $j \in \mathcal{A}(\mathbf{u}_s(0))$ if and only if $\mathbf{a}_j^{\mathrm{T}} \mathbf{u}_s(0) = \mathbf{b}_j$.

Next, we present a one-step active set algorithm to solve the quadratic program (19)-(20) partially.

Algorithm 4. Given an initial feasible $\mathbf{u}_s(0)$ and a working set $\mathcal{W}_0 \subset \mathcal{A}(\mathbf{u}_s(0))$. Let the set of working constraints

$$\mathbf{a}_j^{\mathrm{T}} \mathbf{u}_s(0) \le \mathbf{b}_j, \quad j \in \mathcal{W}_0, \tag{21}$$

be represented by

$$\mathbf{A}_0 \mathbf{u}_s(0) \le \mathbf{b}_0, \tag{22}$$

where the inequality sign applies componentwise, that is, each row of $\mathbf{A}_0$, $\mathbf{b}_0$ represents a working constraint in (21).

(1) Compute the gradient

$$\mathbf{g}_0 = \mathbf{H}\mathbf{u}_s(0) + \mathbf{F}v(0), \tag{23}$$

and the null space of $\mathbf{A}_0$, denoted $\mathbf{Z}_0$ by

$$\mathbf{A}_0 \mathbf{Z}_0 = 0. \tag{24}$$

If $\mathbf{Z}_0^{\mathrm{T}} \mathbf{g}_0 \approx 0$, go to step (5).

(2) Compute a search direction $\mathbf{w}_0 = \mathbf{Z}_0 \hat{\mathbf{w}}_0$, where

$$\left(\mathbf{Z}_0^{\mathrm{T}} \mathbf{H} \mathbf{Z}_0\right) \hat{\mathbf{w}}_0 + \mathbf{Z}_0^{\mathrm{T}} \mathbf{g}_0 = 0. \tag{25}$$

(3) Let

$$\alpha_0 = \min_{\substack{j \in \mathcal{A}(\mathbf{u}_s(0)) \setminus \mathcal{W}_0 \\ \text{s.t. } \mathbf{a}_j^{\mathrm{T}} \mathbf{w}_0 > 0.}} \frac{\mathbf{b}_j - \mathbf{a}_j^{\mathrm{T}} \mathbf{u}_s(0)}{\mathbf{a}_j^{\mathrm{T}} \mathbf{w}_0}. \tag{26}$$

(4) If $\alpha_0 \ge 1$, go to step (5). Otherwise, update $\mathbf{u}_s(0)$ to $\mathbf{u}_s^*(0)$ by

$$\mathbf{u}_s^*(0) = \mathbf{u}_s(0) + \alpha_0 \mathbf{w}_0, \tag{27}$$

and add a (blocking) constraint to $\mathcal{W}_0$ to form a new working set $\mathcal{W}_0^*$ according to the method described in Remark 7 below. Quit.

(5) Update $\mathbf{u}_s(0)$ to $\mathbf{u}_s^*(0)$ by

$$\mathbf{u}_s^*(0) = \mathbf{u}_s(0) + \mathbf{w}_0. \tag{28}$$

Compute the Lagrange multiplier λ_0 from

$$\mathbf{A}_0^{\mathrm{T}} \lambda_0 + \mathbf{g}_0 = 0 \tag{29}$$

to see whether any component of λ_0 is negative. If yes, remove one of the constraints corresponding to a negative component of λ_0 from $\mathcal{W}_0$ to form a new working set $\mathcal{W}_0^*$ according to the method described in Remark 7 below. Quit.

Algorithm 4 can be interpreted as follows. It solves the equality-constrained quadratic program:

$$\min_{\mathbf{u}_s'(0)} J_s',$$

$$J_s' := \begin{bmatrix} \mathbf{u}_s'(0) \\ v(0) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{H} & \mathbf{F} \\ \mathbf{F}^{\mathrm{T}} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \mathbf{u}_s'(0) \\ v(0) \end{bmatrix}, \tag{30}$$

$$\text{subject to} \quad \mathbf{a}_j^{\mathrm{T}} \mathbf{u}_s'(0) = \mathbf{b}_j, \quad j \in \mathcal{W}_0,$$

and then searches along the direction

$$\mathbf{w}_0 = \mathbf{u}_s'(0) - \mathbf{u}_s(0), \tag{31}$$

until it is either blocked by a constraint not in $\mathcal{W}_0$ (step (4)) or the optimal $\mathbf{u}_s'(0)$ is reached (step (5)). This is indeed a single step of the standard active set method (the null-space approach) for quadratic programming [20, 21] except for the modifications that will be detailed in Remark 7 below. In other words, if we apply Algorithm 4 repeatedly to the new $\mathbf{u}_s^*(0)$, $\mathcal{W}^*(0)$, it will converge to the solution of the original inequality-constrained quadratic program (19)-(20) within a finite number of steps, and the cost function J_s is strictly decreasing. However, here we only apply a single step of the active set method due to the limited computational power available within one sampling interval. Furthermore, we do not even assume that the single step of computation

can be completed within T_s. Let N_aT_s be the time required or allowed to carry out Algorithm 4. To complete the original quadratic program (19)-(20) in a receding horizon fashion, we need to forward $\mathbf{u}_s^*(0)$, $\mathcal{W}^*(0)$ to $\mathbf{u}_s(N_a)$, $\mathcal{W}(N_a)$ by rotating the components of $\mathbf{u}_s^*(0)$ by an amount of N_a since it is supposed to be periodic:

$$\begin{bmatrix} u_s(N_a) \\ u_s(N_a+1) \\ \vdots \\ u_s\left(N_p-1\right) \\ u_s\left(N_p\right) \\ u_s\left(N_p+1\right) \\ \vdots \\ u_s\left(N_a+N_p-1\right) \end{bmatrix} = \begin{bmatrix} u_s^*(N_a) \\ u_s^*(N_a+1) \\ \vdots \\ u_s^*\left(N_p-1\right) \\ u_s^*(0) \\ u_s^*(1) \\ \vdots \\ u_s^*(N_a-1) \end{bmatrix}. \tag{32}$$

Obviously, Algorithm 4 will then continue to solve an equivalent quadratic program as long as v strictly follows the exogenous dynamics (14). Hence we have the following convergence result.

Proposition 5. *Algorithm 4 together with the rotation of the components of* $\mathbf{u}_s^*(0)$ *in* (32) *will solve the quadratic program* (19)-(20) *in finite time as long as* $v(k)$ *satisfies the exogenous dynamics* (14).

