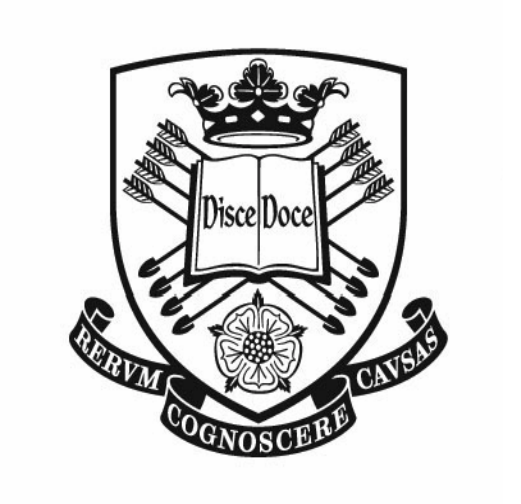
Iterative Learning Control for Constrained Linear Systems

Bing Chu, David H Owens

Department of Automatic Control and Systems Engineering,
University of Sheffield,
Mappin Street, Sheffield, S1 3JD, UK



Research Report No. 990

Department of Automatic Control and Systems Engineering
The University of Sheffield
Mappin Street, Sheffield,
S1 3JD, UK

April 2009

Iterative Learning Control for Constrained Linear Systems

Bing Chu, David H Owens [a]

[a] *Department of Automatic Control and Systems Engineering*
The University of Sheffield, Mappin Street, Sheffield S1 3JD, UK
Bing.Chu@shef.ac.uk, D.H.Owens@shef.ac.uk

Abstract

This paper considers iterative learning control for linear systems with convex control input constraints. First, the constrained ILC problem is formulated in a novel successive projection framework. Then, based on this projection method, two algorithms are proposed to solve this constrained ILC problem. The results show that, when perfect tracking is possible, both algorithms can achieve perfect tracking. The two algorithms differ however in that one algorithm needs much less computation than the other. When perfect tracking is not possible, both algorithms can exhibit a form of practical convergence to a "best approximation". The effect of weighting matrices on the performance of the algorithms is also discussed and finally, numerical simulations are given to demonstrate the effectiveness of the proposed methods.

Key words: iterative learning control; projection method; constraint handling; norm optimization

1 Introduction

Iterative learning control (ILC) is a control method for improving tracking performance of systems that execute the same task repeatedly by learning from the past actions. Applications of ILC can be widely found in industrial robot manipulator, chemical batch process, some medical equipment and manufacturing, etc. Originating from robotics, ILC now attracts more general research interest [1], [2].

In many practical applications, the systems are under some constraints due to physical limitations or performance requirements. Hence, the ILC design must take these constraints into account. However, most of the current ILC research is based on assumed unconstrained systems and few results have been reported regarding the constrained case in the literature. [3] proposes a novel nonlinear controller for process systems with input constraints and the learning scheme only needs a little knowledge of the process model. [4] considers ILC problem with soft constraints and uses Lagrange multiplier methods to solve this problem. [5] uses quadratic optimal design to formulate the constrained ILC problem and suggests quadratic optimal design has the capability of dealing with constraints.

In this paper, ILC design problem with general convex input constraints is discussed. It is shown that the constrained ILC problem can be formulated in a recently developed successive projection framework of ILC [6], which provides an intuitive but rigorous method for system analysis and design. Based on this, a systematic approach for constraints handling is provided and two algorithms are proposed to solve this problem. The convergence analysis shows that when perfect tracking is possible, both algorithms can achieve perfect tracking whereas the computation of one algorithm is much less than the other at the cost of slightly slower convergence rate. When perfect tracking is not possible, both algorithms converge to asymptotic values representing a "best fit" solution. Again the computational complex algorithm has the best convergence properties. It is also found that the input and output weighting matrices have an interesting effect on the convergence properties of the algorithms.

The paper is organized as follows. In Section 2, the constrained ILC problem is formulated. In Section 3, the successive projection method is introduced and the constrained ILC problem is interpreted using this successive projection formulation. In Section 4 and Section 5, two algorithms are proposed and their convergence properties derived. In Section 6, numerical simulations are presented to demonstrate the effectiveness of the proposed methods and finally, conclusions are given in Section 7.

2 Problem Formulation

For simplicity, the formulation is described for linear discrete time systems but more generally applies to linear systems in Hilbert spaces described by equations of the form $y = Gu + d$ where u, y are the system input and output respectively, G is a bounded linear operator from an input Hilbert space to an output Hilbert space and d represents other effects including the effect of initial state conditions. For more details see [7]. Note that the abstract formulation describes many situations of interest including continuous linear state space model, discrete time model and differential delay model of system dynamics.

Consider the following discrete time, linear time-invariant system

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t) \\ y_k(t) &= Cx_k(t), \end{aligned} \tag{1}$$

where t is the time index (i.e. sample number), k is the iteration number and $u_k(t), x_k(t), y_k(t)$ are input, state and output of the system on iteration k. The initial condition $x_k(0) = x_0, k = 1, 2, \cdots$ is the same for all iterations. The control objective is to track a given reference signal $r(t)$ defined on a finite duration $t \in [0, N]$ (i.e. t is the sample number for time series of length $N+1$)and to do so by repeated execution of the task and data transfer from task to task. Mathematically, at the final time $t = N$, the state is reset to x_0 and time is reset to $t = 0$, a new iteration is started and, again, the system is required to track the same reference.

Before presenting the main results, the operator form of the dynamics is demonstrated using the well-known, so-called lifted-system representation, which provides a straightforward "$N \times N$ matrix" approach in the analysis of discrete-time ILC [8], [9].

Assume, for simplicity, the relative degree of the system is unity (i.e. the generic condition $CB \neq 0$ is satisfied), then system model (1) on the k^{th} iteration can be expressed in an equivalent form

$$y_k = Gu_k + d, \tag{2}$$

where G and d are the $N \times N$ and $N \times 1$ matrices

$$G = \begin{bmatrix} CB & 0 & \cdots & 0 & 0 \\ CAB & CB & \ddots & 0 & 0 \\ CA^2B & CAB & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & CB & 0 \\ CA^{N-1}B & \cdots & \cdots & CAB & CB \end{bmatrix}$$

$$d = \begin{bmatrix} CAx_0 & CA^2x_0 & CA^3x_0 & \cdots & CA^Nx_0 \end{bmatrix}^T. \tag{3}$$

The $N \times 1$ vectors of input, output and reference time series u_k, y_k, r are defined as

$$\begin{aligned} u_k &= \begin{bmatrix} u_k(0) & u_k(1) & \cdots & u_k(N-1) \end{bmatrix}^T \\ y_k &= \begin{bmatrix} y_k(1) & y_k(2) & \cdots & y_k(N) \end{bmatrix}^T \\ r &= \begin{bmatrix} r(1) & r(2) & \cdots & r(N) \end{bmatrix}^T \end{aligned} \tag{4}$$

and k represents the iteration number. As the most important signal vector is the tracking error vector $e = r - y$, then, without loss of generality, it can be assumed that $d = 0$ by incorporating it into the reference signal (i.e. replacing r by $r - d$). Hence (2) becomes

$$y_k = Gu_k, \tag{5}$$

where G is nonsingular and hence invertible.

The above representation of the original system (1) is called the lifted-system representation. This approach changes the original ILC problem into a MIMO tracking problem [8], [9]. Note that the above lifted-system form can be easily extended to situation when the system relative degree is larger than one. All the following discussions will be based on the lifted-system representation.

Tracking error improvements from iteration to iteration are achieved in ILC using the following general control updating law

$$u_{k+1} = f\left(e_{k+1}, \ldots, e_{k-s}, u_k, \cdots, u_{k-r}\right), \tag{6}$$

where e_k is the tracking error from the k^{th} trial/iteration and is defined as $e_k = r - y_k$. When $s > 0$ or $r > 0$, (6) is called a high order updating law. This paper only considers algorithms of the form $u_{k+1} = f\left(e_{k+1}, e_k, u_k\right)$. For higher order algorithms, please refer to [10], [11] and the references therein.

The ILC Algorithm Design Problem: The ILC algorithm design problem can now be stated as finding a control updating law (6) such that the system output has the asymptotic property that $e_k \to 0$ as $k \to \infty$.

There are many design methods to solve the ILC problem. The one used here is based on a quadratic (norm) optimal formulation [12] where, at each iteration, a performance index is minimized to obtain the system input time series vector to be used for that iteration. The basis of this paper is Norm-Optimal ILC (NOILC) which uses the following performance index

$$J_{k+1}(u_{k+1}) = \|e_{k+1}\|_Q^2 + \|u_{k+1} - u_k\|_R^2, \tag{7}$$

minimized subject to the constraint that $e_{k+1} = r - Gu_{k+1}$, G is the operator form of the system (1) and

Q and R are positive definite weighting matrices. Also $\|e\|_Q^2$ denotes the quadratic form e^TQe and similarly with $\|\cdot\|_R^2$. Solving this optimization problem gives the following optimal choice for the time series vector u_{k+1}

$$u_{k+1} = u_k + R^{-1}G^TQe_{k+1} \tag{8}$$

which, when $k \to \infty$, asymptotically achieves perfect tracking. This well-known NOILC algorithm has many appealing properties including implementation in terms of Riccati state feedback. More details on NOILC can be found in [7], [12], [13], [14], [15].

In practical applications, system constraints are widely encountered. There are different kinds of constraints, e.g., input constraint, input rate constraint and state or output constraint. Constraints can be divided into two classes: hard constraints and soft constraints. Hard constraints are constraints on magnitude(s) at each point in time, for example, the output limits on actuators. Soft constraints are constraints that are applied to the whole function rather than its point-wise values e.g. constraints on total energy usage. The input constraints are often hard constraints. This paper only considers the input constraint. Suppose the input is constrained to be in a set Ω, which is taken to be a closed convex set in some Hilbert space H. In practice, the set Ω is often simple one. For example, the following constraints are often encountered:

- $\Omega = \{u \in H : |u(t)| \leq M(t)\}$
- $\Omega = \{u \in H : \lambda(t) \leq |u(t)| \leq \mu(t)\}$
- $\Omega = \{u \in H : 0 \leq u(t)\}$

If there are no constraints, the ILC design problem is relatively easy to solve and there are many design methods in the literature. However, when constraints are present, the problem becomes more complicated. The problem is to decide how to incorporate the constraints into the design process while retaining known performance properties. In the following sections, the successive projection method proposed by Owens and Jones in [16] is used to interpret iterative learning control, and a systematic approach for constraints handing in ILC is then proposed in the form of two new algorithms. The algorithms are related to but distinct from recently published work [6] where successive projection was used to accelerate norm optimal ILC.

