UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Iterative Learning Control for Spatial Path Tracking

by

Yiyang Chen

A Thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

October 2017

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
Electronics and Computer Science

Doctor of Philosophy

ITERATIVE LEARNING CONTROL FOR SPATIAL PATH TRACKING

by Yiyang Chen

Iterative learning control (ILC) is a high performance method for systems operating in a repetitive manner, which aims to improve tracking performance by learning from previous trial information. In recent years research interest has focused on generalizing the task description in order to achieve greater performance and flexibility. In particular, researchers have addressed the ease of tracking only at a single, or a collection of time instants. However, there still remain substantial open problems, such as the choice of the time instants, the need for system constraint handling, and the ability to release explicit dependence on time. A number of ILC methods have tackled the latter problem, loosely termed spatial ILC, but are all application specific and limited in scope.

The aim of this thesis is to unlock this potential, and the specific contributions are as follows: first a mechanism to optimize the time instants at the critical tracking positions within point-to-point tracking is developed. This is achieved by embedding an additional cost function and deriving a 'Two Stage' design framework to yield an iterative algorithm which minimizes control effort as well as guaranteeing high performance tracking. This approach is based on norm optimal ILC and gradient minimization. Then the task description is further generalized by expanding the formulation to embed via-point constraint and linear planar constraints. This embeds the incorporation of various design objectives including spatial path tracking in a general class of systems, and a mixed form of system constraints are added into this framework. An algorithmic ILC solution is derived using the successive projection method to achieve high performance tracking of the design objectives. Finally, the Two Stage design framework and the generalized ILC framework are combined together to yield the first spatial ILC algorithm capable of optimizing an additional cost function whilst completing the spatial path tracking objective for a general class of systems. All proposed algorithms are verified experimentally on a gantry robot platform, whose experimental results demonstrate their practical efficacy and ability to substantially widen the scope of the current ILC framework.

Contents

List of Figures

List of Tables

Acknowledgements

Thanks to my supervisors Dr Bing Chu and Dr Christopher Freeman for their help and guidance for my work. Then I want to say thanks to my colleagues for their advice on my research. Many thanks to my parents for their support of my study. This thesis is supported by the China Scholarship Council.

Declaration of Authorship

I, Yiyang Chen , declare that the thesis entitled *Iterative Learning Control for Spatial Path Tracking* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: Chen et al. (2015, 2016, 2017a,b,c)

Signed:.....

Date:.....

Nomenclature

$\mathbb{N}$	Set of non-negative integers
$\mathbb{R}^n$	Set of n dimensional real vectors
$\mathbb{R}^{n\times m}$	Set of $n \times m$ real matrices
$\mathbb{S}^n_{++}$	Set of all $n \times n$ real positive definite matrices
$L_2^\ell[0,\ T]$	Space of functions defined on $[0,\ T]$ whose function value belongs to $\mathbb{R}^\ell$ and 2 power is Lebesgue integrable
$l_2^\ell[0,\ N]$	Space of $\mathbb{R}^\ell$ valued Lebesgue square-summable sequences defined on an interval $[0,\ N]$
$\langle x,\ y\rangle$	Inner product of x and y in some Hilbert space
$\mathbb{X} \times \mathbb{Y}$	Cartesian product of two spaces $\mathbb{X}$ and $\mathbb{Y}$
$P_\Theta(x)$	Projection x to the set Θ in some Hilbert space
s	Laplace transform variable
z	z-transform variable
k	Trial number
j	Loop number
i	Tracking point number
u	Input signal
y	Output signal
e	Error signal
t	Time
$\|\cdot\|$	Euclidean norm
$\|\cdot\|_\infty$	Infinity norm

Chapter 1

Introduction

In many industrial applications, the output of a system, y, is required to follow a desired reference trajectory, r. The system may involve moving through a given path or keeping the temperature of some chemical process constant. To achieve this requirement, a control action, u, must be applied, typically achieved via feedback control. However, feedback control cannot in general achieve perfect tracking performance.

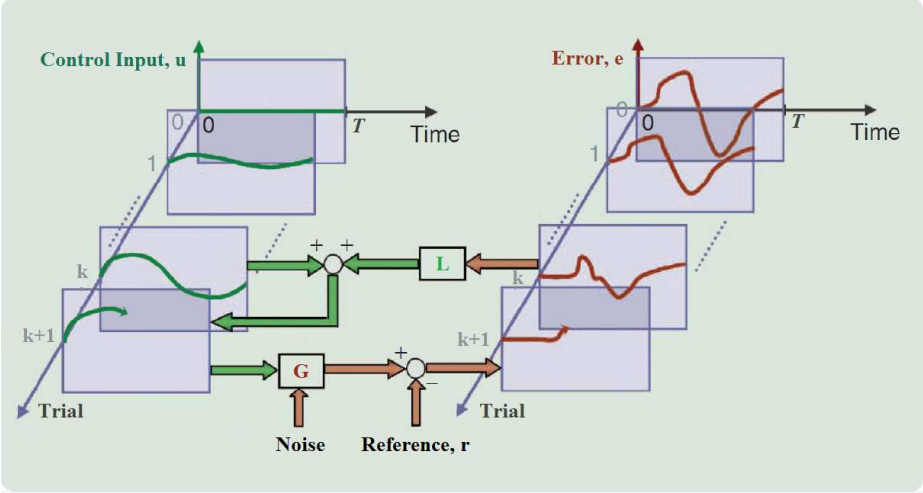


Figure 1.1: An Example of the ILC Update Procedure (Bristow et al., 2006).

To improve tracking accuracy, the field of ILC has been developed and is applicable to systems that execute the same finite time horizon task multiple times. An illustrative example of the ILC update procedure is shown in Figure 1.1, where k denotes the trial number and T is the task duration. This illustrates that ILC attempts to reduce the tracking error, e_{k+1}, by updating the input signal, u_{k+1}, using information u_k and e_k from the previous trial (Bristow et al., 2006). Distinct to traditional feedback control, ILC can theoretically enable the tracking error to converge to zero after sufficient historical attempts for a broad class of systems. This has led ILC to find rich application to industrial tasks which require a high level of accuracy, with examples including inkjet

printing (Bolder et al., 2014), robotic manipulator (Jin, 2016), chemical batch processing (Tao et al., 2017), rehabilitation (Freeman, 2017) and addictive manufacturing (Lim et al., 2017). See Bristow et al. (2006); Ahn et al. (2007) for a detailed background of ILC.

A large body of work, e.g. Tayebi and Zaremba (2003); Moore et al. (2005); Oomen et al. (2009); Hladowski et al. (2010); Paszke et al. (2013); Huang et al. (2014), has studied the properties of classical ILC, which assumes the reference profile, r, is specified over the entire finite time horizon, $[0,\ T]$. However, the output signal is not critical at every time instant, t, along the time horizon in many ILC applications, e.g. robotic pick-and-place tasks. Therefore, the task description of classical ILC has recently been extended to frameworks that can handle a wider range of tracking problems. These extended ILC frameworks have been termed terminal ILC, point-to-point ILC and spatial ILC.

Terminal ILC was initially studied in Xu et al. (1999) to address the class of ILC problems, whose tracking performance is only critical at the end of the finite time horizon, $[0,\ T]$. As a further extension of terminal ILC, the concept of point-to-point ILC was proposed in Freeman et al. (2011), whose design objective is to track a number of critical positions at a subset of time instants, t_i. Furthermore, spatial ILC was initially proposed in Moore et al. (2007) to track a path defined in space without any temporary constraints. Due to this relaxation of the tracking requirement, the aforementioned extended ILC frameworks have facilitated significant design freedom.

1.1 Research Motivation

As stated, the aforementioned extended ILC frameworks do not assume a unique reference trajectory, r, defined over the entire time horizon, $[0,\ T]$. This eliminates the unnecessary output tracking requirements, and exploits significant control design flexibility to embed additional performance (Freeman, 2012). In point-to-point ILC, the tracking accuracy at each special time point along the trial duration is the critical information, and is typically embedded in a cost function subsequently optimized over multiple trials. However, these critical tracking time instants are assumed to be known *a priori*, and the extra flexibilities in choosing the tracking time instants (which affect the system performance) have not been explored. This introduces the possibility of selecting the time instants at the critical tracking positions within point-to-point ILC to optimize an additional cost function whilst completing the tracking task.