Proof. From the argument above it is easy to see that as long as $v(k)$ follows the dynamics (14), the algorithm is consistently solving essentially the same quadratic program. So it remains to check that the convergence proof of the standard active set algorithm remains valid despite the modifications we shall detail in Remark 7, which is indeed the case. □

Of course, the most interesting feature of the receding horizon approach is that the solution will adapt autonomously to the new quadratic program if there is an abrupt change in v. Since constraint (20) is independent of v, an abrupt change in v will not destroy the feasibility of the previously determined $\mathbf{u}_s$ and the working set $\mathcal{W}$ determined previously also remains a valid subset of the active set. Hence, the receding horizon active set method will continue to work even though the objective function (19) has changed. However, if it is necessary to include not only control but also state constraints into the original problem formulation, we shall then require a quadratic programming algorithm (other than the active set method in its simplest form) that does not assume the initial guess to be feasible.

Remark 6. There could be two possible ways to arrange the steps in Algorithm 4. One is to update the working set $\mathcal{W}$ followed by $\mathbf{u}_s$, and the other is to update $\mathbf{u}_s$ followed by the working set $\mathcal{W}$. In the receding horizon framework, it might seem at first glance that the first choice is the right one, since we shall then avoid basing the optimization of $\mathbf{u}_s$ on an "outdated" working set if v happens to have changed. However, it turns out that the first choice is actually undesirable. One well-known "weakness" of the active set method is that it is not so easy to remove a constraint once it enters the working set $\mathcal{W}$. This becomes even more a concern in the receding horizon framework. If v has changed, and so has the objective function (19), the original stationary $\mathbf{u}_s^*$ obtained in step (5) may no longer be stationary, and it will require at least an additional iteration to identify the new stationary point before we can decide whether any constraint can be dropped from the working set or not. This will seriously slow down the transient response of the steady-state subproblem. Indeed, once v has changed, many of the constraints in the previous working set are no longer sensible, and it will be wiser to drop them hastily rather than being too cautious only to find much later that the constraints should still be dropped eventually.

Remark 7. One key element in the active set method of the quadratic program is to add or drop a constraint to or from the working set $\mathcal{W}$. The constraint being added belongs to the blocking set $\mathcal{B}$, defined as those constraints corresponding to the minimum α_0 in (26). Physically, they are the constraints that will be first encountered when we try to move $\mathbf{u}_s(0)$ to $\mathbf{u}_s'(0)$ in (31). The constraint being dropped belongs to the set $\mathcal{L}$, defined as those constraints corresponding to a negative component of the Lagrange multiplier in (29). The active set method will converge in finite time no matter which constraint in $\mathcal{B}$ will be added or which constraint in $\mathcal{L}$ will be dropped. One standard and popular choice in the conventional active set method is that the one in $\mathcal{L}$ corresponding to the most negative component of λ will be dropped, whereas the choice from $\mathcal{B}$ will be arbitrary. This is a very natural choice when there is no other consideration.

However, in the receding horizon framework, one other (and in fact important) consideration emerges, which is the execution time of the control input(s) associated with a constraint. Specifically, if Algorithm 4 takes time N_aT_s to carry out, then $\mathcal{W}_0^*$ updated in the current iteration will be used to optimize $\mathbf{u}_s(N_a)$ in the next iteration,

$$\mathbf{u}_s(N_a) := \begin{bmatrix} u_s(N_a) \\ u_s(N_a+1) \\ \vdots \\ u_s\left(N_a+N_p-1\right) \end{bmatrix}, \tag{33}$$

but the outcome of that optimization is ready only at $k = 2N_a$, based on which the transient control u_t is computed. Suppose that the transient subproblem takes one sampling interval to solve, then the transient subproblem at $k = 2N_a$ will update $u(2N_a+1) = u_s(2N_a+1) + u_t(2N_a+1)$ (see Section 5 below for a more detailed discussion). Hence, the "time priority" of u_s is $2N_a+1, 2N_a+2, \ldots, N_p-1, 0, 1, \ldots, 2N_a$ and from this argument, we choose to drop the constraint in $\mathcal{L}$ that is associated with the first u_s in this sequence or to add the constraint in $\mathcal{B}$ that is associated with the last u_s in this sequence (of course the most negative Lagrange multiplier can still be used as the second criterion

if two constraints in $\mathcal{L}$ happen to have the same urgency). The proposal here aims to assign most freedom to the most urgent control input in the optimization, which makes sense in the receding horizon framework since the less urgent inputs may be reoptimized later.

Remark 8. Basically, the approach proposed in this section is to spread the original quadratic program over many intervals, so that each interval only carries out one iteration of the algorithm, and also to ensure that the quadratic program being solved is consistent when the prediction of the exogenous model is valid, but will migrate to a new quadratic program when there is a change in $v(k)$. It is worth mentioning that the original standard MPC is a static controller by nature, since the true solution of a complete quadratic program is independent of the MPC's decisions in the past (past decisions can help to speed up the computations but will not affect the solution), but by spreading the algorithm over several intervals, it is turned into a dynamic controller with internal state $\mathbf{u}_s(k)$, $\mathcal{W}(k)$.

4. Steady-State Subproblem: A Dynamic Policy Approach

The approach proposed in Section 3 optimizes $\mathbf{u}_s$. Consequently, the number of (scalar) variables being optimized is proportional to N_p. To further cut down the computations required, this section proposes another approach based on the idea of a dynamic policy, inspired by the works of [13, 22, 23]. This approach optimizes a smaller number of variables typically, and the number is independent from N_p, although the number of linear inequality constraints remains the same. In return, the optimization result is expected to be slightly inferior to that of Section 3 due to the reduction of variables (degree of freedom). However, it should be noted that the number of optimized variables in this second approach is adjustable based upon the designer's wish.

The central idea of the dynamic policy approach [13, 22, 23] is that instead of optimizing the control input directly, we generate the control input by a dynamic system of which the initial system state is optimized. This is similar to what we have done to w and r in Section 3. Specifically, we assume u_s is also generated from a state-space model:

$$\hat{v}^+ = A_{\hat{v}}\hat{v}, \tag{34}$$

$$u_s = C_{\hat{v}}\hat{v}, \tag{35}$$

where

$$A_{\hat{v}}^{N_p} = I. \tag{36}$$

This state-space model is designed *a priori* but the initial state $\hat{v}(0)$ will be optimized online. Obviously, the quadratic program (19)-(20) becomes

$$\min_{\hat{v}(0)} \hat{J}_s,$$

$$\hat{J}_s := \begin{bmatrix} \hat{v}(0) \\ v(0) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \hat{\mathbf{H}} & \hat{\mathbf{F}} \\ \hat{\mathbf{F}}^{\mathrm{T}} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \hat{v}(0) \\ v(0) \end{bmatrix}, \tag{37}$$

$$\text{subject to} \quad \hat{\mathbf{a}}_j^{\mathrm{T}}\hat{v}(0) \le \mathbf{b}_j, \quad j = 1, 2, \ldots, N_m, \tag{38}$$

where

$$\hat{\mathbf{H}} = \hat{\mathcal{O}}^{\mathrm{T}}\mathbf{H}\hat{\mathcal{O}},$$

$$\hat{\mathbf{F}} = \mathbf{F}\hat{\mathcal{O}}, \tag{39}$$

$$\hat{\mathbf{a}}_j = \mathbf{a}_j\hat{\mathcal{O}}, \quad j = 1, 2, \ldots, N_m,$$

and $\hat{\mathcal{O}}$ is the observability matrix

$$\hat{\mathcal{O}} := \begin{bmatrix} C_{\hat{v}} \\ C_{\hat{v}}A_{\hat{v}} \\ \vdots \\ C_{\hat{v}}A_{\hat{v}}^{N_p-1} \end{bmatrix}. \tag{40}$$