3 Interpretation of ILC Using Successive Projection

In this section, the concept of successive projection is summarised and its use in the ILC problem is demonstrated (for more details, please refer to [6] which uses the concepts to successfully accelerate norm optimal ILC). It is shown that the convex constrained ILC problem can be formulated in the successive projection framework, the consequence of which is that a systematic approach for constraints handling is produced with known convergence properties. The notation in [16] is adopted in order to be consistent with the original paper and make the proof of our results more understandable. The notation r, k, t is also used elsewhere in the paper to denote other variables, parameters or signals. This should cause no confusion as their meaning can be inferred from the context.

3.1 Successive Projection Method: An Overview

The successive projection method in the form described by Owens and Jones [16] is a technique for finding a point in the (assumed non-empty) intersection $K_1 \cap K_2$ of two closed, convex sets K_1 and K_2 in some real Hilbert space H. The basic idea is to first select an initial iterate k_0 in H. Subsequent points are obtained successively by projection of previous iterates onto one and then the other of the two convex sets. It is formally described in the following theorem.

Theorem 1 *[16] Let $K_1 \subset H, K_2 \subset H$, be two closed convex sets in a real Hilbert space H with $K_1 \cap K_2$ non-empty. Define*

$$K_j = \begin{cases} K_1, \ j \ odd \\ K_2, \ j \ even \end{cases}$$

Then, given the initial guess $k_0 \in H$, the sequence $\{k_j\}_{j\geq 0}$ satisfying

$$\|k_j - k_{j-1}\| = \min_{k\in K_j} \|k - k_{j-1}\|, \quad j \geq 1 \tag{9}$$

with $k_j \in K_j, j \geq 1$, is uniquely defined for each $k_0 \in H$ and satisfies

$$\|k_{j+1} - k_j\| \leq \|k_j - k_{j-1}\|, \quad j \geq 2. \tag{10}$$

Furthermore, for any $x \in K_1 \cap K_2$,

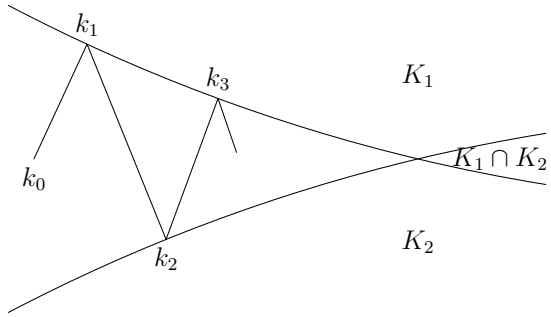
$$\|x - k_j\|^2 \geq \|x - k_{j+1}\|^2 + \|k_{j+1} - k_j\|^2 \tag{11}$$

so that the sequence $\{\|x - k_{j+1}\|\}_{j\geq 0}$ is monotonically decreasing and $\{k_j\}_{j\geq 0}$ continuously gets closer to every point in $K_1 \cap K_2$. In addition

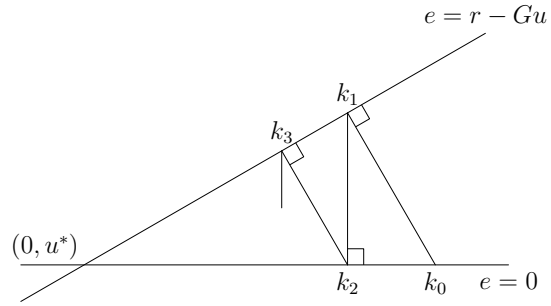
$$\sum_{j=1}^{\infty} \|k_{j+1} - k_j\|^2 \leq \|x - k_1\|^2 \tag{12}$$

so that, for each $\epsilon > 0$, there exists an integer N such that for $j \geq N$

$$\inf_{k\in K_{j+1}} \|k - k_j\| < \epsilon. \tag{13}$$



(a) Illustration of the Successive Projection Algorithm



(b) Geometric Illustration of NOILC

Fig. 1. Successive Projection Interpretation of NOILC

That is, the iterates $k_j \in K_j$ become arbitrarily close to K_{j+1}.

Moreover, when $K_1 \cap K_2$ is empty, the algorithm converges in the sense that $\|k_{j+1} - k_j\| \to d(K_1, K_2)$ defining the minimum distance $d(K_1, K_2)$ between the two sets K_1 and K_2.

The process is illustrated in Figure 1(a) which indicates convergence schematically to a point in the intersection $K_1 \cap K_2$. In [16], this convergence is proved and a number of related and improved iterative schemes are presented. Here, the one related to our ILC results is used. For more details please see [16].

3.2 Interpretation of ILC with Input Constraints

Consider the ILC design problem initially without constraints. If the original system is injective, then for every achievable $r(t)$, there exists a unique input $u^*(t)$ such that $r(t) = [Gu^*](t)$. The task of the ILC control law is to iteratively find a series of inputs such that $u_k \to u^*$ as k tends to infinity. That is equivalent to iteratively finding the unique point $(0, u^*) \in H = \mathbb{R}^N \times \mathbb{R}^N$ in the intersection of the following two sets in H:

- $S_1 = \{(e, u) \in H : e = r - Gu\}$
- $S_2 = \{(e, u) \in H : e = 0\}$

The successive projection method then can be applied to generate an algorithm with the defined convergence properties. In general it is required to verify that these two sets are closed and convex in H. This is trivially satisfied, for example, in finite dimensional time series spaces such as $H = \mathbb{R}^N \times \mathbb{R}^N$. In this case,the inner product will be taken to be

$$\langle (e, u), (z, v) \rangle = e^T Q z + u^T R v \tag{14}$$

with $Q > 0, R > 0$ symmetric positive definite and the associated induced norm will be $||(e, u)|| = \sqrt{\langle (e, u), (e, u) \rangle}$.

Then, using successive projection method in Theorem 1, the well-known NOILC algorithm can be easily derived [6], which is illustrated geometrically in Figure 1(b). Its convergence properties can also be easily derived. For more details, please refer to [6].

Now consider the constrained ILC problem discussed in Section 2. The problem is to find the intersection of the following closed, convex sets in $H = \mathbb{R}^N \times \mathbb{R}^N$:

- $S_1 = \{(e, u) \in H : e = r - Gu\}$
- $S_2 = \{(e, u) \in H : e = 0\}$

under the constraint $S_3 = \{(e, u) \in H : u \in \Omega\}$. Note that, it is normal that $S_1 \cap S_2 \cap S_3$ is either a singleton pair $(e, u) = (0, u^*)$ solving the ILC problem or it is the empty set $\emptyset$. In this second case, perfect tracking is not achievable due to the introducing of input constraint Ω.

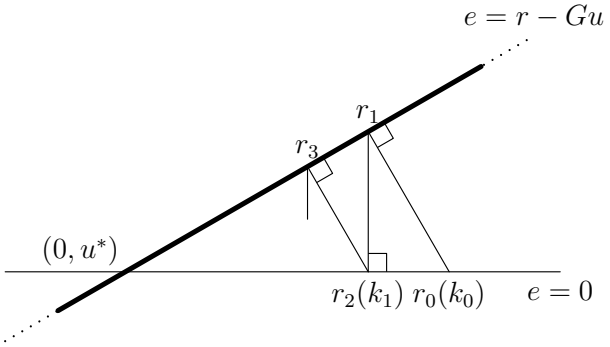
There are three sets in the constrained problem. It seems the results in Owens and Jones [16] can not be directly used. However, set S_3 can be associated with either S_1 (yielding two sets $S_1 \cap S_3$ and S_2) or S_2 (yielding two sets $S_2 \cap S_3$ and S_1) and also notice that the intersection of two closed convex sets is still a closed convex set. Then, the original 3-set problem becomes a 2-set problem, which is to find the intersection of $K_1 = S_1 (resp. S_1 \cap S_3)$ and $K_2 = S_2 \cap S_3 (resp. S_2)$.

The successive projection method in Section 3.1 hence generates two new iterative algorithms for the constrained ILC problem, which are demonstrated in the following two sections. In what follows, we do not specify the exact form of the constraints other than that they are closed and convex.

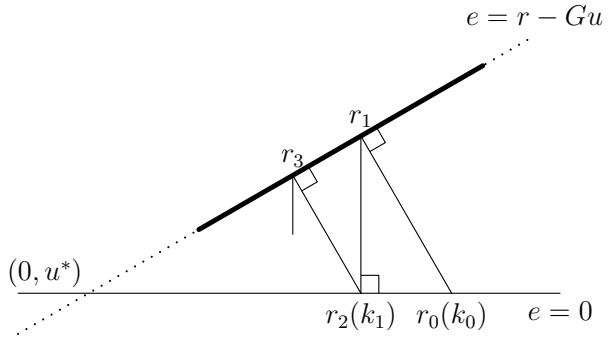
4 Constrained ILC: Algorithm 1

This algorithm identifies that input constraints with the dynamics, a situation that explains why the algorithm is more computationally intensive than its alternative form (introduced later). The formal construction sets $K_1 = S_1 \cap S_3$ and $K_2 = S_2$ to be the closed convex sets in Theorem 1, which can be described as follows.

- $K_1 = \{(e, u) \in H : e = r - Gu, u \in \Omega\}$



(a) $(S_1 \cap S_3) \cap S_2 \neq \emptyset$ and perfect tracking is possible



(b) $(S_1 \cap S_3) \cap S_2 = \emptyset$ and perfect tracking is not possible

Fig. 2. Illustration of Algorithm 1

- $K_2 = \{(e, u) \in H : e = 0\}$

The following algorithm can now be constructed and is illustrated schematically in Figure 2(a) and Figure 2(b), in which the cases when perfect tracking is possible (intersection occurs) or not (the intersection is empty)are shown, respectively.