To clearly illustrate the need to include temporal optimization in ILC tasks, consider the pick-and-place task shown in Figure 1.2 on a gantry robot platform. In this problem, the robotic arm is required to start from the resting position (shown as a green dot) at the beginning of a trial ($t = 0$), move to the 'pick' position (shown as a yellow dot) at the specified time t_1 and then move to the 'place' position (shown as a red dot) at

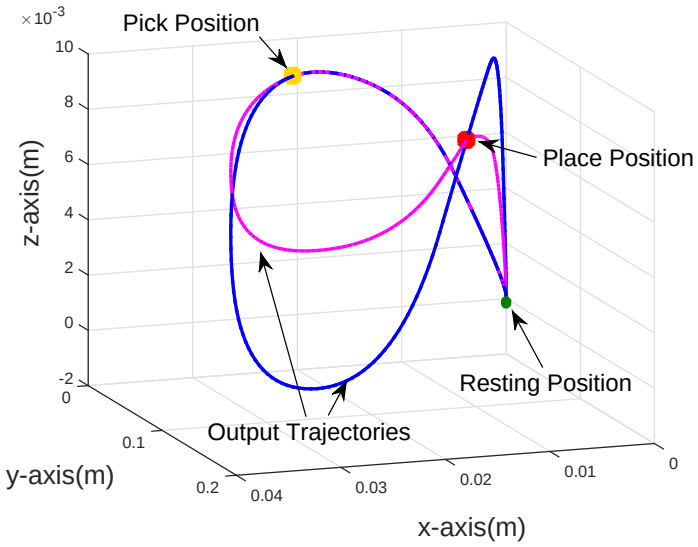


Figure 1.2: A 'Pick-and-Place' Tracking Example on Gantry Robot.

a specified time t_2, before finally moving back (resetting) to the resting position at the end of the trial ($t = T$). Note that in this problem, the pre-specified tracking time allocation, i.e. the 'pick' and 'place' time instants t_1 and t_2, plays a key role in the system performance. Intuitively, if the two tracking time instants are chosen closer to each other, the robotic arm will have to move from the desired 'pick' location to the 'place' position within a very short time, therefore requiring rapid movement and thus incurring high control effort (for example, the difference is more than 30 % of the control effort in the two trajectories shown in the figure).

The spatial ILC problem can be thought of as a generalization of this idea and is difficult to formulate and solve due to the elimination of temporal constraint. Instead, all initial attempts to spatial ILC are application specific, with applications that are limited to a certain class of systems and tracking problems. This fact motivates the work in this thesis to propose a generalized ILC framework which is applicable to various design objectives including spatial path tracking for a general class of systems with a mixed form of system constraints. Furthermore, all initial attempts to spatial ILC focus on purely achieving high performance path tracking, and have not fully harnessed its design freedom to optimize an additional cost function while performing the tracking task, which prevents achieving certain potential practical benefits as well as performing accurate path tracking. This limitation provides a clear motivation for the spatial ILC framework to be further expanded to allow flexibility in the selection of the tracking time instant of each position along the path, with their values also updated by an iterative algorithm to embed further optimization of the overall spatial ILC tracking objective. This is the final topic focused upon in this thesis.

1.2 Contribution and Organization of the Thesis

This thesis addresses the existing limitations within extended ILC frameworks mentioned in the previous section. The contribution and organization of this thesis are given as follows:

Background and Literature Review of ILC (Chapter 2): This chapter describes background information and the current research landscape of ILC. The traditional definition of ILC is first provided with six postulates, and an overview of ILC is given with specific respect to important criteria, e.g. convergence properties and practical implementation. Then, several leading ILC structures are introduced and specific focus is given to optimization based ILC approaches, whose techniques are used later in this thesis. Following this, the constraint handling problem is introduced. Finally, the extended ILC frameworks are described in detail and current limitations are highlighted. These underpin the motivation of the research work in this thesis.

Point-to-Point ILC with Optimal Tracking Time Allocation (Chapter 3 and 4): This chapter formulates the first algorithm capable of solving the high performance point-to-point tracking problem with an additional optimal cost function. A point-to-point ILC framework is proposed based on an abstract operator form in Hilbert space. Using this framework, an optimization problem is formulated to optimize a cost function of interest as well as maintain the ability of precise point-to-point tracking at optimally allocated time instants, t_i. A 'Two Stage' design framework is then developed to solve the non-trivial optimization problem. To exemplify the approach, the control effort is selected to be the target cost function in this thesis, and an iterative implementational algorithm is proposed from the design framework to update both the control action, using norm optimal ILC, and the tracking time allocation, using the gradient projection method. These enable completion of the task with reduction in the control effort compared with *a priori* tracking time allocation. In addition, the input constraint is also considered and a modified algorithm is proposed to give a practical solution to the constrained condition. Note that parts of the above work have been published in Chen et al. (2015, 2017b,c).

Generalized ILC with Application to Spatial Path Tracking (Chapter 5): This chapter formulates a generalized ILC algorithm, which comprises the first solution to various design objectives including spatial path tracking for a general class of systems with mixed forms of system constraints. To solve the spatial path tracking problem, a generalized ILC framework is first formulated to embed intermediate point tracking constraint of the form $F_i y(t_i) = F_i r(t_i)$ at time instant t_i, as well as linear constraints between outputs of the form $P_i y(t) = P_i r(t)$ on defined sub-intervals $[t_{i-1},\ t_i]$, thereby enforcing tracking along lines or planes with no *a priori* timing constraints. This framework also extends the class of ILC task description to incorporate a range of system

constraints with significant relevance to industrial manufacture (e.g. preventing overshoot). Therefore, this framework can be applied to most types of ILC design problems, including spatial ILC, by setting appropriate parameter values. A generalized ILC algorithm is then derived using the successive projection method. Furthermore, rigorous convergence analysis is undertaken on this algorithm to guarantee desirable convergence properties. Note that parts of the above work have been published in Chen et al. (2017a).

Spatial Path Tracking using ILC (Chapter 6): This chapter formulates the first spatial ILC algorithm capable of optimizing an additional cost function whilst completing the spatial path tracking objective for a general class of systems. The general spatial tracking problem is first formulated based on the definition of a spatial path, and then specified to the class of piecewise linear path tracking problems. The optimal tracking time allocation problem within spatial ILC is then formulated using the generalized ILC framework derived in Chapter 5. After that, the Two Stage design framework developed in Chapter 3 is applied to the problem to yield a spatial ILC algorithm based on the generalized ILC update derived in Chapter 5 and the gradient projection method. Note that parts of the above work have been published in Chen et al. (2016).

Practical Verification (Chapter 3, 5, and 6): Furthermore, all the proposed algorithms are evaluated on a three-axis gantry robot test platform to confirm their practical effectiveness. As a major problem arising in practice is due to model uncertainty, the robust performance of these algorithms against model uncertainty is also examined by assuming an inaccurate system model. The experimental results show that the proposed algorithms have a certain level of robustness against model uncertainty, which is an attractive feature in practice. In addition, the experimental results suggest the potential application of the algorithms to the fields outside robotics, which involve a spatial repetitive mode such as stroke rehabilitation.

Chapter 2

Background and Literature Review

This chapter first gives a general description of ILC before providing an overview of 'standard' ILC algorithms, which track a reference defined over the full trial duration, $t \in [0,\ T]$, $T < \infty$. After the general overview, several optimization based ILC algorithms are focused upon in more detail. In addition, recent developments in ILC are described, which mainly focus on the generalization of the traditional ILC tracking requirement.

2.1 General Background to ILC

2.1.1 Definition of ILC

The concept of ILC was first proposed in Uchiyama (1978) by the Japanese scholar M. Uchiyama in 1978. However, as this paper is written in Japanese, it did not receive significant attention outside Japan. The research of ILC is commonly considered to start in 1984 with the publication Arimoto et al. (1984b). This defines ILC as a novel control technique applicable to systems operating in a repetitive manner over a finite time interval, $[0,\ T]$. The system dynamics can be written in the general form

$$y_k = Gu_k \tag{2.1}$$

where u_k and y_k are the control input and measured output on trial k respectively. In the ILC problem, the tracking objective is to sequentially force the system output y_k to track a given reference trajectory, $r \in L_2^m[0,\ T]$, over the finite time horizon, $[0,\ T]$. At the end of the k^{th} trial, ILC uses the input u_k and error $e_k = r - y_k$ to update the input u_{k+1} for the next trial, typically using an algorithm of the form

$$u_{k+1} = f(u_k, e_k) = u_k + Le_k \tag{2.2}$$

where L is a learning operator. There were six postulates of ILC made in Arimoto et al. (1984b) as follows:

- Each trial has an identical trial length, $T > 0$.
- The desired reference trajectory, r, is given *a priori* over the time interval, $[0,\ T]$.
- At the end of each trial, the initial state, $x_k(0)$, for the next trial is reset to a constant value, x_0.
- Invariance of the system dynamics is guaranteed throughout each repeated trial.
- The tracking error, e_k, can be measured and used to update the input, u_{k+1}, for the next trial based on the current input, u_k.
- The system dynamics are invertible, i.e. there exists a causal and stable system that can invert the given system dynamics.

In recent research, the above postulates have gradually been relaxed to provide an ILC tracking task description which achieves more flexibility. In order to understand more about their functions, the general properties of ILC are reviewed in the following subsections.

2.1.2 Convergence of ILC

The general objective of ILC is to force the tracking error, e_k, to converge to zero after sufficient updating trials, and the input, u_k, to converge to a unique input, u^*, at the same time, i.e.

$$\lim_{k\to\infty} e_k = 0, \quad \lim_{k\to\infty} u_k = u^*. \tag{2.3}$$

There are two main types of convergence: asymptotic and monotonic convergence. Asymptotic convergence requires the 2-norm of the error to converge to zero, i.e.

$$\lim_{k\to\infty} \|e_k\| = 0. \tag{2.4}$$

On the other hand, monotonic convergence requires the 2-norm of the error at each trial to be no larger than that of the previous one, and the norm of the error to converge to zero as well, i.e.

$$\|e_{k+1}\| \leqslant \|e_k\|, \quad \lim_{k\to\infty} \|e_k\| = 0 \tag{2.5}$$

Figure 2.1 demonstrates the difference between the two types of convergence. As monotonic error convergence is generally considered of primary importance, it will be considered as the main ILC convergence requirement throughout this thesis.