The number of variables in this new quadratic program is the dimension of $\hat{v}(0)$, denoted by $n_{\hat{v}}$. If $A_{\hat{v}}$ is constructed from the method of Fourier decomposition described in Section 3, Shannon's sampling theorem implies that a sufficiently large but finite $n_{\hat{v}}$ will guarantee a full reconstruction of the original optimization variable $\mathbf{u}_s(0)$. On the other hand, a smaller $n_{\hat{v}}$ restricts the search to a lower dimensional subspace of $\mathbf{u}_s(0)$ and hence the optimization is easier but suboptimal.

One natural choice of the dynamics $A_{\hat{v}}$ is to make $A_{\hat{v}} = A_v$ in the exogenous model (14)–(17). Of course, it should be noted that constrained control is generally a nonlinear problem, and therefore the number of harmonics to be included in u_s may exceed that of w and r in order to achieve the true optimal performance. However, we could have overdesigned the exogenous model (14)–(17) to include more harmonics in A_v than necessary for w and r, thus making the choice $A_{\hat{v}} = A_v$ here not so conservative. The simulation results in Section 6 will demonstrate both cases.

It remains to choose the matrix $C_{\hat{v}}$ in (35). The one we suggest here is based on the linear servomechanism theory [24–26], which solves the linear version of our problem when there is no input constraint. Essentially, when there is no constraint, perfect asymptotic tracking (i.e., $y_s = r$ or $e_s = 0$) can be achieved by solving the regulator equation:

$$\begin{aligned} XA_v &= AX + B_1C_w + B_2U, \\ C_r &= CX + D_1C_w + D_2U, \end{aligned} \tag{41}$$

for the matrices X, U, and then let

$$u_s = Uv, \tag{42}$$

which also implies

$$x_s = Xv. \tag{43}$$

Therefore, to recover the optimal (or perfect) solution in the linear case when u_s does not hit any constraint, the state-space model of u_s may be chosen as

$$\begin{aligned} \tilde{v}^+ &= A_v\tilde{v}, \\ u_s &= U\tilde{v}. \end{aligned} \tag{44}$$

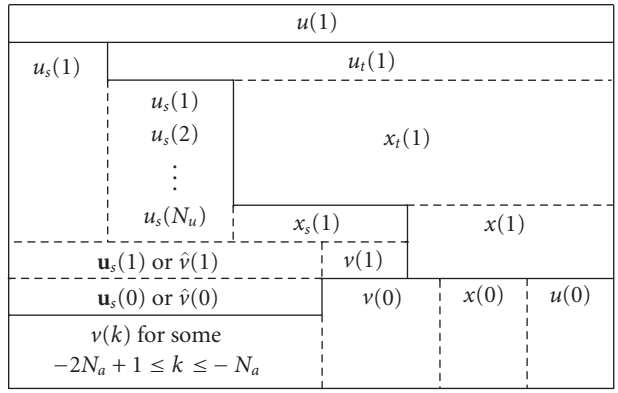

Figure 1: Derivations of unknown variables from known variables.

However, this state-space model is not guaranteed to be observable. When it is not, the resulting $\hat{\mathbf{H}}$ in (37) will become semidefinite instead of strictly positive definite. To overcome this, we suggest to perform an orthogonal state transformation

$$\begin{bmatrix} * \\ \hat{v} \end{bmatrix} = T\tilde{\hat{v}}, \tag{45}$$

to bring (44) to the Kalman decomposition form

$$\begin{aligned} \begin{bmatrix} * \\ \hat{v} \end{bmatrix}^+ &= \begin{bmatrix} * & * \\ 0 & A_{\hat{v}} \end{bmatrix} \begin{bmatrix} * \\ \hat{v} \end{bmatrix}, \\ u_s &= \begin{bmatrix} 0 & C_{\hat{v}} \end{bmatrix} \begin{bmatrix} * \\ \hat{v} \end{bmatrix}, \end{aligned} \tag{46}$$

and hence obtain a reduced-order model to become (34)-(35). It is easy to verify that $A_{\hat{v}}^{N_p} = I$ since

$$\begin{bmatrix} * & * \\ 0 & A_{\hat{v}} \end{bmatrix} \tag{47}$$

is upper block-triangular and

$$\begin{aligned} \begin{bmatrix} * & * \\ 0 & A_{\hat{v}} \end{bmatrix}^{N_p} &= \left(TA_v T^{-1}\right)^{N_p} \\ &= TA_v^{N_p} T^{-1} \\ &= I. \end{aligned} \tag{48}$$

Certainly, the discussion above only provides a suggestion of how to choose the state-space model for u_s, which we shall also adopt for our simulations in Section 6, but, in general, the designer should feel free to employ any valid state-space model to suit his problem.

Remark 9. Having reparameterized the quadratic program in terms of $\hat{v}(0)$ rather than $\mathbf{u}_s(0)$, we can apply a similar version of Algorithm 4 to (37)-(38). In other words, it is not necessary to solve the quadratic program completely within one sampling interval. Instead of rotating the components of $\mathbf{u}_s^*(0)$ to obtain $\mathbf{u}_s(N_a)$, we obtain $\hat{v}(N_a)$ by $A_{\hat{v}}^{N_a}\hat{v}^*(0)$.

5. Implementation Issues and Impacts on Transient Performance

Before we present the simulation results, let us comment on the impact of computational delay on the transient subproblem in Section 2. First of all, we assume that the transient quadratic program can be solved completely within one sampling interval. Therefore, despite the way we presented the cost function J_t in (11), in actual implementation we shall optimize $u_t(1), u_t(2), \ldots, u_t(N_u)$ based on $x_t(1)$ at time $k = 0$, instead of $u_t(0), u_t(1), \ldots, u_t(N_u - 1)$ based on $x_t(0)$. The unknown $x_t(1)$ can be projected from the known variables and system dynamics. After the optimization is carried out to obtain $u_t(1), u_t(2), \ldots, u_t(N_u)$, the control input to be executed at $k = 1$ will be $u(1) = u_s(1) + u_t(1)$. Bear in mind that all these calculations can only be based on the best knowledge of the signals at $k = 0$.