4.1 Algorithm Description

Algorithm 1 *Given any initial input u_0 satisfying the constraint with associated tracking error e_0, the input sequence $u_{k+1}, k = 0, 1, 2, \cdots$, defined by*

$$u_{k+1} = \arg\min_{u \in \Omega} \left\{ \|r - Gu\|_Q^2 + \|u - u_k\|_R^2 \right\} \tag{15}$$

also satisfies the constraint and iteratively solves the constrained ILC problem.

Proof. According to Theorem 1, let $K_1 = S_1 \cap S_3$ and $K_2 = S_2$. Given $r_0 = (0, u_0) \in K_2$, the sequence $\{r_1, r_2, \cdots\}$ given by

$$\|r_i - r_{i-1}\| = \inf_{y \in K_j} \|y - r_{i-1}\|, \tag{16}$$

where K_j is defined as

$$K_j = \begin{bmatrix} K_1, \ j \ odd \\ K_2, \ j \ even \end{bmatrix},$$

iteratively finds the intersection of K_1 and K_2. The subsequence $\{k_1, k_2, \cdots\} \subset K_2$ defined by

$$k_k = r_{2k}, \tag{17}$$

also iteratively finds the intersection of K_1 and K_2. That is, it solves the ILC problem.

Note that, $k_{k+1} = r_{2(k+1)}$ is solved by

$$\|r_{2k+1} - k_k\| = \inf_{y \in K_1} \|y - k_k\| \tag{18}$$

and

$$\|k_{k+1} - r_{2k+1}\| = \inf_{y \in K_2} \|y - r_{2k+1}\|. \tag{19}$$

Note that (18) is actually solving the following optimization problem

$$\begin{aligned} r_{2k+1} &: (e_{k+1}, u_{k+1}) \\ &= \arg \min_{(e,u) \in K_1} \left\{ \|e - 0\|_Q^2 + \|u - u_k\|_R^2 \right\}, \end{aligned} \tag{20}$$

which is equivalent to solving

$$u_{k+1} = \arg\min_{u \in \Omega} \left\{ \|r - Gu\|_Q^2 + \|u - u_k\|_R^2 \right\} \tag{21}$$

and (19) simply gets $k_{k+1} : (0, u_{k+1})$. That completes the proof. ■

4.2 Convergence Analysis

This section discusses the convergence properties of Algorithm 1. As mentioned in Section 3.2, due to the introducing of the input constraints Ω, there may be no intersection of S_1, S_2 and S_3, which means perfect tracking of the reference signal may be not possible. In this case, the convergence properties may have some difference. Hence the convergence results are presented in two parts: $(S_1 \cap S_3) \cap S_2 \neq \emptyset$ and $(S_1 \cap S_3) \cap S_2 = \emptyset$.

4.2.1 $(S_1 \cap S_3) \cap S_2 \neq \emptyset$

In this case, perfect tracking of the reference signal is possible. Algorithm 1 has the highly desirable property that the norm of the tracking error will decrease monotonically, which is shown in the following theorem.

Theorem 2 *When perfect tracking is possible, Algorithm 1 can achieve monotonic convergence to zero tracking error, that is*

$$\|e_{k+1}\| \leq \|e_k\|, k = 0, 1, \cdots. \tag{22}$$

and

$$\lim_{k\to\infty} e_k = 0, \lim_{k\to\infty} u_k = u^*. \tag{23}$$

Proof. Monotonic convergence can be easily obtained from the algorithm itself. Note that, at $k+1$ iteration, the input u_{k+1} is given by

$$u_{k+1} = \arg\min_{u\in\Omega} \left\{ \|r - Gu\|_Q^2 + \|u - u_k\|_R^2 \right\}. \tag{24}$$

Define

$$J_{k+1}(u) = \|r - Gu\|_Q^2 + \|u - u_k\|_R^2. \tag{25}$$

Then, it is easy to see

$$\begin{aligned} J_{k+1}(u_k) &= \|e_k\|^2 \geq J_{k+1}(u_{k+1}) \\ &= \|e_{k+1}\|^2 + \|u_{k+1} - u_k\|^2 \geq \|e_{k+1}\|^2 \end{aligned} \tag{26}$$

Hence we have

$$\|e_{k+1}\| \leq \|e_k\|. \tag{27}$$

Zero tracking error is got by noticing that Algorithm 1 iteratively finds the intersection of $K_1 = S_1 \cap S_3$ and $K_2 = S_2$, which is $(0, u^*)$ when $(S_1 \cap S_3) \cap S_2 \neq \emptyset$, and hence, achieves perfect tracking. That completes the proof. ■

Algorithm 1 also has the desirable property that the distance between the k^{th} input and the solution u^* is decreasing monotonically, as proved in the following theorem:

Theorem 3 *When perfect tracking is possible, Algorithm 1 has the property that, for all $k \geq 0$ and for all u_0 and u^**

$$\|u_{k+1} - u^*\| \leq \|u_k - u^*\|, \tag{28}$$

i.e., the input iterates approach the solution monotonically in norm.

Proof. According to Theorem 1 and the proof of Algorithms 1, and given that $x \in K_1 \cap K_2 = (0, u^*)$, then

$$\|k_k - x\|^2 \geq \|r_{2k+1} - x\|^2 \geq \|k_{k+1} - x\|^2. \tag{29}$$

As x is $(0, u^*)$, k_k is $(0, u_k)$ and k_{k+1} is $(0, u_{k+1})$, it immediately follows that

$$\|u_{k+1} - u^*\| \leq \|u_k - u^*\|. \tag{30}$$

as required. ■

4.2.2 $(S_1 \cap S_3) \cap S_2 = \emptyset$

In this case, perfect tracking is not possible. The algorithm does however compute an approximation of the unconstrained input u^*. For the convergence of the tracking error, the following theorem holds.

Theorem 4 *When perfect tracking is not possible, Algorithm 1 converges to point u_s^* which is uniquely defined by the following optimization problem*

$$u_s^* = \arg\min_{u\in\Omega} \|r - Gu\|_Q^2. \tag{31}$$

Moreover, this convergence is monotonic in the tracking error, that is,

$$\|e_{k+1}\| \leq \|e_k\|, k = 0, 1, \cdots. \tag{32}$$

Proof. According to Theorem 1, when $(S_1 \cap S_3) \cap S_2 = \emptyset$, that is, perfect tracking can not be achieved, Algorithms 1 will converge to point u_s^*, where $r_1 = (e, u) \in K_1, r_2 = (0, u_s^*) \in K_2$ defining the minimum distance of the two sets, which is the solution of the following optimization problem

$$(r_1, r_2) = \arg \min_{r_1\in K_1, r_2\in K_2} \|r_1 - r_2\|^2. \tag{33}$$

Remember the definition of K_1 and K_2, (33) is equivalent to solve

$$(u, u_s^*) = \arg \min_{u\in\Omega, u_0} \left\{ \|r - Gu\|_Q^2 + \|u - u_0\|_R^2 \right\}. \tag{34}$$

Hence, Algorithm 1 converges to point u_s^*, which is defined by

$$\begin{aligned} u_s^* &= \arg \min_{u\in\Omega, u_0} \left\{ \|r - Gu_0\|_Q^2 + \|u_0 - u\|_R^2 \right\} \\ &= \arg\min_{u\in\Omega} \left\{ \min_{u_0} \|r - Gu_0\|_Q^2 + \|u_0 - u\|_R^2 \right\}. \end{aligned} \tag{35}$$

Notice that the inner minimization has the solution

$$u_0 = u. \tag{36}$$

Hence, substitute (36) into (35) and the optimization problem can be transformed into

$$u_s^* = \arg\min_{u\in\Omega} \|r - Gu\|_Q^2. \tag{37}$$

Note that G is invertible, then the performance index to be minimized is strictly convex, also notice that the constraint is convex, hence this quadratic programming problem has the unique solution.

The proof of monotonic convergence is similar to that of $(S_1 \cap S_3) \cap S_2 \neq \emptyset$ and is omitted here. That completes the proof. ■

Remark 1 *From the discussion above, it can be seen that Algorithm 1 has the appealing property of monotonic convergence of the tracking error. The main difficulty with Algorithm 1 is the solution of the constrained quadratic programming (QP) problem (15). In practice, the dimension of the time series u_k and plant operator G may be very large and the QP problem will be difficult or even become unmanageable. This is discussed in the next section and two methods are given as possible solutions.*

4.3 Solution of the Subproblem

As mentioned above, the solving of the large QP problem is the main obstacle in applying Algorithm 1. In this section, two methods are given to solve the problem, that is, iterative solution and receding horizon method.

4.3.1 Using Iterative Algorithms

There are a number of iterative algorithms in the literature to solve large scale QP problem, see [17], [18], [19], [20]. Here, the Goldstein-Levitin-Polyak (GLP) method is introduced [17], [18].

The GLP method minimizes a continuously differentiable function $f : H \to R$ over a closed convex set $\Omega \subset H$ using the iterative algorithm

$$x_{k+1} = P_\Omega \left[x_k - a_k \nabla f(x_k) \right], k = 0, 1, \cdots \tag{38}$$

where $P_\Omega(z)$ denote the projection of $z \in H$ onto Ω, $\nabla f(x_k)$ denotes the gradient of f at x_k and $a_k \geq 0$ is the step size and should satisfy

$$0 < \epsilon \leq a_k \leq \frac{2(1-\epsilon)}{L}, \forall k \tag{39}$$

where L is a Lipschitz constant satisfying

$$|\nabla f(x) - \nabla f(y)| \leq L|x - y|, \forall x, y \in \Omega. \tag{40}$$

The convergence properties of this algorithm are also included in [17] and omitted here. One appealing property is that $f(x_k)$ will decrease monotonically, which is very useful in ILC problem.