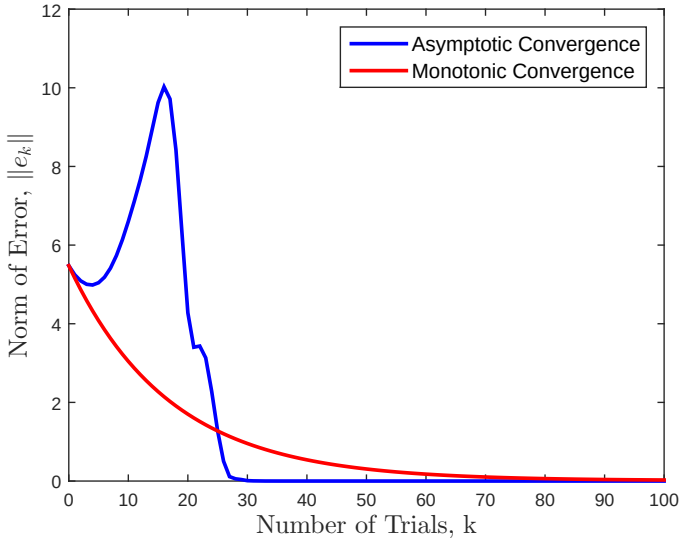


Figure 2.1: Comparison between Asymptotic and Monotonic Convergence

2.1.3 Robustness of ILC

Accurate tracking performance with desired convergence performance has been achieved by many ILC algorithms on nominal systems, and good performance has been achieved in practical applications. However, as the nominal model is only an imperfect representation of the real plant, robustness is a serious issue. If the model uncertainty is large enough, the convergence rate of ILC algorithms becomes slow or the algorithms even cannot provide the desired solution of the original control problem.

In practice it has been found that long term instabilities degrade the performance and convergence of the standard algorithms. This performance issue is at least in part due to model mismatch. Even where this performance issue does not result, convergence to the desired trajectory is affected by model uncertainty. This performance issue of ILC has been an open problem for many years, and a variety of methods (e.g. quantization, filtering, suspension of learning) have been proposed that often lack theoretical basis. Previous robustness results relate to multiplicative and additive uncertainty descriptions (Moon et al., 1998; de Roover and Bosgra, 2000; Tayebi and Zaremba, 2003; Harte et al., 2005; van de Wijdeven and Bosgra, 2007; Donkers et al., 2008; van de Wijdeven et al., 2009; Paszke et al., 2013; Son et al., 2016; Ge et al., 2017). Parametric uncertainties have also been considered (Ahn et al., 2006b). Unstructured uncertainties were addressed for a class of adaptive ILC algorithms (French, 2008). An example of a multiplicative uncertainty is

$$\hat{G} = (1 + \Delta W)G, \quad \|\Delta\|_\infty < 1 \tag{2.6}$$

where W is a fixed stable transfer function and $\|\cdot\|_\infty$ denotes the H-infinity norm. Various robust ILC algorithms have been developed against the model uncertainty problem.

Another issue is considered as initialization error. According to the third postulate in Section 2.1.1, the initial state x_0 of ILC should be reset to constant values at the end of each trial. However, due to practical issues, the initial state x_0 at the beginning of each trial may be slightly different, which affects the convergence and performance of ILC (Sun and Wang, 2003). This has led to development of algorithms which are robust to the initial state error (Jiang and Unbehauen, 1999), or reduce the sensitivity of ILC to initial state error (Chen et al., 1999).

2.1.4 ILC Implementation Structure

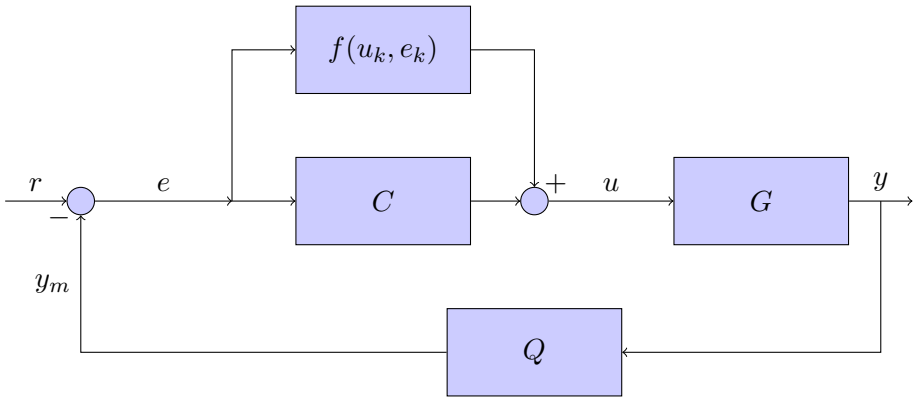


Figure 2.2: ILC Implementation Structure.

ILC is usually applied in combination with other controllers in practice. This is because ILC addresses the repeated disturbance at each trial, but there inevitably exists non-periodic random disturbance in the measured error signal. This motivates employing another control method together with ILC to improve disturbance rejection or base-line tracking.

Furthermore, while performing frequency domain analysis of the error signal, it is noted that ILC instabilities typically manifest at high frequencies, due to model uncertainty in this range. Different filters has been designed to improve the tracking performance such as low pass filters, band pass filters, zero pass filters (Ratcliffe, 2005) and Kalman filters (Ahn et al., 2006a). The block diagram in Figure 2.2 provides an example of practical ILC implementation structures, where C and Q denote the controller and filter respectively.

2.2 Leading ILC Structures

In this section, several leading ILC structures are introduced with their own specific features, and the corresponding ILC update algorithms are also provided. These leading ILC structures have been selected to comprise ILC designs that are supported by

substantial experimental benchmarking.

2.2.1 Simple Structure ILC

Simple structure ILC does not require any information about the system dynamics, and the first simple ILC algorithm is often considered as the proportional-type ('P-type') ILC update law given by

$$u_{k+1} = u_k + \gamma e_k \tag{2.7}$$

where the learning gain γ is a constant scalar that influences the rate of convergence. The authors considered linear time invariant, continuous time state space systems, $S(A, B, C)$. In order to ensure the ILC tracking objective (2.3), the convergence condition

$$\|I - \gamma CB\| < 1 \tag{2.8}$$

is given in Arimoto et al. (1985). It should be noted that the convergence condition does not contain A and is hence independent of the system dynamics. In addition, the value of $CB \neq 0$, otherwise the condition 2.8 does not hold and the system loses control at the beginning.

The next simple ILC algorithm is called derivative-type ('D-type') ILC, and its update law is given by

$$u_{k+1} = u_k + \alpha \dot{e_k} \tag{2.9}$$

where $\dot{e_k}$ denotes the derivative of the error e_k. It is similar to the P-type ILC algorithm, but uses the rate of change of the error to update the input of the next trial instead of the error itself. In order to ensure the ILC tracking objective (2.3), the convergence condition

$$\|I - \alpha CB\| < 1 \tag{2.10}$$

is derived in Arimoto et al. (1985) and Arimoto et al. (1984a). It should be noted that the convergence condition is independent of matrix A even if the system is unstable because of the finite time interval of each trial (Ahn et al., 2007). Due to the same reason, it follows that $CB \neq 0$. However, the D-type ILC algorithm is sensitive to measurement error and process disturbance, and it is seldom used in practical cases.

In order to increase the convergence rate and applicable system classs, P-type and D-Type ILC can be combined together. The resulting 'PD-type' ILC update law is given by

$$u_{k+1} = u_k + \gamma e_k + \alpha \dot{e_k} \tag{2.11}$$

The convergence condition is similar to that of the D-type ILC algorithm (Chen and Moore, 2002). Due to the noise sensitive D-type component, it also has limited practical use.

Inspired by the analogy with traditional proportional plus integral plus derivative control 'PID', the above algorithm has been generalized to include an integral term (Moore, 1993), i.e.

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t) + \alpha \dot{e_k}(t) + \beta \int_0^t e_k(s)\mathrm{d}s. \tag{2.12}$$

Due to the same noise sensitivity problem, PID-type ILC has similar limitations in practical application.

2.2.2 Phase-lead ILC

Phase-lead ILC has a similar form to P-type ILC, and its update law is given by

$$u_{k+1}(t) = u_k(t) + \beta e_k(t + \lambda) \tag{2.13}$$

where $\lambda > 0$ is the phase-lead of the error signal. The idea behind the phase-lead ILC method is to first estimate the phase-lead λ in the control system, and then shift the error signal in order to compensate for the phase-lead and hence achieve satisfactory performance. More information concerning phase-lead ILC, such as the convergence properties, is given in Freeman et al. (2005); Park et al. (1998).