Figure 1 summarizes how the unknown variables can be computed from the known variables. The variables in each layer are derived from those variables directly below them, but in actual implementation it is sometimes possible to derive explicit formulas to compute the upper layer from the lower layer bypassing the intermediate layer, thus not requiring those intermediate calculations online. The variable on top is the control action $u(1)$, computed from the variables of the steady-state subproblem on the left, and those of the transient subproblem on the right, separated by a solid line. The variables in the bottom layer, $v(0)$ is provided by the observer described in Section 3, $x(0)$ is a measurement of the current plant state, and $u(0)$ is the control input calculated by the algorithm at $k = -1$. Note that to compute $u_t(1)$, the values of $u_s(1), u_s(2), \ldots, u_s(N_u)$ are needed to form the constraints for the transient quadratic program since the original linear inequality constraints apply to $u(k) = u_s(k) + u_t(k)$. On the other hand, $x_s(1)$ can be written as a linear function of $v(1)$ and $\mathbf{u}_s(1)$ (or $\hat{v}(1)$) explicitly. Finally, the steady-state subproblem requires a computational time of $N_a T_s$, implying that the solution $\mathbf{u}_s(0)$

provided by the steady-state subproblem at $k = 0$ is based on a measurement of $v(k)$ at some k between $-2N_a+1$ and $-N_a$. So in the worst case, $u(1)$ is based on some information as old as $v(-2N_a + 1)$, which corresponds to a worst-case delay of $2N_aT_s$. For instance, if $N_a = 1$, the control $u(k)$ is computed from a measurement of $x(k-1)$, $v(k-1)$, and $v(k-2)$.

Remark 10. Although we said in Section 2 that the transient subproblem was not the main interest of this paper, it is an ideal vehicle to demonstrate the power of MPC since the "useful freedom" of $u_t(k)$ may have been totally consumed by $u_s(t)$ when the latter hits a constraint. For example, the simulations to be discussed in Section 6 have the constraint

$$|u(k)| = |u_s(k) + u_t(k)| \le 1. \tag{49}$$

If $u_s(k)$ already saturates at ± 1, one side of $u_t(k)$ is lost but that could be the only side to make the reduction of J_t possible, thus forcing $u_t(k)$ to zero. So the input constraint does not only restrict the magnitude, but also the time instant to apply the transient control u_t. Such problem is extremely difficult for conventional control techniques.

6. Simulation Results

In this section we use an example to demonstrate the performance of our algorithms by computer simulations. The plant is borrowed from [19] and represents a power quality device called Unified Power Quality Conditioner (UPQC), which has the following continuous-time state-space model:

$$\begin{aligned}
\dot{x} &= Ax + B_1 w + B_2 u,\\
y &= Cx + D_1 w + D_2 u,\\
A &= \begin{bmatrix}
\frac{-R_l}{L_l} & 0 & 0 & \frac{-1}{L_l} & \frac{-1}{L_l}\\
0 & \frac{-R_{se}}{L_{se}} & 0 & \frac{-1}{L_{se}} & 0\\
0 & 0 & \frac{-R_{sh}}{L_{sh}} & 0 & \frac{-1}{L_{sh}}\\
\frac{1}{C_{se}} & \frac{1}{C_{se}} & 0 & 0 & 0\\
\frac{1}{C_{sh}} & 0 & \frac{1}{C_{sh}} & 0 & 0
\end{bmatrix},\\
B_1 &= \begin{bmatrix}
\frac{1}{L_l} & 0\\
0 & 0\\
0 & 0\\
0 & 0\\
0 & \frac{-1}{C_{sh}}
\end{bmatrix}, \qquad
B_2 = \begin{bmatrix}
0 & 0\\
\frac{V_{dc}}{2L_{se}} & 0\\
0 & \frac{V_{dc}}{2L_{sh}}\\
0 & 0\\
0 & 0
\end{bmatrix},\\
C &= \begin{bmatrix}
0 & 0 & 0 & 0 & 1\\
1 & 0 & 0 & 0 & 0
\end{bmatrix},\\
D_1 &= \begin{bmatrix}
0 & 0\\
0 & 0
\end{bmatrix}, \qquad
D_2 = \begin{bmatrix}
0 & 0\\
0 & 0
\end{bmatrix}.
\end{aligned} \tag{50}$$

The exogenous input w is composed of the supply voltage and the load current, which are periodic at 50 Hz but may consist of higher-order harmonics. The plant output y is composed of the load voltage and the supply current, which will be made to track designated pure sine waves of 50 Hz. The control input u is composed of two switching signals across the voltage source inverters (VSIs), both of which are required to satisfy the bounds $-1 \le u \le 1$. The general control objective is to maintain y to the desired waveforms despite possible fluctuations in w like supply voltage sags/swells or load demand changes. To apply the MPC algorithms proposed, we obtain a discrete-time version of the above state-space model by applying a sampling interval of $T_s = 0.2$ ms (i.e., 100 samples per period). Small-sized quadratic programs (such as our transient subproblem) can possibly be solved within such a short time thanks to the state-of-the-art code optimization [12], which reports sampling rates in the range of kHz, but to solve a big quadratic program like our steady-state subproblem we shall resort to the technique of Algorithm 4. Note that in our formulation, the transient subproblem and the steady-state subproblem can be solved in parallel. Although the optimization of u_t depends on u_s, the transient control $u_t(k+1)$ is computed from $\mathbf{u}_s(k)$ which is made available by the steady-state subproblem in the previous step. So it is independent of the current steady-state subproblem being solved.