Algorithm 1 requires the solution of a QP problem, for which the gradient and the Lipschitz constant can be easily got. Also notice that the constraint is rather simple and the projection can be carried on conveniently. Hence, the GLP method can be used to solve the QP problem arising in Algorithm 1. For the convergence property of this method, we have the following theorem.

Theorem 5 *Using Goldstein-Levitin-Polyak method to solve the constrained QP problem, Algorithm 1 can maintain monotonic convergence in the tracking error.*

Proof. Define

$$J_{k+1}(u) = \|r - Gu\|_Q^2 + \|u - u_k\|_R^2 \tag{41}$$

Then, at trial $k+1$, the algorithm minimizes the above performance index subject to input constraints using Goldstein-Levitin-Polyak method and the initial point is u_k. Note that using Goldstein-Levitin-Polyak method, $J_{k+1}(u)$ will decrease monotonically. Suppose the algorithm stops and gives the $k+1^{th}$ input u_{k+1}, we have

$$J_{k+1}(u_k) \geq J_{k+1}(u_{k+1}), \tag{42}$$

which is actually

$$\|e_k\|^2 \geq \|e_{k+1}\|^2 + \|u_{k+1} - u_k\|^2. \tag{43}$$

We immediately get

$$\|e_k\| \geq \|e_{k+1}\| \tag{44}$$

and that completes the proof. ■

Remark 2 *In practice, due to the computational expenses, we can't wait too many iterations to get the exact solution. Then, a prescribed maximum iteration or an expected accuracy can be given as criteria to terminate the GLP algorithm.*

4.3.2 Receding Horizon Method

Another approach to solve the QP problem is using receding horizon method, which is also noticed in [5]. The main idea is introduced here, and for more details please refer to [21].

At iteration $k+1$, the following problem needs to be solved

$$u_{k+1} = \arg\min_{u \in \Omega} \left\{ \|r - Gu\|^2 + \|u - u_k\|^2 \right\} \tag{45}$$

The difficulty of this problem lies in the possible large dimension of u and G. The receding horizon method, however solves this problem approximately through solving a series of smaller scale QP problems:

(1) At time t and for the current state x_t , solve an optimal control problem over a fixed future interval, say $[t; t + N_u - 1]$, taking into account the constraints.
(2) Apply only the first step in the resulting optimal control sequence.
(3) Measure the state reached at time $t + 1$.
(4) Repeat the fixed horizon optimization at time $t+1$ over the future interval $[t+1; t+N_u]$, starting from the (now) current state x_{t+1}.

Then, the original problem is reduced to a series of following QP problems in Step 1:

$$u_{k+1,t}^{opt} = \arg \min_{u_{k+1,t} \in \Omega} \left\{ \begin{array}{l} \sum_{i=t}^{t+N_u-1} \|r(i) - y_{k+1}(i)\|^2 \\ +\|u_{k+1,t}(i) - u_k(i)\|^2 \end{array} \right\} \quad (46)$$

where

$$u_{k+1,t} = \left[u_{k+1}(t) \;\; u_{k+1}(t+1) \;\; \cdots \;\; u_{k+1}(t+N_u-1) \right]^T.$$

Note that this problem is of small size and can be solved easily.

The choosing of N_u is very important. If $N_u = N$, (46) becomes the original problem (45). Large value of N_u will give more accurate solution at the cost of large computational load while too small value may result in poor performance. There are a number of results in the literature on how to choose the horizon, please refer to [21].

Note that, unlike Goldstein-Levitin-Polyak method, using receding horizon control, Algorithm 1 may lose the monotonic convergence in the tracking error norm.

Remark 3 *In this section, two methods are given to solve the large size constrained QP problem. There are also many other algorithms that can be used. It should be kept in mind that due to the practical restrictions, only an approximate solution of the QP problem can be got. Using this solution, the appealing convergence properties of Algorithms 1 may not be maintained, depending on the property of the methods used.*

4.4 Effect of Weighting Matrices Q and R

In this section, the effect of weighting matrices Q and R on the convergence properties of Algorithm 1 is discussed.

According to (15), the weighting matrices Q and R provide scaling on the tracking error and the change of input. Intuitively, if Q is fixed, then a smaller R implies larger acceptable change of input, and which in turn, results in smaller tracking error. This leads to faster convergence rate.

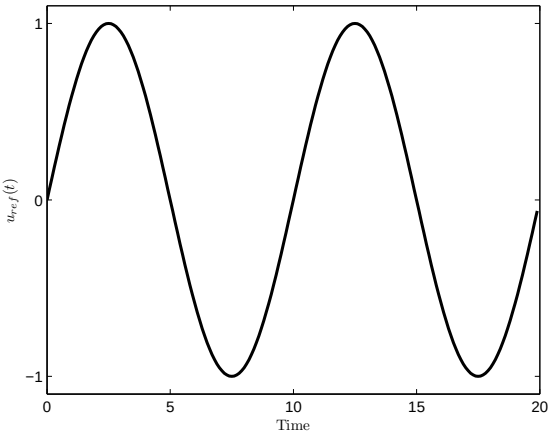


Fig. 3. The input signal

Consider SISO systems with scalar weighting Q and R. Choose $Q = 1$ and consider the effect of R on algorithm performance. The following results can be easily derived. Whether perfect tracking is possible or not, it can be expected (as with NOILC) that smaller R will result in faster convergence rate. The algorithm will converge to the solution of (31), which is independent of R. This implies that the weighting matrix R has no effect on the asymptotic accuracy but only affects the convergence rate. This is illustrated by the following example.

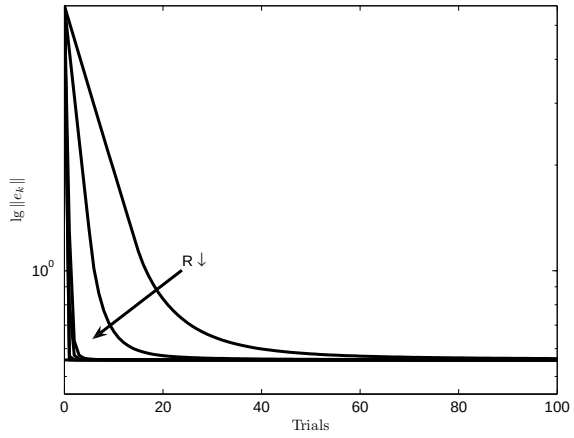
Example 1 *Consider the following simple second order system*

$$G(s) = \frac{s-4}{s^2+5s+6}, \quad (47)$$

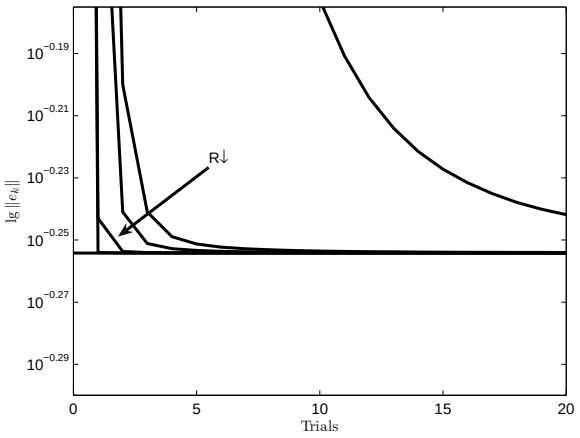
which is sampled using a zero-order hold and a sampling time of 0.1s. The trial length is 20s, zero initial conditions are assumed and the reference signal is generated by the sine-wave input u^* shown in Figure 3. The constraint is taken to be $|u(t)| \leq 0.8, t = 0, 1, \cdots$, which is violated by u^* so that perfect tracking is not possible. The initial input is chosen to be $u_0 = 0$. The simulation is designed to evaluate the effect of weighting matrices on the performance of Algorithm 1 over 100 iterations. Simulations are run in six cases with weighting chosen to be $Q = 1$ and $R = 3, 1, 0.1, 0.05, 0.01, 0.001$, respectively. The norms of the tracking error for each test are plotted and shown in Figure 4.

From the figure, it can be seen that the weighting matrix R have no effect on the asymptotic accuracy. However, smaller values of R result in faster convergence, which verifies our expectations.

When the system is MIMO or the weighting matrices are not scalar, the analysis of the effect of weighting matrices is expected to indicate a similar but more complex pattern.



(a) Original figure



(b) Magnified figure

Fig. 4. Effect of different weighting matrices on convergence performance

5 Constrained ILC: Algorithm 2

In this section, an alternative algorithm is given by taking $K_1 = S_1$ and $K_2 = S_2 \cap S_3$ to be the closed, convex sets in Theorem 1, which can be expressed as follows.

- $K_1 = \{(e, u) \in H : e = r - Gu\}$
- $K_2 = \{(e, u) \in H : e = 0, u \in \Omega\}$

The following alternative algorithm to Algorithm 1 can be constructed and is illustrated schematically in Figure 5(a) and Figure 5(b).

5.1 Algorithm Description

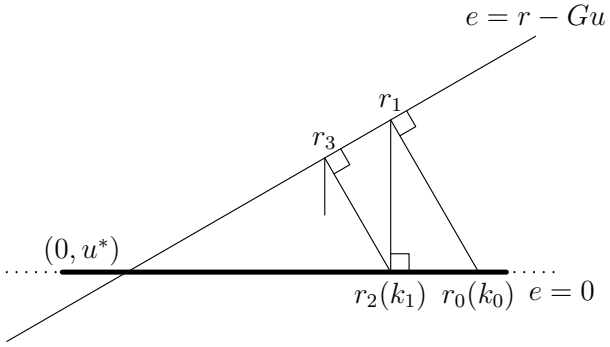
Algorithm 2 *Given any initial input u_0 satisfying the constraint with associated tracking error e_0, the input sequence $u_{k+1}, k = 0, 1, 2, \cdots$, defined by the solution of the input unconstrained NOILC optimization problem*

$$\tilde{u}_k = \arg\min_u \left\{ \|r - Gu\|_Q^2 + \|u - u_k\|_R^2 \right\} \tag{48}$$

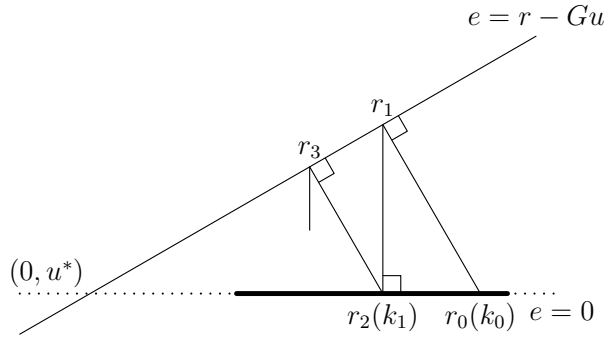
followed by the simple input projection

$$u_{k+1} = \arg\min_{u \in \Omega} \|u - \tilde{u}_k\| \in \Omega \tag{49}$$

also satisfies the constraint and iteratively solves the constrained ILC problem.