The principle advantage of phase-lead ILC is that it can be considered model free and does not requires a full model of the plant, since the phase-lead, λ, and the gain, β, can be regarded as tuning parameters (Freeman et al., 2005; Cai et al., 2008). However unless the system is a pure delay, phase-lead ILC must always be used in combination with a low-pass filter. Furthermore, the phase-lead ILC update law is equivalent to the P-type ILC update law at the particular condition $\lambda = 0$.

2.2.3 ILC based on 2D Systems Theory

The previously mentioned ILC structures are motivated by achieving trial-to-trial error convergence, but do not fully consider stability in the trial direction. ILC can be considered as a 2D system whose information propagating directions are both along the trial and from trial-to-trial. Therefore, the ILC problem setup can be transferred into a 2D system framework, and then 2D systems theory can be used to develop ILC update algorithms which not only satisfy trial-to-trial error convergence, but also achieve stability in the trial direction. Research in Roesser (1975) formulates a discrete time state

space model

$$\begin{bmatrix} x^h(i+1,j) \\ x^v(i,j+1) \end{bmatrix} = A \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} + Bu(i,j)$$
$$y(i,j) = C \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} \tag{2.14}$$

for the system dynamics over the positive quadrant of the 2D plane, where x^h and x^v are the horizontal and vertical state components, and i, and j are the horizontal and vertical coordinates. An alternative state space representation to (2.14) is given by

$$x(i+1,j+1) = A_1 x(i,j+1) + A_2 x(i+1,j) + A_0 x(i,j) + Bu(i,j)$$
$$y(i,j) = Cx(i,j) \tag{2.15}$$

in Fornasini and Marchesini (1978). One advantage of (2.15) over (2.14) is that the state vector is not split into horizontal and vertical sub-vectors. The information propagation in both directions has a infinite duration in (2.15) and (2.14), but the ILC information propagation only takes a finite number of trials and has a fixed trial length. Furthermore, even if the plant is unstable, a closed feedback loop can be added to the system plant to make it stable and update (2.14) and (2.15) with new closed loop system dynamics.

A number of works have studied the 2D system approach of ILC. The work in Hladowski et al. (2010) considered the stability theory for linear repetitive process, and used used the tools of linear matrix inequalities to design a robust ILC algorithm, which provided desired performance at trial-to-trial as well as along the trial. Subsequent work in Hladowski et al. (2011) used a 2D system setting and applied ILC to the deterministic discrete linear systems with uniform rank greater than unity. Recent work in Paszkea et al. (2016) used the generalized KYP lemma to transform frequency domain inequities to linear matrix inequalities, and designed a stabilized feedback controller to ensure the performance along the trial for system with relative degree larger than unity. In Bolder and Oomen (2016), study was made into the class of systems whose performance variables are distinguished from measured variables. In this setting, previous ILC approaches cannot be directly implemented, and hence an ILC approach was derived in this paper to handle this problem, whose stability is analyzed in a 2D framework.

2.2.4 High Order ILC

In some complex practical applications, it has been suggested that information from more than one previous trial should be employed in the update to improve the system

performance. The resultant update law has the general form

$$u_{k+1} = \sum_{i=1}^{M} \alpha_{k+1}(i) u_{k+1-i} + \sum_{i=1}^{M} \beta_{k+1}(i) e_{k+1-i} \tag{2.16}$$

where $\alpha_{k+1} = [\alpha^1_{k+1}, \alpha^2_{k+1}, \ldots, \alpha^M_{k+1}]$, $\beta_{k+1} = [\beta^1_{k+1}, \beta^2_{k+1}, \ldots, \beta^M_{k+1}]$, and M is the number of previous trials. The parameters may be calculated by minimizing the cost function

$$J_{k+1} = \|e_{k+1}\|^2 + \alpha^\top_{k+1} W_1 \alpha_{k+1} + \beta^\top_{k+1} W_2 \beta_{k+1} \tag{2.17}$$

where $W_1, W_2 \in \mathbb{R}^{M \times M}$. The proof of the convergence properties is shown in Owens and Feng (2003); Owens and Hatonen (2005). As high order ILC uses information from M past trials, it is considered to provide a better output tracking performance than those approaches which only learn from the last trial (Bien and Huh, 1989).

2.3 Optimization Based ILC Approaches

In this section, ILC algorithms that use system model information to update their input are considered. The control system model that is generally considered is an ℓ-input, m-output linear time-invariant system, which leads to the continuous time description given in state space form by $S(A, B, C)$, i.e.

$$\begin{aligned} \dot{x}_k(t) &= A x_k(t) + B u_k(t), \\ y_k(t) &= C x_k(t) \end{aligned} \tag{2.18}$$

where $t \in [0,\ T]$ is the time, $x_k(t) \in \mathbb{R}^n$, $u_k(t) \in \mathbb{R}^\ell$ and $y_k(t) \in \mathbb{R}^m$ are the state, input and output respectively; A, B and C are system matrices of compatible dimensions; the subscript $k \in \mathbb{N}$ again denotes the trial number. The system can be represented in equivalent operator form

$$y_k = G u_k + d \tag{2.19}$$

The convolution operator G and signal d take the form

$$(Gu)(t) = \int_0^t C e^{A(t-s)} B u(s) \mathrm{d}s,\ d(t) = C e^{At} x_0 \tag{2.20}$$

where, without loss of generality, $d(t)$ may be absorbed into the reference to give $x_0 = 0$, $d(t) = 0$. The general ILC update law is defined by

$$u_{k+1} = u_k + L e_k \tag{2.21}$$

where $L : L_2^m[0,\ T] \to L_2^\ell[0,\ T]$ is the learning operator. Thus, from (2.19) and (2.21) the error evolution is

$$e_{k+1} = (I - GL)e_k \tag{2.22}$$

where $(I - GL)$ is the error transition operator mapping e_k to e_{k+1}. A number of convergence conditions have been proposed, and the most general one among them is the spectral radius condition

$$\rho(I - GL) < 1 \tag{2.23}$$

where the spectral radius $\rho(\cdot)$ denotes the largest absolute value of the eigenvalues of a square matrix or a bounded linear operator. The condition (2.23) can be implied by a simpler contraction mapping condition

$$\|I - GL\| < 1. \tag{2.24}$$

which leads to

$$\|e_{k+1}\| = \|(I - GL)e_k\| \leqslant \|(I - GL)\|\,\|e_k\| < \|e_k\|\,. \tag{2.25}$$

Specific forms of L are now reviewed.

2.3.1 Inverse ILC

Inspection of the condition (2.24) clearly motivates choosing L as the inverse of G, so that $e_{k+1} = 0,\ \forall u > 0$. The continuous time version of inverse ILC update law is modified from Harte (2007) as

$$u_{k+1} = u_k + \beta G^{-1} e_k \tag{2.26}$$

where the scalar β is a learning gain which influences the convergence properties. The error update equation directly follows as

$$e_{k+1} = (I - \beta I)e_k \tag{2.27}$$

by substituting $L = \beta G^{-1}$ into (2.22). Applying the general convergence condition of (2.24) yields

$$|1 - \beta| < 1. \tag{2.28}$$

It is clear that $0 < \beta < 2$ ensures the convergence of the error. Note that the error will converge to zero in one trial when $\beta = 1$ since

$$e_1 = r - y_1 = r - G_e u_1 = r - G(u_0 - G^{-1}e_0) = 0. \tag{2.29}$$

Inverse ILC can be considered as a theoretically ideal ILC algorithm. However, if the control system is non-minimum phase, its inverse operator is unstable and cannot be applied within this algorithm (Owens and Chu, 2014). Moreover, the exact inverse system model always has a degree of model uncertainty in practice. See Harte et al. (2005) for the related robustness analysis. In practice, a Q-filter can be included in the design to remove the high frequency disturbance, and the value of β should be chosen small enough to make inverse ILC become robust to model uncertainty and random disturbance.

2.3.2 Gradient Descent ILC

An alternative method of guaranteeing (2.24) is to use the system adjoint operator G^*. This leads to $L = \beta G_e^*$, which yields the update law

$$u_{k+1} = u_k + \beta G^* e_k \tag{2.30}$$

where the scalar $\beta > 0$ is a learning gain. Note that the update (2.30) is also the solution of the problem

$$\min \|e_{k+1}\|^2 \tag{2.31}$$

using the gradient minimization method. Substituting $L = \beta G^*$ into the general convergence conditions (2.24) yields

$$\|I - \beta G G^*\| < 1. \tag{2.32}$$

Following the idea of the discrete time version in Owens et al. (2008), since GG^* is positive definite, the convergence condition becomes

$$0 < \beta < \frac{2}{\|GG^*\|} = \frac{2}{\|G\|^2} \tag{2.33}$$

Algorithms of the form (2.30) have been considered in Freeman et al. (2007).