As typical in a power system, we assume only odd harmonics in the supply voltage and the load current. Hence we can reduce the computations in the steady-state subproblem by the following easy modifications from the standard algorithms presented in Sections 3 and 4; N_p may be chosen to represent *half* of the period instead of the whole period, satisfying

$$x_s(0) = -x_s\left(N_p\right), \tag{51}$$

instead of (6), and

$$\begin{aligned}
A_v^{N_p} &= -I,\\
A_{\hat{v}}^{N_p} &= -I,
\end{aligned} \tag{52}$$

instead of (17) and (36), with $\omega T_s \times N_p = \pi$ instead of 2π. The rotation operation in (32) should also become

$$\begin{bmatrix}
u_s(N_a)\\
u_s(N_a+1)\\
\vdots\\
u_s\left(N_p-1\right)\\
u_s\left(N_p\right)\\
u_s\left(N_p+1\right)\\
\vdots\\
u_s\left(N_a+N_p-1\right)
\end{bmatrix} = \begin{bmatrix}
u_s^*(N_a)\\
u_s^*(N_a+1)\\
\vdots\\
u_s^*\left(N_p-1\right)\\
-u_s^*(0)\\
-u_s^*(1)\\
\vdots\\
-u_s^*(N_a-1)
\end{bmatrix}. \tag{53}$$

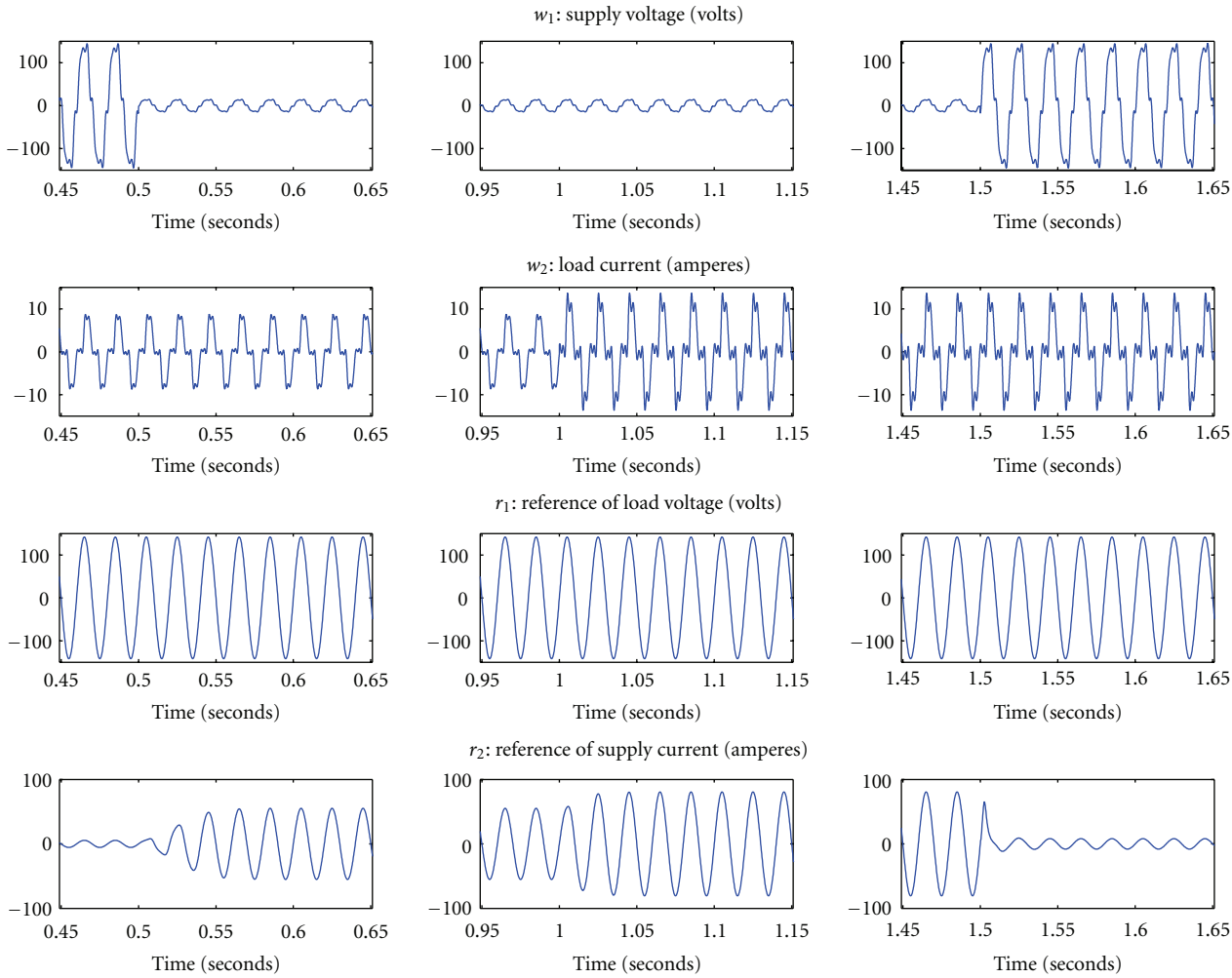

Figure 2: Simulation scenario. Voltage sag at $t = 0.5$ s; load demand changed at $t = 1.0$ s; sag cleared at $t = 1.5$ s.

Table 1: Values of the components of the UPQC.

Component	V_{dc}	L_{se}	L_{sh}	C_{se}	C_{sh}
Value	320 V	5.0 mH	1.2 mH	10 μF	20 μF

Table 2: Line impedance and VSI impedances of the UPQC.

Component	R_l	L_l	R_{se}	R_{sh}
Value	0.01 Ω	1.0 mH	0.01 Ω	0.01 Ω

This cuts down half of the scalar variables as well as constraints in the quadratic program.

The model parameters of the UPQC used in our simulations are summarized in Table 1 for the circuit components and Table 2 for the line and VSI impedances. They are the same as those values in [19], except for V_{dc} which we have changed from 400 V to 320 V so as to produce a saturated control u more easily. Note that V_{dc} is the DC-link voltage, which determines how big a fluctuation in the supply voltage or load current the UPQC can handle. In other words, saturation occurs when the UPQC is trying to deal with an unexpected voltage sag/swell or load demand that is beyond its designed capability.

The simulation scenario is summarized in Figure 2. Both the supply voltage and the load current consist of odd harmonics up to the 9th order. Despite the harmonics, it is desirable to regulate the load voltage to a *fixed* pure sine wave, whereas the supply current should also be purely sinusoidal, but its magnitude and phase are selected to maintain a power factor of unity and to match the supply active power to the active power demanded by the load, which means the reference of this supply current is w-dependent. The waveforms of both w and r are shown in Figure 2. The simulation scenario is designed such that the steady-state control u_s is not saturated at the beginning. At $t = 0.5$ s, a voltage sag occurs which reduces the supply voltage to 10% of its original value. The UPQC is expected to keep the load voltage unchanged but (gradually) increase the supply current so as to retain the original active power. This will drive u_s into a *slightly* saturated situation. At $t = 1.0$ s, the load demand increases, causing the reference of the supply current to increase again, and u_s will become *deeply* saturated. At $t = 1.5$ s, the voltage sag is cleared and the