(a) $S_1 \cap (S_2 \cap S_3) \neq \emptyset$ and perfect tracking is possible



(b) $S_1 \cap (S_2 \cap S_3) = \emptyset$ and possible tracking is not possible

Fig. 5. Illustration of Algorithm 2

Proof. According to Theorem 1, let $K_1 = S_1$ and $K_2 = S_2 \cap S_3$. Given $r_0 = (0, u_0) \in K_2$, the sequence $\{r_1, r_2, \cdots\}$ given by

$$\|r_i - r_{i-1}\| = \inf_{y \in K_j} \|y - r_{i-1}\|, \tag{50}$$

where K_j is defined as

$$K_j = \begin{bmatrix} K_1, \ j \ odd \\ K_2, \ j \ even \end{bmatrix},$$

iteratively finds the intersection of K_1 and K_2. Then, the sub-sequence $\{k_1, k_2, \cdots\} \subset K_2$ defined by

$$k_k = r_{2k}, \tag{51}$$

also iteratively finds the intersection of K_1 and K_2 i.e. it solves the ILC problem.

Note that, $k_{k+1} = r_{2(k+1)}$ is solved by

$$\|r_{2k+1} - k_k\| = \inf_{y \in K_1} \|y - k_k\| \tag{52}$$

and

$$\|k_{k+1} - r_{2k+1}\| = \inf_{y \in K_2} \|y - r_{2k+1}\|. \tag{53}$$

Note that (52) is actually solving the following optimization problem

$$\begin{aligned} r_{2k+1} &: (\tilde{e}_k, \tilde{u}_k) \\ &= \arg \min_{(e,u) \in K_1} \left\{ \|e - 0\|_Q^2 + \|u - u_k\|_R^2 \right\} \end{aligned} \tag{54}$$

which is the solution of NOILC and (53) simply gives $k_{k+1} : (0, u_{k+1})$, where

$$u_{k+1} = \arg \min_{u \in \Omega} \|u - \tilde{u}_k\|. \tag{55}$$

That completes the proof. ■

Remark 4 *Note that the second step of Algorithm 2 requires the solution of the problem (49). It seems this may need the application of some optimization methods. However, in practice the input constraint Ω is often a pointwise constraint and the solution of (49) can be computed easily. For example, when $\Omega = \{u \in H : |u(t)| \leq M(t)\}$, the solution is simply as follows,*

$$u_{k+1}(t) = \begin{cases} M(t) & : \tilde{u}_k(t) > M(t) \\ \tilde{u}_k(t) & : |\tilde{u}_k(t)| \leq M(t) \\ -M(t) & : \tilde{u}_k(t) < -M(t) \end{cases}, \tag{56}$$

for $t = 0, \cdots, N-1$.

5.2 Convergence Analysis

This section discusses the convergence properties of Algorithm 2. As for Algorithm 1, the results are presented in two parts: $(S_1 \cap S_3) \cap S_2 \neq \emptyset$ and $(S_1 \cap S_3) \cap S_2 = \emptyset$.

5.2.1 $(S_1 \cap S_3) \cap S_2 \neq \emptyset$

In this case, perfect tracking of the reference signal is possible with a unique input u^*. The theorem below directly follows from Theorem 1.

Theorem 6 *When perfect tracking is possible, Algorithm 2 solves the ILC problem in the sense that*

$$\lim_{k \to \infty} e_k = 0, \lim_{k \to \infty} u_k = u^*. \tag{57}$$

Moreover, this convergence is monotonic with respect to the following performance index,

$$J_k = \|Ee_k\|_Q^2 + \|Fe_k\|_R^2 \tag{58}$$

where

$$\begin{aligned} e_k &= r - Gu_k \\ E &= I - G\left(G^TQG + R\right)^{-1} G^TQ\,. \\ F &= \left(G^TQG + R\right)^{-1} G^TQ \end{aligned} \tag{59}$$

Proof. Equation (57) can be easily deduced from Theorem 1. Algorithm 2 iteratively finds the intersection of $K_1 = S_1$ and $K_2 = S_2 \cap S_3$, which is $(0, u^*)$ when $S_1 \cap (S_2 \cap S_3) \neq \emptyset$, and hence, achieves perfect tracking.

Monotonic convergence with respect to the defined performance index can be obtained as follows. According to Theorem 1 and the proof of Algorithms 2, the distance between $\{k_0, r_1, k_1, r_2, \cdots\}$ is decreasing, that is

$$\begin{aligned} \|k_k - r_{2k+1}\| &\geq \|r_{2k+1} - k_{k+1}\| \\ &\geq \|k_{k+1} - r_{2(k+1)+1}\| \end{aligned} \tag{60}$$

Note the left side is actually the minimum distance between k_k and K_1, which is

$$\|k_k - r_{2k+1}\| = \min_u \left\{ \|r - Gu\|_Q^2 + \|u - u_k\|_R^2 \right\} \tag{61}$$

Note that this is the NOILC solution

$$u_{k_r} = u_k + \left(G^TQG + R\right)^{-1} G^TQ\,(r - Gu)\,. \tag{62}$$

Substituting the solution, (61) can be further written as

$$\|k_k - r_{2k+1}\| = \min_u \left\{ \|Ee_k\|^2 + \|Fe_k\|^2 \right\} \tag{63}$$

with E, F defined as (59). Note that this is performance index J_k. Similarly, the right side of (60) is J_{k+1}. Then according to (60), we have

$$J_{k+1} \leq J_k \tag{64}$$

That is, performance index J_k is decreasing monotonically, which completes the proof. ■

Algorithm 2 first computes the NOILC solution and then projects this solution onto the constraint. This approach is much simpler than the previously described algorithm in the sense that the computational load is much less and hence is a simpler way to implement successive projection in practice. Intuitively, this strategy may, however, lead to other problems such as a slower convergence rate.

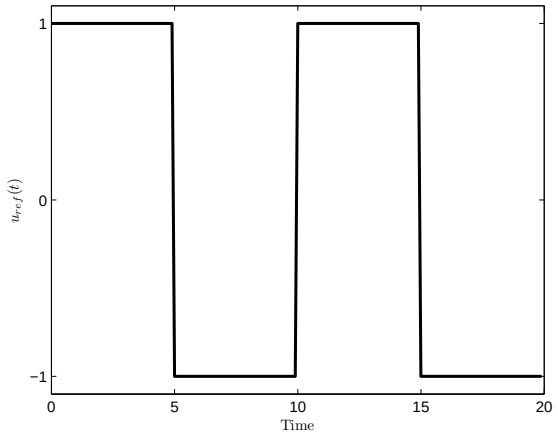


Fig. 6. The input signal

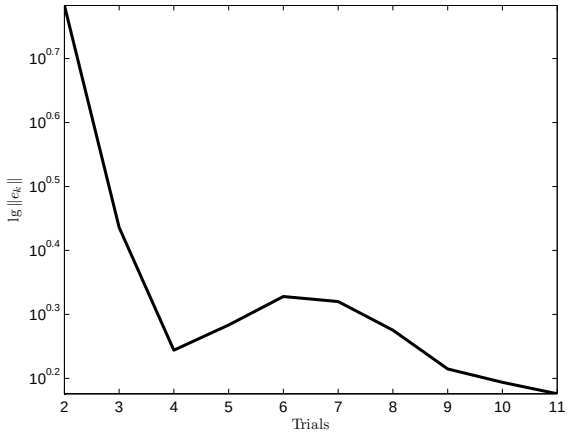


Fig. 7. The tracking performance of Algorithm 2

It is well-known that NOILC achieves monotonic convergence in the tracking error. The following example shows that, Algorithm 2, however, may not have this property.

Example 2 *Consider the following system*

$$G(s) = \frac{4.2130s - 2.5164}{s^2 - 0.1312s + 3.6624}, \tag{65}$$

which is sampled using a zero-order hold and a sampling time of 0.1s. The trial length is 20s, zero initial conditions are assumed and the reference signal is generated by the square-wave input shown in Figure 6. The constraint is $|u(t)| \leq 1, t = 0, 1, \cdots$, which is satisfied by the input u^* so that perfect tracking is possible. The initial input is chosen to be $u_0 = 0$. The simulation evaluates the performance of Algorithm 2 over 100 iterations. The weighting matrices are chosen to be $Q = R = I$ for simplicity. The norm of the tracking error from 2^{th} to 11^{th} iteration is plotted and shown in Figure 7.

From the figure, it is clear that Algorithm 2 may not produce monotonic convergence in the tracking error norm.

Although Algorithm 2 may not maintain monotonic convergence in the tracking error, it has the property that the distance between the k^{th} input and the optimal solution is decreasing monotonically, which is shown in the following theorem.

Theorem 7 *When perfect tracking is possible, Algorithm 2 has the property that, for all $k \geq 0$ and for all u_0 and u^**

$$\|u_{k+1} - u^*\| \leq \|u_k - u^*\|, \tag{66}$$

i.e., the input iterates approach the solution monotonically in norm.

Proof. The proof is similar to that of Algorithm 1 and is omitted here. ■

5.2.2 $(S_1 \cap S_3) \cap S_2 = \emptyset$

In this case, perfect tracking is not possible and only an approximation of the original input u^* can be achieved. The following theorem describes algorithm behaviour.