2.3.3 Norm Optimal ILC

Norm optimal ILC can be regarded as a generalization of inverse ILC and gradient ILC. The idea of norm optimal ILC is to calculate input signal u_{k+1} to minimize the cost function

$$J_{k+1} = \|r - Gu\|_S^2 + \|u - u_k\|_R^2 \tag{2.34}$$

at the end of each trial where S and R are weighting matrices (Amann et al., 1996a). The minimum solution, u_{k+1}, can be derived by applying partial differentiation to J_{k+1} with respect to u. This involves the stationary point, $\partial J_{k+1}/\partial u$ being set to zero, leading

to the norm optimal ILC update law

$$u_{k+1} = u_k + G^* e_{k+1} \tag{2.35}$$

where G^* is the adjoint operator of the system G. The corresponding error update equation is

$$e_{k+1} = (I + GG^*)^{-1} e_k \tag{2.36}$$

It can be shown that the convergence condition (2.24) is satisfied as

$$\|e_{k+1}\| \leqslant \frac{1}{1 + \underline{\sigma}^2(GG^*)} \|e_k\| \leqslant \|e_k\| \tag{2.37}$$

where $\underline{\sigma}^2(GG^*)$ is the smallest spectral radius value of symmetric, positive definite operator GG^*. As (2.35) is non-causal, it can be manipulated to produce the causal feedforward update law given by:

$$u_{k+1} = u_k + G^*(I + GG^*)^{-1} e_k \tag{2.38}$$

Alternatively, a feedback and feedforward implementation can be used. The causal implementation

$$u_{k+1}(t) = u_k(t) + R^{-1}B^\top[-K(t)(x_{k+1}(t) - x_k(t)) + \xi_{k+1}(t)] \tag{2.39}$$

was introduced in Amann et al. (1996a); Amann (1996). Here $K(t)$ denotes the Riccati feedback matrix computed via

$$0 = \dot{K}(t) + (A^\top - K(t)BR^{-1}B^\top)K(t) + K(t)A + C^\top SC, \;\; K(T) = 0, \tag{2.40}$$

and $\xi_{k+1}(t)$ denotes the predictive feedforward term at the $(k+1)^{th}$ trial, given by

$$0 = \dot{\xi}_{k+1}(t) + (A^\top - K(t)BR^{-1}B^\top)\xi_{k+1}(t) + C^\top S e_k(t), \;\; \xi_{k+1}(T) = 0. \tag{2.41}$$

In practice, the state x can be obtained either from a model run in parallel of the system or the estimation of an observer using the real time measured data.

2.3.4 Parameter Optimal ILC

To address the fact that norm optimal ILC requires full knowledge of the system model $S(A, B, C)$ during the feedback and feedforward update computation procedure, a parameter optimal ILC algorithm was developed to achieve monotonic convergence by employing a simple form to result in a simplified control law. The corresponding ILC update law is

$$u_{k+1} = u_k + \gamma_{k+1} e_k \tag{2.42}$$

where the scalar γ_{k+1} is selected to optimize the cost function

$$J_{k+1} = \|e_{k+1}\|^2 + \omega\gamma_{k+1}^2 \tag{2.43}$$

in which the scalar $\omega > 0$. To yield the stationary condition, the partial derivatives of the cost function with respect to γ_k, i.e. $\partial J_{k+1}/\partial\gamma_{k+1}$, is set to zero, which corresponds to

$$\gamma_{k+1} = \frac{\langle e_k, Ge_k\rangle}{\omega + \|Ge_k\|^2}. \tag{2.44}$$

The above solution results in the tracking error update equation

$$e_{k+1} = (I - \gamma_{k+1}G)e_k \tag{2.45}$$

where $\langle\cdot,\cdot\rangle$ is the inner product and $\|\cdot\|$ is the induced norm. The associated convergence properties are derived in Owens and Feng (2003). If $(G + G^\top)$ is positive definite, the error converges to zero as k tends to infinity. Otherwise, the final error converges to some non-zero value. This framework can be applied to inverse, gradient descent and norm optimal ILC to improve their convergence performance.

2.3.5 L-Q Optimal ILC

An alternative generalization of form (2.21) has also been considered. L-Q optimal ILC has an update law form

$$u_{k+1} = Q_{opt}(u_k + L_{opt}e_k) \tag{2.46}$$

where Q_{opt} is a filter and L_{opt} is a learning operator. This algorithm minimizes the cost function

$$J_k = \|r - Gu\|_S^2 + \|u\|_{R_{LQ}}^2 + \|u - u_k\|_R^2 \tag{2.47}$$

at each trial where $R_{LQ} \in \mathbb{S}_{++}^{\ell}$, R and S are weighting matrices (Norrlof, 2000). Similarly, the optimal solution u_{k+1} is computed by setting the partial differentiation of J_k with respect to u_{k+1} to zero. The solutions Q_{opt} and L_{opt} are modified from Bristow et al. (2006) as

$$Q_{opt} = (G^*G + R_{LQ}R^{-1} + I)^{-1}(G^*G + I),\ L_{opt} = (G^*G + I)^{-1}G^*. \tag{2.48}$$

It should be noted that the values of S and R affect both the convergence rate and the final error norm. When k tends to infinity, the final error is

$$e_\infty = [I - G(G^*G + I)^{-1}G^*]r. \tag{2.49}$$

It is clear that if $R_{LQ} = 0$ the error will converge to zero, and the update (2.46) collapses to norm optimal ILC update in Section 2.3.3. If $R_{LQ} \neq 0$ this algorithm can be used to prevent actuator saturation, and it can also be applied to decrease the ILC sensitivity at

high frequencies in order to reduce possible equipment damage (Gunnarsson and Norrlof, 2001). Note the (2.47) is a generalization of (2.34), however the implementation (2.46) is feedforward rather than the combined feedforward and feedback structure of (2.39).

2.3.6 Newton Method Based ILC

This approach addresses a general class of non-linear systems, and was motivated by well-established non-linear optimization methods. The Newton approach can be written as

$$x_{k+1} = x_k - \frac{f(x_k)}{f^{'}(x_k)} \tag{2.50}$$

where $f^{'}(x_k)$ is the derivative of the function $f(x_k)$ with respect to vector x_k. The Newton method aims at solving $f(x_k) = 0$. It should be noted that this method is only valid if $f^{'}(x_k)^{-1}$ exists. See Ortega and Rheinboldt (2000) for more details concerning the Newton method.

The Newton method is based upon performing a local linearization of the dynamics, followed by a linear ILC update over the tracking error. A discrete non-linear state space model is given in Lin et al. (2006), in the form

$$\begin{aligned} x(t+1) &= f(x(t), u(t)) \\ y(t) &= h(x(t)) \end{aligned} \tag{2.51}$$

with initial state $x(0) = x_0$. From (2.51) the algebraic relationship between input and output over the k^{th} trial over the finite horizon $t \in [0,\ T]$ can be expressed as

$$\begin{aligned} y_k(0) &= h(x_k(0), u_k(0)) = g_0(x_k(0), u_k(0)) \\ y_k(1) &= h(x_k(1), u_k(1)) = h(f(x_k(0), u_k(0)), u_k(1)) \\ &= g_1(x_k(0), u_k(0), u_k(1)) \\ &\vdots \\ y_k(T) &= h(x_k(T), u_k(T)) \\ &= h(f(x_k(T-1), u_k(T-1)), u_k(T)) \\ &= g_N(x_k(0), u_k(0), u_k(1), \ldots, u_k(T)). \end{aligned} \tag{2.52}$$

Thus, the non-linear system (2.51) can be expressed by the algebraic function $g(\cdot) : l_2^{\ell}[0,\ T] \mapsto l_2^{m}[0,\ T]$ given by

$$y_k = g(u_k),\ g(\cdot) = [g_0(\cdot), g_1(\cdot), \ldots, g_T(\cdot)]^{\top}. \tag{2.53}$$

The error at the k^{th} trial is hence $e_k = r - g(u_k)$. Newton method based ILC can then be applied to solve the ILC problem by setting $f(x_k)$ to $r - g(u_k)$, which corresponds to $r - g(u) = 0$. Application in (2.50) leads to the Newton method based ILC update law

$$u_{k+1} = u_k - g^{'}(u_k)^{-1} e_k. \tag{2.54}$$

See Lin et al. (2006) for a description of the convergence properties. While Newton based method based ILC enables application to non-linear control systems, it should be noted that Newton based method ILC provides the same update law as the inverse ILC algorithm when applied to a linear system.

The term $g^{'}(u_k)$ is simply a linearization of the system about input u_k. Hence the form (2.54) can be replaced by the general form $u_{k+1} = u_k + Ke_k$ with K any learning-gain operator which ensures convergence of the linearized plant model $g^{'}(u_k)$. An example is gradient ILC, where $K = \beta(g^{'}(u_k))^{\top}$. Robust convergence properties then follow by applying the analysis in Ortega and Rheinboldt (2000).

2.4 System Constraints of ILC

In practice, system constraints have significant relevance to industrial manufacture due to physical limitations or performance requirements. Various constrained ILC algorithms have been derived in the literature to solve the constrained tracking problem with a high level of accuracy. In this section, constrained ILC algorithms with three typical system constraints are reviewed.