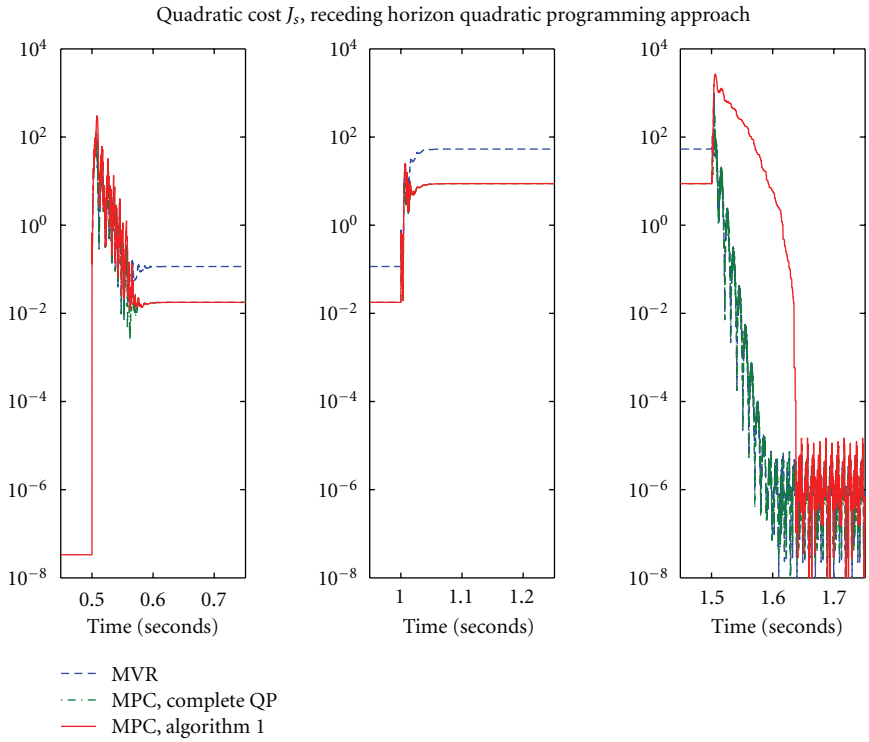

FIGURE 3: Quadratic cost J_s of the steady-state subproblem, the receding horizon quadratic programming approach.

supply voltage returns to its starting value, but the (new) load demand remains. Although the load demand is still higher than its initial value, the u_s required will just be within the bounds of ± 1, thus leaving the saturation region to return to the linear region. So, in short, u_s is expected to experience "*linear → slightly saturated → deeply saturated → linear but nearly saturated*" in this simulation scenario.

To evaluate the performance of our algorithms, we compare them to two other cases. In the first case, instead of Algorithm 4, the complete quadratic program is solved in each iteration of the steady-state subproblem every N_a-sampling intervals. We call this case the complete QP, and it serves to indicate how much transient performance has been sacrificed (in theory) by spreading the quadratic program over a number of iterations. In the second case, the constraints are totally ignored, such that the optimal $\mathbf{u}_s(0)$ in the steady-state subproblem is supposed to be

$$\mathbf{u}_s(0) = \begin{bmatrix} U \\ UA_v \\ \vdots \\ UA_v^{N_p-1} \end{bmatrix} v(0), \tag{54}$$

where U is the solution to the regulator equation (41). The transient subproblem can also be solved by

$$u_t = Fx_t, \tag{55}$$

where F is the optimal state-feedback gain minimizing the transient quadratic cost

$$\sum_{k=0}^{\infty} \left(y_t(k)^{\mathrm{T}} Q y_t(k) + u_t(k)^{\mathrm{T}} R u_t(k) \right). \tag{56}$$

However, the combined input u is still clipped at ± 1. We label this control law the multivariable regulator (MVR) following the linear servomechanism theory. This case serves to indicate how bad the quadratic cost J_s can be if a linear control law is used without taking the constraints into consideration.

Note that in both cases, the computational delays discussed in Section 5 will be in force, where $u(k+1)$ instead of $u(k)$ will be optimized and the steady-state control $\mathbf{u}_s$ is only updated every N_a-sampling intervals. In reality, of course the MVR should involve negligible computational delay, whereas the complete QP should need a longer time to solve than Algorithm 4, but we are merely using their associated quadratic costs here to analyze the behaviours of our algorithms.

Figure 3 plots the steady-state cost J_s of our first approach in Section 3 based on receding horizon quadratic programming together with the costs in the other two cases. N_a is assumed to be 3 in this simulation. The transient subproblem has a control horizon of $N_u = 5$, corresponding to 10 scalar variables and 20 scalar inequality constraints. On the other hand, the steady-state subproblem has $N_p = 50$ corresponding to 100 variables and 200 constraints. As shown in Figure 3, all J_s are zero prior to $t = 0.5$ s and should also settle down to zero after $t = 1.5$ s. It is observed

Table 3: Summary of J_s values for various cases studied.

	Steady-state cost J_s		N_a	Number of	
	at $t < 1.0$ s	at $t < 1.5$ s		Variables	Constraints
MVR	0.1156	53.276			
MPC, receding horizon quadratic programming	0.0178	8.7527	3	100 ($N_p \times 2$)	200 ($N_p \times 2 \times 2$)
MPC, dynamic policy (up to 9th harmonics)	0.0225	9.0273	1	$n_{\hat{v}} = 20$ ($5 \times 2 \times 2$)	200 ($N_p \times 2 \times 2$)
MPC, dynamic policy (up to 29th harmonics)	0.0182	8.7640	2	$n_{\hat{v}} = 60$ ($15 \times 2 \times 2$)	200 ($N_p \times 2 \times 2$)
Transient subproblem				10 ($N_u \times 2$)	20 ($N_u \times 2 \times 2$)