Theorem 8 *When perfect tracking is not possible, Algorithm 2 converges to point u_s^* which is uniquely defined by the following optimization problem,*

$$u_s^* = \arg\min_{u \in \Omega} \left\{ \|Ee\|_Q^2 + \|Fe\|_R^2 \right\}. \tag{67}$$

Moreover, this convergence is monotonic with respect to the following performance index,

$$J_k = \|Ee_k\|_Q^2 + \|Fe_k\|_R^2 \tag{68}$$

where

$$\begin{aligned} e &= r - Gu \\ E &= I - G\left(G^TQG + R\right)^{-1} G^TQ\,. \\ F &= \left(G^TQG + R\right)^{-1} G^TQ \end{aligned} \tag{69}$$

Proof. According to Theorem 1, when $S_1 \cap (S_2 \cap S_3) = \emptyset$, that is, perfect tracking can not be achieved, Algorithms 2 will converge to a point u_s^*, where $r_1 = (e, u) \in K_1, r_2 = (0, u_s^*) \in K_2$ defining the minimum distance between the two sets, which is the solution of the following optimization problem

$$(r_1, r_2) = \arg \min_{r_1 \in K_1, r_2 \in K_2} \|r_1 - r_2\|^2. \tag{70}$$

Remember the definition of K_1 and K_2, (70) is equivalent to solve

$$(u, u_s^*) = \arg \min_{u\in\Omega, u_0} \left\{ \|r - Gu_0\|_Q^2 + \|u_0 - u\|_R^2 \right\}. \quad (71)$$

Hence, Algorithm 2 converges to point u_s^*, which is defined by

$$\begin{aligned} u_s^* &= \arg \min_{u\in\Omega, u_0} \left\{ \|r - Gu_0\|_Q^2 + \|u_0 - u\|_R^2 \right\} \\ &= \arg \min_{u\in\Omega} \left\{ \min_{u_0} \|r - Gu_0\|_Q^2 + \|u_0 - u\|_R^2 \right\}. \end{aligned} \quad (72)$$

Notice that the inner minimization is the solution of NOILC and is given by

$$u_0 = u + \left(G^T Q G + R\right)^{-1} G^T Q \left(r - Gu\right) \quad (73)$$

Hence, substitute (73) into (72) and the optimization problem can be transformed into

$$u_s^* = \arg \min_{u\in\Omega} \left\{ \|Ee\|_Q^2 + \|Fe\|_R^2 \right\} \quad (74)$$

where

$$\begin{aligned} e &= r - Gu \\ E &= I - G\left(G^T Q G + R\right)^{-1} G^T Q\,. \\ F &= \left(G^T Q G + R\right)^{-1} G^T Q \end{aligned} \quad (75)$$

Note that E and F are invertible, then the performance index to be minimized is strictly convex, also notice that the constraint is convex, hence this quadratic programming problem has unique solution.

The proof of monotonic convergence with respect to J_k is similar to that of Theorem 6 and omitted here, which completes the proof. ■

Remark 5 *For the constrained ILC problem, the best result we can achieve in terms of tracking error is defined by the following QP problem*

$$u^* = \arg \min_{u\in\Omega} \|r - Gu\|^2. \quad (76)$$

Compared to Theorem 8, it can be found that Algorithm 2 actually minimizes weighted norm of tracking error. In this case, only nearly optimal performance can be achieved.

5.3 Effect of Weighting Matrices Q and R

In this section, the effect of weighting matrices Q and R on the convergence properties of Algorithm 2 is discussed. As with Algorithm 1, the effect is illustrated in an intuitive way.

Consider SISO systems with scalar weighing Q and R. Choose $Q = 1$ and consider the effect of variation of R. When perfect tracking is possible, perfect tracking can be achieved and smaller R will result in faster convergence. When perfect tracking is not possible, reducing R will again result in faster convergence rate but the asymptotic error changes (in contrast to Algorithm 1). This can be explained as follows. Algorithm 2 converges to the solution of the following problem

$$\begin{aligned} u_s^* = \arg\min\nolimits_{u\in\Omega} \Big\{ &\|(I - G\left(G^T Q G + R\right)^{-1} G^T Q)e\|_Q^2 \\ &+ \|\left(G^T Q G + R\right)^{-1} G^T Q e\|_R^2 \Big\} \end{aligned}$$

When $R \to \infty$, the first term of the last equation becomes $\|e\|_Q^2$ and the second term becomes zero. Hence, the optimization problem becomes

$$u_s^* = \arg \min_{u\in\Omega} \|e\|_Q^2 \quad (77)$$

This is the constrained optimal solution and is the best result that can be achieved with constrained control. However, in this case, since the weighting of input change R is very large, the convergence rate is expected to be very slow. One the other hand, when $R \to 0$, it can be seen the first term of the last equation becomes zero and the second term becomes $\|G^{-1}e\|_R^2$, which can be further written as $\|u - u^*\|_R^2$, where u^* is the unique input generating the reference signal. Hence, the optimization problem becomes

$$u_s^* = \arg \min_{u\in\Omega} \|u - u^*\|_R^2 \quad (78)$$

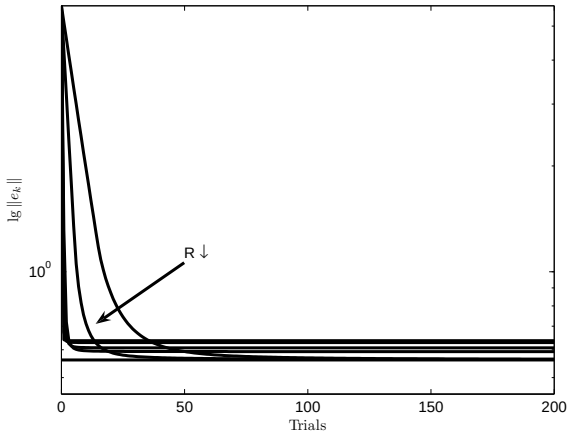
This is just the projection of u^* onto the constraint set Ω. Clearly the tracking error may be larger than that of the constrained optimal solution. However, in this case, the convergence rate is fast.

From the discussion above, it can be seen that when perfect tracking is not possible, the weighting matrix R provides a compromise between the convergence rate and the tracking performance, which is very different from that of Algorithm 1. This is illustrated in the following example.

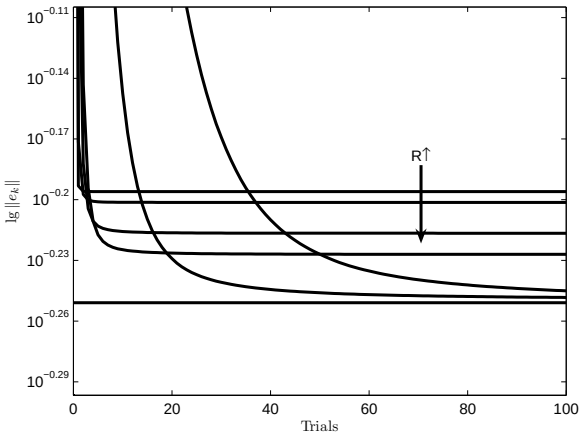
Example 3 *Consider the following system*

$$G(s) = \frac{s + 4}{s^2 + 5s + 6}, \quad (79)$$

which is sampled using a zero-order hold and a sampling time of 0.1s. The trial length is 20s, zero initial conditions are assumed and the reference signal is generated by the sine-wave input shown in figure 3. The constraint set is defined by $|u(t)| \le 0.8, t = 0, 1, \cdots$, which doesn't contain the input u^*. The initial input is chosen to be $u_0 = 0$.



(a) Original figure



(b) Magnified figure

Fig. 8. Effect of different weighting matrices on convergence performance

The simulation aims to investigate the effect of weighting matrices on the performance of Algorithm 1 over 100 iterations. Six simulations are shown with the weighting matrices $Q = I$ and $R = 3, 1, 0.1, 0.05, 0.01, 0.001$, respectively. The results are shown in Figure 8 and Figure 9.

From the figure, it can be seen that smaller R results in faster convergence and the weighting matrix R does have an effect on the asymptotic performance/accuracy with larger values of R giving smaller asymptotic error norms. The asymptotic tracking error norm of Algorithm 2 against different weighting matrices R is also plotted and shown in Figure 10. Note that the lower horizonal line is the tracking error norm with the input (77) and the upper one is (78).

When the system is MIMO or the weighting matrices are not scalar, the effect of weighting matrices would not be so easy to analyze but a similar pattern could be expected.

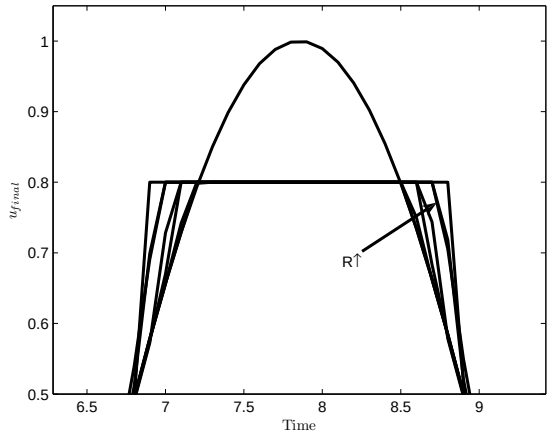


Fig. 9. Part of the resulting input

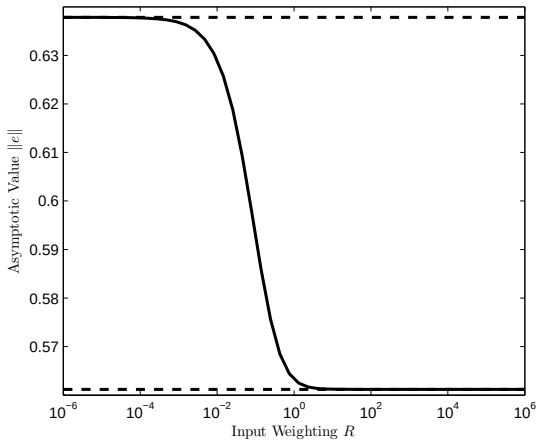


Fig. 10. Effect of different weighting matrices on asymptotic performance

6 Numerical Simulation

In this section, three examples are given to demonstrate the effectiveness of the proposed methods. First, consider the following example where perfect tracking is achievable.