2.4.1 Input Constraint

In practice the input constraint typically represents saturation of the system's control action. Therefore, the input u provided by the corresponding ILC update must belong to the input constraint set Ω. Due to different practical requirement, the input constraint set Ω may have different forms and typically assumes one of the following forms using the notation of a signal space:

- *Input saturation constraint:*

$$\Omega = \{u \in L_2^{\ell}[0,\ T] : |u(t)| \preceq M(t),\ t \in [0,\ T]\}, \tag{2.55}$$

- *Input amplitude constraint:*

$$\Omega = \{u \in L_2^{\ell}[0,\ T] : \lambda(t) \preceq u(t) \preceq \mu(t),\ t \in [0,\ T]\}, \tag{2.56}$$

- *Input sign constraint:*

$$\Omega = \{u \in L_2^{\ell}[0,\ T] : 0 \preceq u(t),\ t \in [0,\ T]\}, \tag{2.57}$$

- *Input energy constraint:*

$$\Omega = \{u \in L_2^{\ell}[0,\ T] : \int_0^T u^{\top}(t)u(t)\mathrm{d}t \leqslant M\} \tag{2.58}$$

where the sign $\preceq$ means 'precedes or equals'. The input constraint handling problem within ILC has been studied in a large body of work, and Various constrained ILC algorithms have been formulated to solve this problem using different techniques. An ILC approach was established in Xu et al. (2005) and proved to have the ability to compensate for input deadzone through control repetitions. The work in Chu and Owens (2010); Chu et al. (2015) used the successive projection method to handle input constrained problems within classical ILC and point-to-point ILC respectively. In Mishra et al. (2011), convex quadratic programing was applied to develop optimization-based ILC schemes for input constrained linear systems. The barrier method was used in Freeman and Tan (2013) to solve input constrained problems within point-to-point ILC. Norm optimal optimization techniques were used in Janssens et al. (2013b) to solve input constrained problems for linear time invariant systems, and further generalized in Volckaert et al. (2013) to solve input constrained problems for nonlinear systems.

2.4.2 Output Constraint

Similarly to the previous case, the output constraint typically represents the boundary of the acceptable moving space needed to prevent the potential overshoot problem (e.g. causing damage to the device and reducing the product quality). In the same way, the measured output y from the system must belong to the output constraint set, Φ. Due to different practical conditions, the output constraint set, Φ, may also have different forms and typically assumes one of the following forms using the notation of a signal space:

- *Output saturation constraint*

$$\Phi = \{y \in L_2^{m}[0,\ T] : |y(t)| \preceq N(t),\ t \in [0,\ T]\}, \tag{2.59}$$

- *Output polyhedral constraint*

$$\Phi = \{y \in L_2^{m}[0,\ T] : a_i^{\top} y(t) \leqslant b_i,\ a_i \in \mathbb{R}^m,\ b_i \in \mathbb{R},\ i = 1, \ldots, M,\ t \in [0,\ T]\}. \tag{2.60}$$

Unlike the input constraint handling problem, currently little research has been carried out in the field of the output constraint handling problem within ILC. One typical example of the research in this field is given in Jin and Xu (2013), which uses a Barrier Lyapunov function to handle the constraints with non-parametric uncertainties satisfying the Lipchitz condition or norm boundedness.

2.4.3 State Constraint

Another significant case to be considered is the state constraint representing the limited range of the environment change within the ILC tracking task, such as the working temperature of the system. Similar to the previous two system constraints, the state variable x of the system must belong to the state constraint set Υ. Due to the different practical environmental requirements, the state constraint set, Υ may also have different forms and typically assumes one of the following types

- *State saturation constraint*

$$\Upsilon = \{x \in L_2^n[0,\ T] : |x(t)| \preceq L(t),\ t \in [0,\ T]\}, \tag{2.61}$$

- *State amplitude constraint*

$$\Upsilon = \{x \in L_2^n[0,\ T] : \lambda(t) \preceq u(t) \preceq \mu(t),\ t \in [0,\ T]]\}. \tag{2.62}$$

There is scant existing research focusing on the state constrained ILC problem. One example is given in Xu and Jin (2013), which uses a Barrier Lyapunov function to formulate an ILC scheme for a state constrained non-linear MIMO system with parametric and non-parametric uncertainties satisfying the Lipchitz condition or norm boundedness. Further note that all previous research into constrained ILC only focuses on a specific class of ILC problem with a single type of constraint, e.g. either input, output or state constraints, and there does not exist any attempts to handle a mixed form of constraints for a general class of systems.

2.5 Generalized ILC Task Description

In the previous sections, a wide class of ILC algorithms obeying all six postulates in Section 2.1.1 have been reviewed. However, recent ILC work removes some of these postulates to generalize the ILC tracking task description. For example, a varying trial length T was studied in Li et al. (2014); Shen et al. (2016) and a random initial condition was studied in Chi et al. (2008), which remove the first and the third postulates respectively. Further note that the second postulate significantly influences the range of tasks

to which ILC can be applied and hence is of high practical importance. Therefore, the second postulate is next considered and extensions concerning a non-specified reference trajectory, r, are reviewed.

2.5.1 Terminal ILC

In traditional ILC, the control system is required to track every point in the reference trajectory. However, tracking performance is only required at the end of the finite time horizon, $[0,\ T]$ in some particular practical applications, such as the station stop control of a train (Y.Wang and Hou, 2011) and the temperature control of a thermoforming oven (Gauthier and Boulet, 2008). In these applications, only the final conditions are of interest, e.g. the stop position and final temperature, and how the output attains the final condition is not considered.

Terminal ILC, which only specifies the reference trajectory tracking at the final time instant T, has been formulated as an extension of classical ILC to solve this kind of ILC problem. It was studied in Gauthier and Boulet (2008); Y.Wang and Hou (2011); Xu et al. (1999); Xu and Huang (2008a). Compared to the classical ILC framework, terminal ILC has more design freedom as it removes the unnecessary tracking requirements at these not-critical time instants. However, previous research into terminal ILC has not fully exploited the resulting design freedom to tackle an additional cost function of practical interest whilst performing the terminal point tracking. Furthermore, papers on terminal ILC consider only a single point-to-point movement, rather than a sequence of actions needed to build up complex movements required in robotic automation and production line assembly.

2.5.2 Point-to-Point ILC

In a large class of practical systems, such as industrial machines (Freeman et al., 2011), robotic pick and place tasks (Freeman, 2012), UAV surveillance (Barton and Kingston, 2013) and stroke rehabilitation (Freeman et al., 2009), it is required that the output achieves perfect tracking at more than one defined time instants $t = t_i$ respectively. Therefore, point-to-point ILC was developed in Freeman et al. (2011) as an extension of terminal ILC to solve problems which only require tracking of a number of critical positions at a subset of time instants, i.e.

$$\Lambda = [t_1, t_2, \ldots, t_M]^\top,\ 0 < t_1 < t_2 < \ldots < t_M \leqslant T \tag{2.63}$$

along the time duration without the performance of the output signal being specified between these time instants. An illustrative example is shown in Figure 2.3, in which the point-to-point output trajectory only tracks the critical positions along the reference

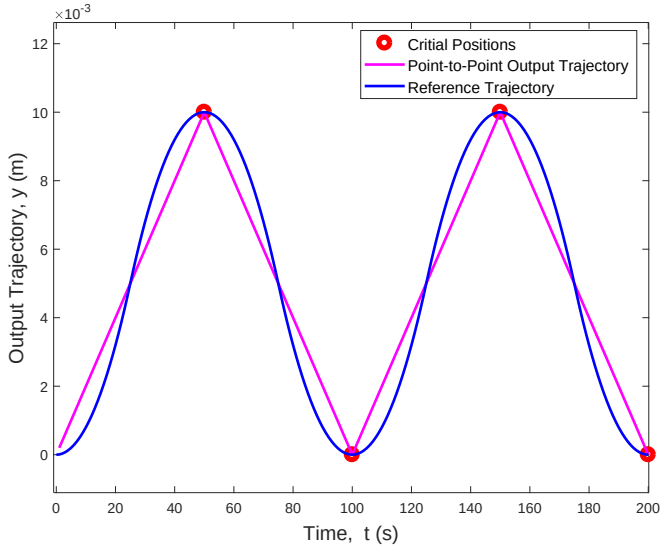


Figure 2.3: Point-to-point ILC

trajectory. For a similar reason to terminal ILC, point-to-point ILC leverages significant control design flexibility by eliminating the unnecessary output constraints. The control design freedom was exploited in Freeman (2012); Son et al. (2013) to address the optimization of an additional cost function as well as tracking accuracy. Practical performance of point-to-point ILC has been illustrated in a large number of applications, e.g. high-acceleration positioning tables (Ding and Wu, 2007), robotic manipulators (Park et al., 2006), two-mass systems (van de Wijdeven and Bosgra, 2008), electro-mechanical systems (Freeman and Dinh, 2015) and human motor systems (Zhou et al., 2017).