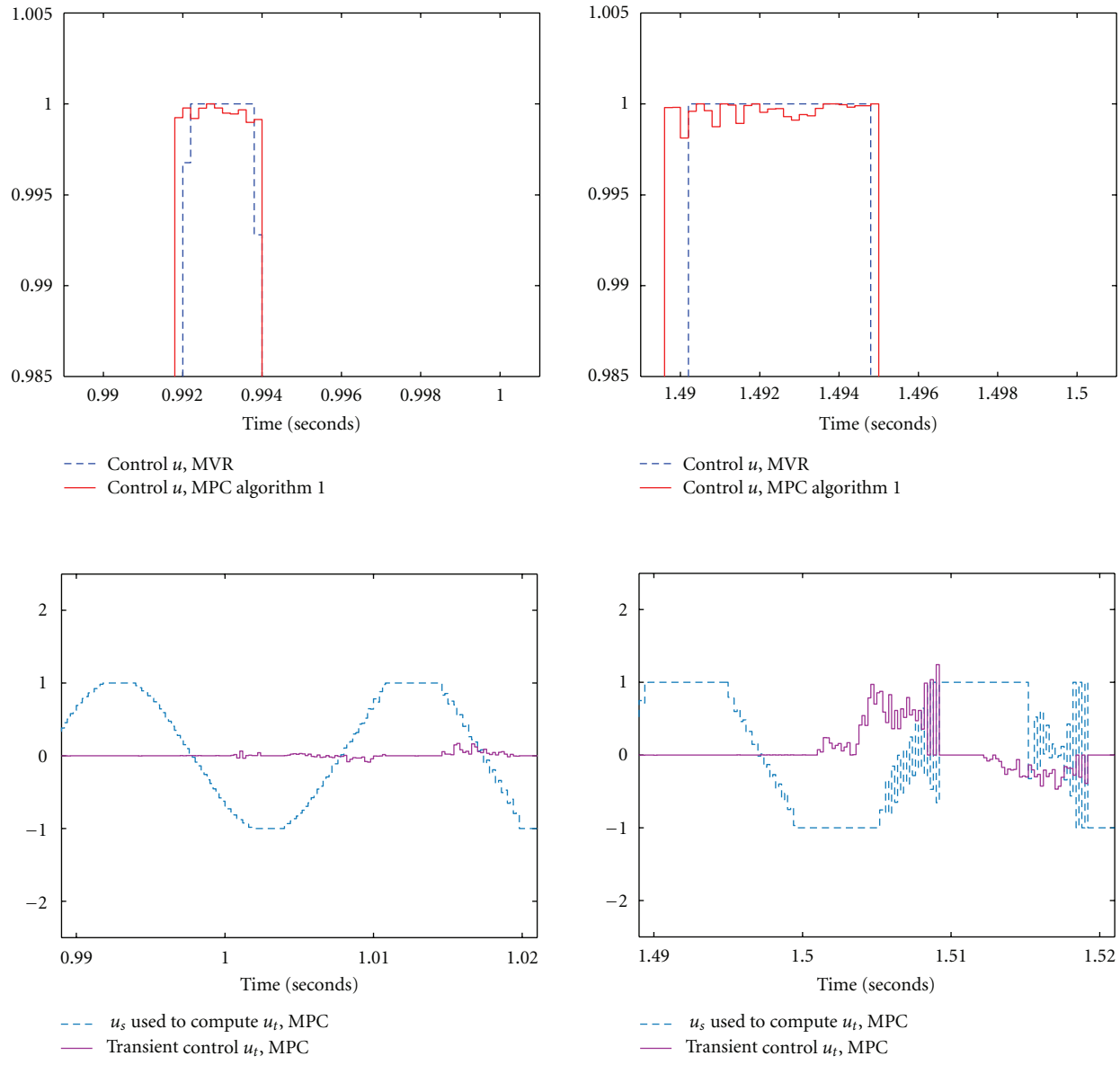

Figure 4: The first component of controls u and u_t before and after transitions at $t = 1.0$ s and $t = 1.5$ s.

that the transient response of our J_s is pretty close to that of the complete QP during the transitions from "linear" to "slightly saturated" and from "slightly saturated" to "deeply saturated," but is poorer when it tries to return from "deeply saturated" to "linear." This can probably be attributed to the weakness of the active set method in removing a constraint from the working set as discussed in Remark 6. Figure 3 also indicates that the MVR settles down to a much higher J_s value when a saturation occurs, due to its ignorance of the control constraints. The exact values of J_s just prior to $t = 1.0$ s and $t = 1.5$ s are summarized in Table 3, which are about 6-7 times the J_s values of our algorithm.

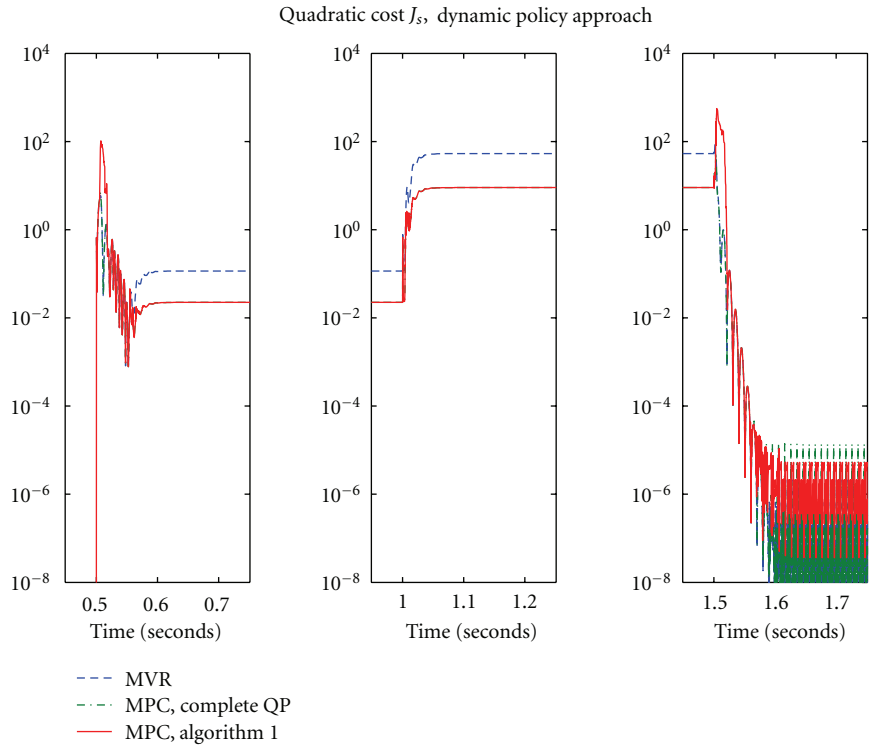

FIGURE 5: Quadratic cost J_s of the steady-state subproblem, the dynamic policy approach.

Figure 4 zooms into the first component of the control input u. The top two plots draw attentions to the steady-state control u_s (u is essentially u_s just prior to $t = 1.0$ s and $t = 1.5$ s). It is observed that the differences in u_s between MVR and MPC are very subtle. Compared with the MVR, the MPC u_s merely reaches and leaves its limit (+1 here) at very slightly different time instants and also produces some "optimized ripples" of less than 0.25% around that limit instead of a "flat" value as adopted in the clipped linear control law, but by doing these little things the MPC manages to bring J_s down by almost one order of magnitude. This demonstrates how nontrivial the optimal u_s can be. We can also see from the plots that only one constraint is active in the "slightly saturated" situation whereas multiple constraints are active in the "deeply saturated" saturation. On the other hand, the bottom two plots in Figure 4 illustrates our discussion in Remark 10. The plots clearly show that during certain moments of the transient stages ($t > 1.0$ s and $t > 1.5$ s), the transient control u_t is "disabled" due to the saturation of the steady-state control u_s. Note that we are labeling the dashed blue curve as "u_s used to compute u_t" since it is slightly different from the actual u_s. For instance, $u_t(k+1)$ is computed from the knowledge of u_s at time k, which is not exactly the same as $u_s(k+1)$. Obviously, u_t is not just disabled whenever u_s saturates. It happens only when the desired direction of u_t violates the active constraint.