Example 4 *Consider the following non-minimum phase system*

$$G(s) = \frac{s-4}{s^2+5s+6}, \tag{80}$$

which is sampled using a zero-order hold and a sampling time of 0.1s. The trial length is 20s, zero initial conditions are assumed and the reference signal is generated by

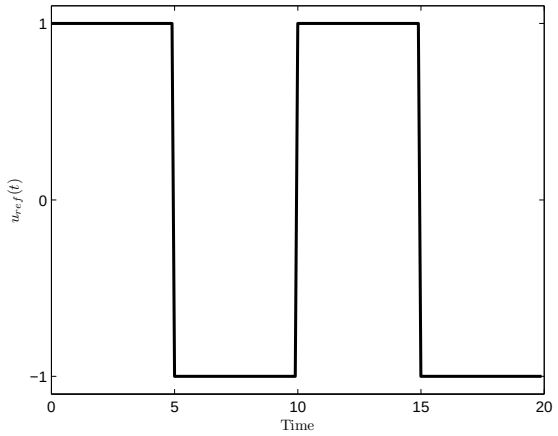


Fig. 11. The input signal

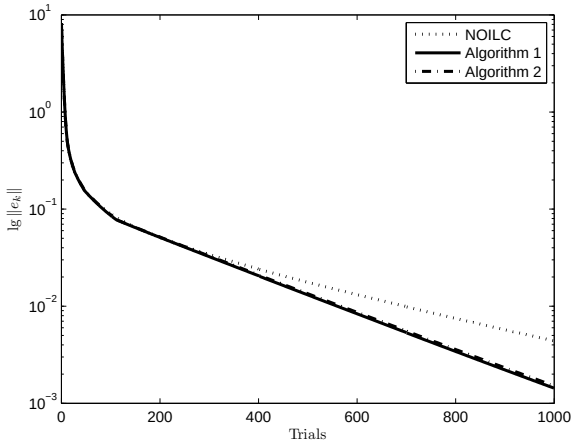


Fig. 12. Comparison of convergence

the square-wave input shown in figure 11. The constraint is $|u(t)| \leq 1, t = 0, 1, \cdots$, *which just contains the input* u^*. *The initial input is chose to be* $u_0 = 0$. *The simulation compares the NOILC, Algorithm 1 and Algorithm 2 with over* 1000 *iterations. For simplicity, the weighting matrices are chosen to be* $Q = R = I$. *The results are shown in Figure 12 and Figure 13.*

Note that in this example, perfect tracking is possible. According to Theorem 2 and Theorem 6, perfect tracking can be achieved by both algorithms. However, it is expected that the constraint will be active during the iterations, which means the resulting input of NOILC may violate the constraint.

From Figure 12, it can be seen that Algorithm 1 and Algorithm 2 is approaching perfect tracking, which verifies the previous expectations. During the first iterations, as $u_0 = 0$, u_k *increases in point-wise magnitude gradually and doesn't violate the constraint in any of the three algorithms. In subsequent iterations, the input computed using NOILC then begins to violate the constraint and differences begin to emerge. It is interesting to see that Algorithm 1 and Algorithm 2 outperform NOILC at this stage. It can also be seen that Algorithm 1 performs a little better than Algorithm 2.*

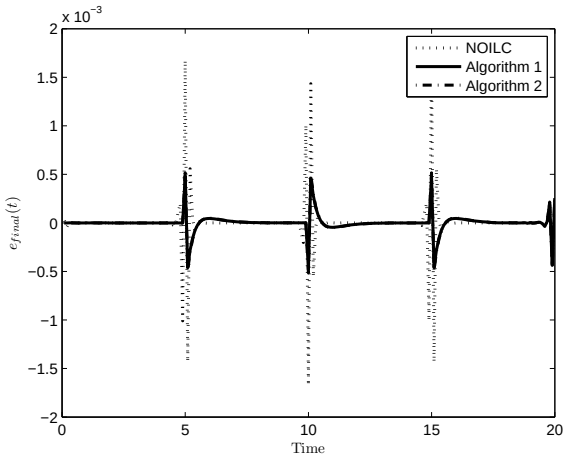


Fig. 13. The tracking error at 1000^{th} iteration

The second example is to illustrate what will happen if perfect tracking is not possible.

Example 5 *Consider the same non-minimum phase system*

$$G(s) = \frac{s-4}{s^2+5s+6}, \tag{81}$$

which is sampled using a zero-order hold and a sampling time of 0.1s. The trial length is 20s, zero initial conditions are assumed and the reference signal is generated by the sine-wave input as shown in Figure 14. The constraint is replaced by $|u(t)| \leq 0.8, t = 0, 1, \cdots$, *so that perfect tracking is not possible. The initial input is chosen to be* $u_0 = 0$. *The simulation compares Algorithm 1, Algorithm 2 and the constrained optimal (76) over 200 iterations. The weighting matrices are chosen to be* $Q = R = I$ *for simplicity. In Algorithms 1, the constrained QP problem is solved by the Matlab optimization toolbox. The results are shown in Figure 15, Figure 16 and Figure 17.*

From Figure 15, it can be seen that Algorithm 1 does converge to the constrained optimal solution, which verifies Theorem 4. Algorithm 2 converges to the solution of (67), which verifies Theorem 8. It can be seen that Algorithm 2 converges faster than Algorithm 1, which is not difficult to understand. Figure 17 shows the original input and resulting input of the three algorithms at the $200th$ *iteration. It can be seen that the resulting input of Algorithm 1 converges to the constrained optimal solution, while Algorithm 2 doesn't. It is also noticed that the resulting final input of Algorithm 2 is not just putting saturation*

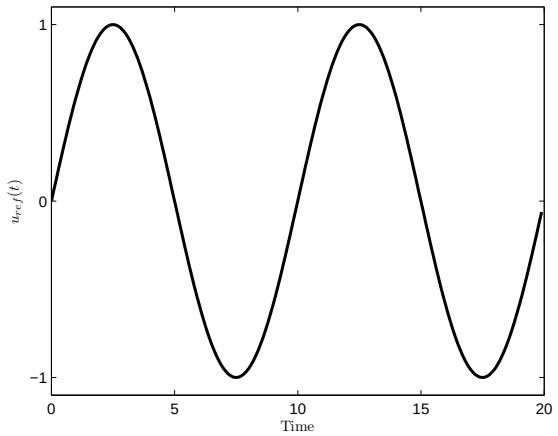
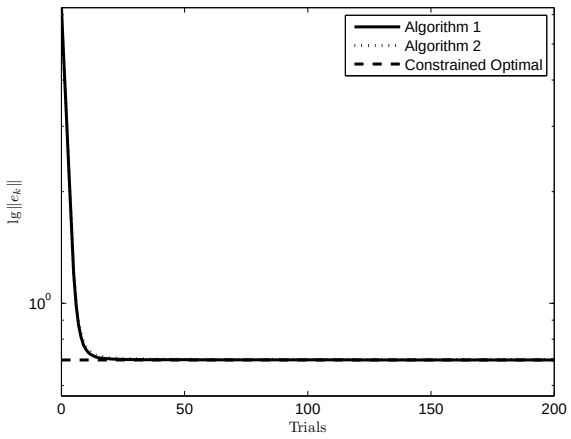
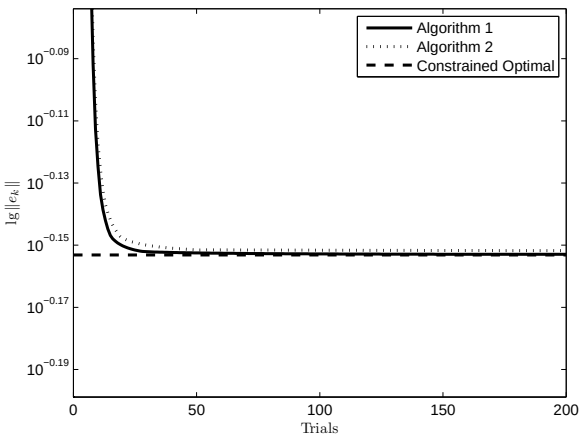


Fig. 14. The input signal



(a) Original figure



(b) Magnified figure

Fig. 15. Comparison of convergence

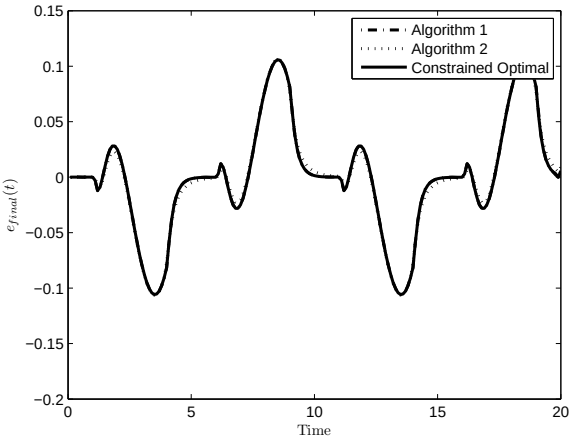


Fig. 16. The tracking error at 200^{th} iteration

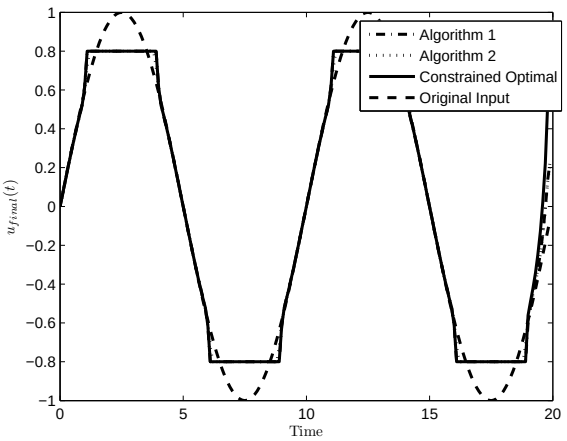


Fig. 17. The resulting input at 200^{th} iteration

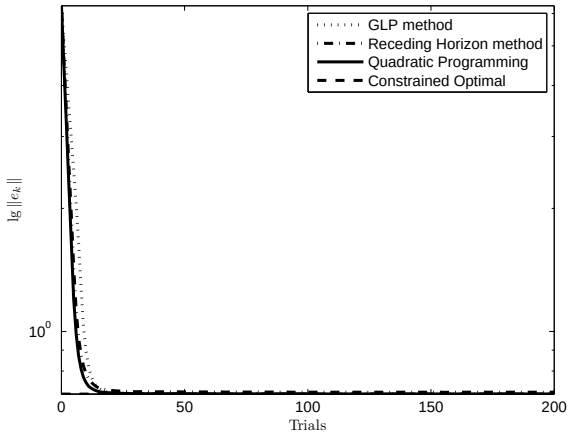
on the original input, instead, it adds some compensation. It should be kept in mind that although Algorithm 1 gives better performance, this is achieved at the expense of large computation load, which may be not acceptable in the real application, whereas Algorithm 2 achieves nearly optimal performance using quite simple computation.