The point-to-point system can be represented in equivalent operator from

$$y_k^p = G_\Lambda^p u_k \tag{2.64}$$

where y_k^p denotes the point-to-point output, i.e.

$$y_k^p = [y_k(t_1), y_k(t_2), \ldots, y_k(t_M)]^\top \tag{2.65}$$

and G_Λ^p denotes the point-to-point system operator. The general point-to-point ILC update law is given by

$$u_{k+1} = u_k + L_p e_k^p \tag{2.66}$$

where L_p is the point-to-point learning operator, and e_k^p is the point-to-point error at the corresponding time instants t_i, i.e.

$$e_k^p = [e_k(t_1), e_k(t_2), \ldots, e_k(t_M)]^\top, \ e_k(t_i) = r(t_i) - y_k(t_i), \ i = 1, \ldots, M. \tag{2.67}$$

Note that in principle several of the approaches in Section 2.3 can be adapted to solve the point-to-point ILC problem. The corresponding point-to-point error transition operator

is $(I - G_{\Lambda}^{p} L_p)$. Similar to classical ILC, the gradient descent point-to-point ILC update law is given in Freeman and Tan (2013), and this paper also embedded input constraints used barrier method, however, this required substantial trials to converge. In addition, the norm optimal point-to-point ILC update law was given in Owens et al. (2013) with corresponding convergence conditions.

General point-to-point ILC design by employing a 'complete' reference that passes through all the desired intermediate points was studied in Ding and Wu (2007); Park et al. (2006); van de Wijdeven and Bosgra (2008). These methods, however, do not fully exploit the extra freedom provided by the point-to-point tracking requirements. As such, the overall system performance could be limited. This drawback was addressed recently in Owens et al. (2013); Janssens et al. (2013b); Shen and Wang (2014); Son et al. (2013) where the intermediate point tracking requirements are directly handled by optimizing a quadratic performance index characterizing the tracking performance at these intermediate points. Results containing convergence properties of these algorithms are also available. More recently, system constraints in point-to-point ILC have been considered Freeman (2012); Freeman and Tan (2013); Chu et al. (2015).

A reference trajectory update-based point-to-point ILC algorithm was proposed in Son et al. (2013) to solve the point-to-point problem for discrete time systems, which not only updates the input signal, but also updates the reference signal at each trial. It fixes the reference at those required tracking time points, but the remaining points can be varied. This method is able to provide faster convergence behavior than traditional ILC methods. Application of this approach ensures the reference signal stays constant at the point-to-point time points, and the convergence condition for this method is derived in this paper. A direct tracking control-based point-to-point ILC algorithm was proposed in Son et al. (2011) based on the L-Q approach, but is able to track the required reference points from trial to trial directly without a full reference signal. This method can also provide a faster convergence rate than traditional ILC methods. The convergence properties of this method are also provided in this paper. In Lim and Barton (2014), a pareto optimization based method was considered to solve the point-to-point problem, which includes no less than two conflicting objectives within a single framework. In this paper, a multi-objective cost function is considered. By minimizing this cost function, an ILC updating algorithm is derived with desirable convergence conditions.

Within all these point-to-point ILC frameworks, the information of the tracking time allocation, Λ, of these critical positions is naturally embedded within some cost function whose optimization is implemented. Hence these cost functions are highly dependent on the tracking time allocation, Λ. If the optimization of Λ within these cost functions can be addressed then significant practical benefit can be realized, such as reducing the energy used, reducing the damage to machine components and increasing the efficiency of production (i.e. throughput). However, all the aforementioned point-to-point ILC problem formulations have assumed that the critical tracking time instants, t_i, are known

a priori, and no one has tried to expand the existing point-to-point ILC frameworks to enable the optimal selection of the tracking time allocation, Λ. Note that existing research has been made in Janssens et al. (2014) to address the optimal tracking time allocation within a series of independent point-to-point robotic motions, but these independent motions are not coupled together and the approach does not take advantage of ILC to enable precise tracking.

2.5.3 Spatial ILC

While generalizing the ILC tracking tasks, the aforementioned ILC frameworks then all specify temporal tracking. However, a class of tracking problems has emerged in which the task comprises following a path defined in space with no *a priori* temporal constraints. By removing the unnecessary temporal tracking constraints, the design freedom of this 'spatial' tracking problem setup can yield significantly better tracking performance in practice. This problem has been studied in Xu and Huang (2008b); Verscheure et al. (2009); Lippa and Boyd (2014), but these approaches did not employ any form of ILC. ILC has been applied to spatial tracking problems by defining a parameter set in space which is then used to update the input signal through some form of spatio-temporal mapping.

A 2D corner tracking problem was considered in Moore et al. (2007). Here the minimum distance to the corner is measured, and used to switch on or off the corresponding motor (operating in a constant velocity mode) in a spatial ILC framework, i.e.

$$u_{k+1}(Pr) = u_k(Pr) + f(\epsilon(Pr_f)) \tag{2.68}$$

where Pr denotes the 'progress', Pr_f denotes the 'future progress' and ϵ denotes the spatial error. This is the first paper to propose a spatial ILC approach and confirmed its advantage in terms of accurate tracking and simple implementation. However, it is limited by its system class, task and lack of global optimality (due to use of a single spatial parameter).

Subsequent research in Sahoo et al. (2007) applied a similar scheme to the torque ripple minimization problem in a switch reluctance motor. A P-type spatial ILC algorithm was proposed, i.e.

$$u_{k+1}(\theta) = u_k(\theta) + \gamma T_k^{err}(\theta) \tag{2.69}$$

where θ denotes the rotor position, γ denotes the learning gain and T_k^{err} denotes the spatial error. This algorithm used the torque error at each rotor position to update the input voltage, which reduced the periodic varying torque ripple at the rotor positions. Although this algorithm provides an elegant and simple implementation, it is restricted to 1D tracking tasks without achieving any additional global objective.

Recently, work in Hoelzle and Barton (2016) used spatial ILC to increase the tracking performance of a specific system class, i.e. additive manufacturing, with spatial steady state output variable coupling. Due to the specific tracking requirement, i.e. steady state tracking variables, they eliminated the unnecessary temporal information by redefining the system parameters in spatial coordinates via 2D convolution reconstruction, and reformulated the following spatial ILC update law:

$$u_{k+1} = L_u(x-m, y-n)u_k(x,y) + L_e(x-m, y-n)e_k(x,y) \tag{2.70}$$

where L_u and L_e are the impulse responses of the two learning filters, $u_k(x,y)$, $e_k(x,y)$ are spatial input, error signals, and x, y are spatial coordinates. Furthermore, they formulated different forms of spatial ILC algorithms, e.g. P-type, model inversion, Q-filter and norm-optimal, in both lifted and frequency domain representations. In a large class of spatial tracking tasks, the temporal information is assumed as a variable rather than specified as *a priori*, and this information indeed plays an significant role in the task specification. However, this ILC framework inherently differs from previous spatial research due to the elimination of temporal information, and its application is restricted to specific tracking tasks without the specification of temporal variables.

From the above literature review, it is clear that all previous implementation of spatial ILC are application specific, and none has attempted to propose a generalized version of spatial ILC which is applicable to a wide range of system classes. In addition, all focus on purely achieving the path tracking, and have not fully harnessed the design freedom of spatial ILC to optimize an additional cost function while following the defined path. Research in Janssens et al. (2013a) attempted to minimize the total time of repeated path tracking task within a 2D plane, but it updated the system model at each trial which caused a change in controller parameters. Furthermore, it could not be generalized to an arbitrary form of cost function and incurred a high computation load.

2.6 Summary

In this chapter, the concept of ILC has been first summarized together with the six postulates made in Arimoto et al. (1984b). Its advantages over traditional control methods and potential problems on both theoretical and practical sides have been reviewed. Then some leading ILC design frameworks were introduced, and particularly optimization based ILC algorithms have been examined and a study of their operation and convergence conditions has been made. In addition, the recent research on the generalization of ILC task description with a non-specified reference trajectory over the finite horizon $[0,\ T]$ has been reviewed. Note that many classical ILC algorithms can be modified to apply to terminal ILC and point-to-point ILC. The advantages of these extended

ILC frameworks over the classical ILC framework has been highlighted in terms of the flexibility within their control design.

From the above literature, it is clear that limitations exist related to the fixed tracking time allocation of the critical positions within point-to-point ILC. These drawbacks motivate the further expansion of the current point-to-point ILC framework to allow extra flexibility of the tracking time allocation, which guarantees an optimal cost function as well as perfect tracking. Also, limitations exist in the field of constrained ILC in terms of generality. These drawbacks motivate work to formulate a generalized ILC framework applicable to various ILC design problems and a general class of systems with a mixed form of system constraints. Limitations have also been found associated with the design freedom exploitation as well as application range of existing spatial ILC frameworks. These drawbacks hence provide strong motivation for the spatial ILC framework to be expanded and generalized to allow extra flexibility in the selection of tracking time allocation of each position along the path within a general class of systems.

Chapter 3

Point-to-Point ILC with Optimal Tracking Time Allocation

In point-to-point ILC, only a subset of critical tracking positions, $\{r_i\}$, are of interest, and the tracking time allocation of the critical positions is generally embedded within a cost function whose optimization is implemented in the ILC framework. Hence this cost function is highly dependent on the tracking time allocation within the point-to-point ILC tracking problem. However, all existing point-to-point ILC problem formulations, including those introduced in Section 2.5.2, have assumed the tracking time allocation is known *a priori*. This hence motivates the expansion of the point-to-point ILC framework to allow flexibility in the tracking time allocation, with its input also updated to achieve the overall point-to-point control objective.