Next, let us look at the performance of our second MPC approach in Section 4 based on dynamic policy. Odd harmonics up to the 9th order are included in $A_{\hat{v}}$ resulting in a total of $n_{\hat{v}} = 20$ variables. See Table 3. Since the number of variables is much lower than that of the first approach, we assume $N_a = 1$ here, that is, *one iteration* of Algorithm 4 (equivalent version) is carried out in each sampling interval, whereas the transient subproblem is solved *completely* within each sampling interval. The transient performance of J_s is plotted in Figure 5. Note that the MVR curve exhibits a slightly different transient from Figure 3 since their N_a values are different. The dynamic policy approach clearly shows a faster transient response than the receding horizon quadratic programming approach, not only because of a smaller N_a but also a smaller-sized quadratic program overall. However, the drawback is a slightly suboptimal J_s, as indicated in Table 3.

As mentioned in Section 4, it is possible to over-design A_v, and hence $A_{\hat{v}}$, so that the optimal J_s in this second MPC method will approach the first MPC method. For example, although we only have odd harmonics up to the 9th order in w, we may include odd harmonics up to the 29th order in A_v and $A_{\hat{v}}$. The results are also recorded in Table 3, and we see that this J_s value is very close to the optimal one in the first method.

7. Conclusions

To apply MPC to fast-sampling systems with input constraints for the tracking of periodic references, efficient algorithms to reduce online computational burdens are necessary. We have decomposed the tracking problem into a computationally complex steady-state subproblem and a computationally simple transient subproblem, and then proposed two approaches to solve the former. The first approach,

based on the concept of receding horizon quadratic programming, spreads the optimization over several sampling intervals, thus reducing the computational burdens at the price of a slower transient response. The second approach, based on the idea of a dynamic policy on the control input, further reduces the online computations at the price of a slightly suboptimal asymptotic performance. Despite the limitations, these approaches make the application of MPC to fast-sampling systems possible. Their transient behaviours and steady-state optimality have been analyzed via computer simulations, which have also demonstrated that the steady-state subproblem and the transient subproblem can be solved in parallel with different sampling rates. When the methods proposed in this paper are combined with modern code optimizations, the applicability of MPC to the servomechanism of fast-sampling constrained systems will be greatly enhanced.

Acknowledgment

This work is supported by HKU CRCG 201008159001.

References

[1] J. M. Maciejowski, *Predictive Control with Constraints*, Prentice-Hall, New Jersey, NY, USA, 2002.

[2] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[3] P. Whittle, *Optimization Over Time*, Wiley, New York, NY, USA, 1982.

[4] W. H. Kwon and S. Han, *Receding Horizon Control*, Springer, New York, NY, USA, 2005.

[5] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[6] A. Bemporad, F. Borrelli, and M. Morari, "Min-max control of constrained uncertain discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 9, pp. 1600–1606, 2003.

[7] A. Bemporad and C. Filippi, "Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming," *Journal of Optimization Theory and Applications*, vol. 117, no. 1, pp. 9–38, 2003.

[8] Z. Wan and M. V. Kothare, "Robust output feedback model predictive control using off-line linear matrix inequalities," *Journal of Process Control*, vol. 12, no. 7, pp. 763–774, 2002.

[9] Z. Wan and M. V. Kothare, "An efficient off-line formulation of robust model predictive control using linear matrix inequalities," *Automatica*, vol. 39, no. 5, pp. 837–846, 2003.

[10] M. Åkerblad and A. Hansson, "Efficient solution of second order cone program for model predictive control," *International Journal of Control*, vol. 77, no. 1, pp. 55–77, 2004.

[11] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of Optimization Theory and Applications*, vol. 99, no. 3, pp. 723–757, 1998.

[12] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control: automatic generation of high-speed solvers," *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 52–65, 2011.

[13] M. Cannon and B. Kouvaritakis, "Optimizing prediction dynamics for robust MPC," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1892–1897, 2005.

[14] A. Richards, K.-V. Ling, and J. Maciejowski, "Robust multiplexed model predictive control," in *Proceedings of the European Control Conference*, Kos, Greece, July 2007.

[15] K. V. Ling, W. K. Ho, Y. Feng, and B. Wu, "Integral-square-error performance of multiplexed model predictive control," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 196–203, 2011.

[16] D. Li and Y. Xi, "Aggregation based robust model predictive controller for systems with bounded disturbance," in *Proceedings of the 7th World Congress on Intelligent Control and Automation, (WCICA '08)*, pp. 3374–3379, Chongqing, China, June 2008.

[17] D. Li and Y. Xi, "Aggregation based closed-loop MPC with guaranteed performance," in *Proceedings of the 48th IEEE Conference on Decision and Control, (CDC/CCC '09)*, pp. 7400–7405, Shanghai, China, December 2009.

[18] J. B. Rawlings and K. R. Muske, "The stability of constrained receding horizon control," *IEEE Transactions on Automatic Control*, vol. 38, no. 10, pp. 1512–1516, 1993.

[19] K. H. Kwan, Y. C. Chu, and P. L. So, "Model-based H∞ control of a unified power quality conditioner," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 7, pp. 2493–2504, 2009.

[20] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, New York, NY, USA, 1996.

[21] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, NY, USA, 2nd edition, 2006.

[22] A. Gautam, Y.-C. Chu, and Y. C. Soh, "Robust MPC with optimized controller dynamics for linear polytopic systems with bounded additive disturbances," in *Proceedings of the 7th Asian Control Conference, (ASCC '09)*, pp. 1322–1327, Hong Kong, China, August 2009.

[23] A. Gautam, Y.-C. Chu, and Y. C. Soh, "Optimized dynamic policy for receding horizon control of linear time-varying systems with bounded disturbances," *IEEE Transactions on Automatic Control*. In press.

[24] E. J. Davison, "The robust control of a servomechanism problem for linear time-invariant multivariable systems," *IEEE Transactions on Automatic Control*, vol. 21, no. 1, pp. 25–34, 1976.

[25] B. A. Francis and W. M. Wonham, "The internal model principle of control theory," *Automatica*, vol. 12, no. 5, pp. 457–465, 1976.

[26] B. A. Francis, "The linear multivariable regulator problem," *SIAM Journal on Control Optimization*, vol. 15, no. 3, pp. 486–505, 1977.