The third example is to illustrate alternative solution methods of Algorithm 1 when perfect tracking is not possible.

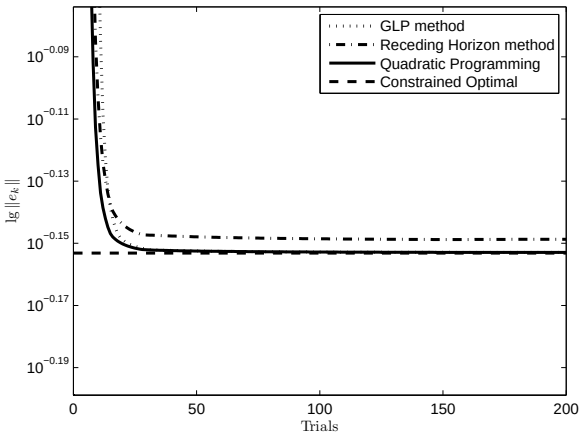
Example 6 *Again, consider the same non-minimum phase system*

$$G(s) = \frac{s-4}{s^2+5s+6}, \tag{82}$$

which is sampled using zero-order hold and a sampling time of 0.1s. The trial length is 20s and the reference signal is generated by the sine-wave input as shown in Figure 14. The constraint is $|u(t)| \leq 0.8, t = 0, 1, \cdots$, *which*



(a) Original figure



(b) Magnified figure

Fig. 18. Comparison of convergence

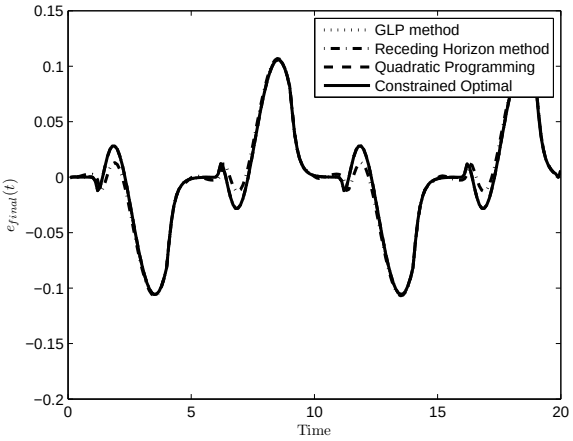


Fig. 19. The tracking error at 200^{th} iteration

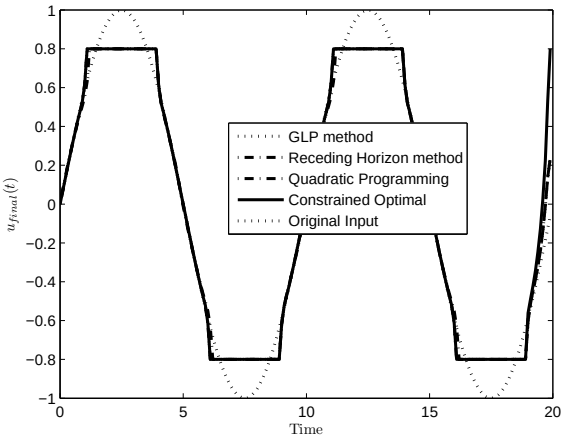


Fig. 20. The resulting input at 200^{th} iteration

doesn't contain the original input and implies that perfect tracking is not possible. The initial input is chosen to be $u_0 = 0$. *The simulation compares the iterative algorithm, receding horizon method and exact solution of the constrained QP problem of Algorithm 1 over 200 iterations. The exact solution is solved by Matlab optimization toolbox. The weighting matrices are chosen to be* $Q = R = I$. *In the iterative algorithm, the algorithm is stopped after 10 iteration. In the receding horizon control method, the horizon is chosen to be* $N_u = 10$. *The results are shown in Figure 18, Figure 19 and Figure 20.*

From Figure 18, it can be seen that, the iterative solution method converges to the constrained optimal solution as $k \to \infty$, *while the receding horizon method doesn't. This is due to the solution accuracy of the receding horizon method. Further simulation shows that when improving the accuracy of the solution by increasing the horizon in the receding horizon method, the limiting point become closer to the constrained optimal solution. It can also be noticed that the convergence rates of iterative solution method and receding horizon control method are lower than that of the exact solution of constrained QP problem. The convergence rate can be further improved by increasing the iteration number in the iterative solution or by enlarging the horizon in the receding horizon method.*

7 Conclusion

Following the success of (unconstrained) norm-optimal iterative learning control, this paper discusses iterative learning control for linear systems with convex input constraints, a situation that approximates to situations met often in practice. First, the constrained ILC problem has been formulated in a novel successive projection framework. Then, based on this projection method, two algorithms have been proposed to solve the constrained ILC problem. It has been shown that, when perfect tracking is possible, both algorithms can achieve perfect tracking whereas one algorithm needs much less

computational effort. When perfect tracking is not possible, both algorithms have been shown to provide useful approximate solutions to the constrained ILC problem but that (1) the asymptotic error will be non-zero and (2) the computational complexity and convergence properties of the algorithms do differ. These observations should be taken into account when choosing the algorithm, which requires a compromise between the performance/accuracy and the computational cost. The effect of weighting matrices on the performance of the algorithms has also been discussed and numerical simulations have been given to demonstrate their effectiveness.

For completeness, two methods are proposed to solve the large scale QP problem arising in Algorithm 1. However, a more accurate and faster solver would be useful. This topic is worthy of further development. There is also more work that needs to be done to extend the results in this paper to nonlinear systems.

Finally, although the presentation has concentrated on sampled data systems (for reasons of both simplicity and practical relevance), the Hilbert space context of successive projection indicates that the ideas and results apply more widely and, in particular, to the case of continuous time systems with no change in the abstract form of the algorithms or results. The realization of these results will however change.

References

[1] D.H. Owens and J. Hatonen. Iterative learning control - an optimization paradigm. *Annual Reviews in Control*, 29(1):57–70, 2005.

[2] D.A. Bristow, M. Tharayil, and A.G. Alleyne. A survey of iterative learning control: A learning-based method for high-performance tracking control. *IEEE Control Systems Magazine*, 26(3):96–114, 2006.

[3] C.T. Chen and S.T. Peng. Learning control of process systems with hard input constraints. *Journal of Process Control*, 9(2):151–160, 1999.

[4] S. Gunnarsson and M. Norrlof. On the design of ILC algorithms using optimization. *Automatica*, 37(12):2011–2016, 2001.

[5] J.H. Lee, K.S. Lee, and W.C. Kim. Model-based iterative learning control with a quadratic criterion for time-varying linear systems. *Automatica*, 36(5):641–657, 2000.

[6] B. Chu and D. H. Owens. Accelerated norm-optimal iterative learning control algorithms using successive projection. *International Journal of Control, to appear*, 2008.

[7] N. Amann, D.H. Owens, and E. Rogers. Iterative learning control using optimal feedback and feedforward actions. *International Journal of Control*, 65(2):277–293, 1996.

[8] J. Hatonen. *Issues of algebra and optimality in iterative learning control*. PhD thesis, University of Oulu, Finland, 2004.

[9] J.J. Hatonen, D.H. Owens, and K.L. Moore. An algebraic approach to iterative learning control. *International Journal of Control*, 77(1):45–54, 2004.

[10] Z. Bien and K.M. Huh. Higher-order iterative learning control algorithm. *IEE Proceedings, Part D: Control Theory and Applications*, 136(3):105–112, 1989.

[11] J. Hatonen, D.H. Owens, and K. Feng. Basis functions and parameter optimisation in high-order iterative learning control. *Automatica*, 42(2):287–294, 2006.

[12] N. Amann, D.H. Owens, and E. Rogers. Predictive optimal iterative learning control. *International Journal of Control*, 69(2):203–226, 1998.

[13] N. Amann. *Optimal algorithms for iterative learning control*. PhD thesis, University of Exeter, UK, 1996.

[14] N. Amann and D.H. Owens. Non-minimum phase plants in norm-optimal iterative learning control. Report, University of Exeter, 1994.

[15] N. Amann, D.H. Owens, and E. Rogers. Iterative learning control for discrete-time systems with exponential rate of convergence. *IEE Proceedings: Control Theory and Applications*, 143(2):217–224, 1996.

[16] D. H. Owens and R. P. Jones. Iterative solution of constrained differential/algebraic systems. *International Journal of Control*, 27(6):957–974, 1978.

[17] A.A. Goldstein. *Constructive real analysis*. New York: Harper & Row, 1967.

[18] D.P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21(2):174–183, 1976.

[19] M. Bierlaire, P.L. Toint, and D. Tuyttens. On iterative algorithms for linear least-squares problems with bound constraints. *Linear Algebra and Its Applications*, 143(1):111–143, 1991.

[20] B.S. He. A projection and contraction method for a class of linear complementarity-problems and its application in convex quadratic-programming. *Applied Mathematics and Optimization*, 25(3):247–262, 1992.

[21] J.A. Rossiter. *Model-based predictive control: a practical approach*. CRC press, 2003.