This chapter develops a comprehensive optimal tracking time allocation framework ('Two Stage' design framework) within point-to-point ILC to allow automatic selection of the tracking time instants to optimize some performance objective of interest and, at the same time, achieve high performance tracking at the chosen critical positions. In doing so, significant practical benefit is realized, such as reducing the energy used, reducing the damage to machine components and increasing the efficiency of production (i.e. throughput).

3.1 Formulation of the Problem

In this section, the design problem is formulated rigorously into an optimization problem using an abstract operator form representation of system dynamics in some Hilbert space.

A linear time-invariant continuous time state space model

$$\begin{aligned} \dot{x}_k(t) &= Ax_k(t) + Bu_k(t), \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{3.1}$$

is considered, where $t \in [0,\ T]$ is the time, $x_k(t) \in \mathbb{R}^n$, $u_k(t) \in \mathbb{R}^\ell$ and $y_k(t) \in \mathbb{R}^m$ are the state, input and output respectively; A, B and C are system matrices of compatible dimensions; the subscript $k \in \mathbb{N}$ denotes the trial number. The system can be represented in equivalent operator form

$$y_k = Gu_k + d \tag{3.2}$$

where $G : L_2^\ell[0,\ T] \to L_2^m[0,\ T]$, $y_k,\ d \in L_2^m[0,\ T]$, $u_k \in L_2^\ell[0,\ T]$ and the input and output Hilbert spaces $L_2^\ell[0,\ T]$ and $L_2^m[0,\ T]$ are defined with inner products and associated induced norms

$$\langle u,\ v\rangle_R = \int_0^T u^\top(t)Rv(t)\mathrm{d}t,\ \ \|u\|_R = \sqrt{\langle u,\ u\rangle_R}, \tag{3.3}$$

$$\langle x,\ y\rangle_S = \int_0^T x^\top(t)Sy(t)\mathrm{d}t,\ \ \|y\|_S = \sqrt{\langle y,\ y\rangle_S} \tag{3.4}$$

in which $R \in \mathbb{S}_{++}^\ell$ and $Q \in \mathbb{S}_{++}^m$ are symmetric positive definite matrices. The convolution operator G and signal d take the form

$$(Gu)(t) = \int_0^t Ce^{A(t-s)}Bu(s)\mathrm{d}s,\ d(t) = Ce^{At}x_0 \tag{3.5}$$

where, without loss of generality, $d(t)$ may be absorbed into the reference to give $x_0 = 0$, $d(t) = 0$.

3.1.1 Point-to-Point ILC Framework

In point-to-point ILC, it is assumed that there are M critical points to be tracked during the whole time interval, $[0,\ T]$. The tracking time allocation of these points is given by

$$\Lambda = [t_1, t_2, \ldots, t_M]^\top \tag{3.6}$$

where $0 < t_1 < t_2 < \ldots < t_M \leqslant T$. Consider a signal $\zeta \in L_2^m[0,\ T]$ and define the linear mapping $\zeta \in L_2^m[0,\ T] \mapsto \zeta^p \in H$ defined as

$$\zeta^p = \begin{bmatrix} \zeta(t_1) \\ \zeta(t_2) \\ \vdots \\ \zeta(t_M) \end{bmatrix} \tag{3.7}$$

where H is a Hilbert space denoted by

$$H = \underbrace{\mathbb{R}^m \times \cdots \times \mathbb{R}^m}_{\text{M times}} \tag{3.8}$$

with inner product and associated induced norm

$$\langle \omega,\ \mu \rangle_{[Q]} = \sum_{i=1}^{M} \omega_i^\top Q_i \mu_i,\ \ \|\omega\|_{[Q]} = \sqrt{\langle \omega,\ \omega \rangle_{[Q]}} \tag{3.9}$$

where

$$\omega = [\omega_1, \ldots, \omega_M]^\top \in H,\ \mu = [\mu_1, \ldots, \mu_M]^\top \in H;$$

and $[Q]$ denotes the data set $\{Q_1,\ Q_2, \ldots,\ Q_M\}$ where for $i = 1, \ldots, M$ each $Q_i \in \mathbb{S}^m_{++}$ is a positive definite matrix. Using this notation, the plant output corresponding to tracking time allocation Λ is given by

$$y^p = (Gu)^p. \tag{3.10}$$

Since G is linear, this can be further written as

$$y^p = G^p_\Lambda u = (Gu)^p = \begin{bmatrix} G_1 u \\ G_2 u \\ \vdots \\ G_M u \end{bmatrix} \tag{3.11}$$

where $G^p_\Lambda : L^\ell_2[0,\ T] \to H$ is a linear operator with each component $G_i : L^\ell_2[0,\ T] \to \mathbb{R}^m$ defined by

$$G_i u = \int_0^{t_i} C e^{A(t_i - t)} B u(t) \mathrm{d}t. \tag{3.12}$$

In point-to-point ILC theory, only tracking of the output signal at each critical time instant, t_i, is required. As the number of trials tends to infinity, the point-to-point output, y^p_k, is required to converge to the point-to-point reference, r^p, (the point-to-point error $e^p_k = r^p - y^p_k$ converges to 0) and the input, u_k, converges to a unique value, u^*, i.e.

$$\lim_{k\to\infty} y^p_k = r^p,\ \ \lim_{k\to\infty} u_k = u^*. \tag{3.13}$$

3.1.2 Point-to-Point ILC with Optimal Tracking Time Allocation

The problem of point-to-point ILC with optimal tracking time allocation can be formulated by firstly considering the tracking time allocation as a variable, i.e. $\Lambda \in \Theta$, where Θ is the admissible set of tracking time allocation

$$\Theta = \{\Lambda \in \mathbb{R}^M : 0 < t_1^- \leqslant t_1 \leqslant t_1^+ \leqslant t_2^- \leqslant t_2 \leqslant t_2^+ \leqslant \ldots \leqslant t_M^+ = T\} \tag{3.14}$$

in which $[t_i^-,\ t_i^+]$ defines the (allowed) allocation interval for t_i representing the requirements on enforcing process timing and synchronization constraints necessary to complete the task, i.e. t_i^- and t_i^+ are the lower and upper bounds of the time instant t_i. Note that the choice of set Θ restricted the tracking time allocation to a subspace of the space $\mathbb{R}^M$, which reduces the complexity of the problem.

The **Point-to-Point ILC with Optimal Tracking Time Allocation Problem** can then be defined as iteratively finding a tracking time allocation, Λ_k, and an input, u_k, with the asymptotic property that the output values at these time instants, i.e. y_k^p, accurately pass through a set of desired points, r^p, with

$$\lim_{k\to\infty} y_k^p = r^p,$$

at the same time minimizing a target cost function $f(u, y)$ as a function of the system input u and output y, i.e.

$$\lim_{k\to\infty} (u_k,\ y_k,\ \Lambda_k) = (u_k^*,\ y_k^*,\ \Lambda_k^*)$$

where u_k^*, y_k^* and Λ_k^* are optimal solutions of the problem

$$\begin{aligned} \underset{u,y,\Lambda}{\text{minimize}} \quad & f(u,y) \\ \text{subject to} \quad & r^p = G_\Lambda^p u, \\ & y = Gu, \\ & \Lambda \in \Theta. \end{aligned} \tag{3.15}$$

Note that this problem formulation comprises a significant expansion of the current point-to-point ILC framework by exploiting the flexibilities in choosing the tracking time allocation, Λ, to optimize some performance of interest in addition to the tracking requirement. This however, as will be seen later, introduces substantial difficulties in algorithm design, which will be addressed in the following sections.

Remark 3.1. The index $f(u, y)$ represents the designer's requirements on the performance and should be chosen according to the specific application. As an example, if it is required to minimize the peak input, $f(u, y)$ can be chosen as

$$f(u, y) = \|u\|_\infty;$$

if it requires the system to minimize a function of the output, e.g. acceleration of a robotic movement, $f(u, y)$ can be chosen as

$$f(u, y) = \|g(y)\|_S$$

where the function $g(y)$ computes the output acceleration. In this chapter, for simplicity, the performance index $f(u, y)$ is chosen as a convex function of u and y. Note that this encompasses many real life applications.

Remark 3.2. It is worth pointing out that the general problem formulation in Hilbert space makes it possible for the techniques used in this chapter to be further extended to other systems, e.g. linear discrete time systems, switched linear systems and linear differential systems, the details of which however will differ and are not described in this chapter.

3.2 A Two Stage Design Framework

In this section, a two stage design framework is developed to solve the above point-to-point ILC design with optimal tracking time allocation problem. Note that while the tracking time allocation, Λ, does not explicitly appear in the performance index $f(u, y)$, they are connected by the tracking requirement $G^p_\Lambda u = r^p$ in a nonlinear manner. Furthermore, the input, u, lies in an infinite dimensional space, $L^\ell_2[0, T]$, and the tracking time allocation, Λ, lies in the finite dimensional space, Θ. The above aspects make the problem (3.15) non-trivial.

3.2.1 Framework Description

Optimization problem (3.15) can be written equivalently as

$$\min_{\Lambda\in\Theta}\left\{\min_u f(u, y),\ \text{subject to } G^p_\Lambda u = r^p,\ y = Gu\right\} \tag{3.16}$$

by optimizing over u first and then optimizing over Λ. Define the function $\tilde{f}$ of Λ by

$$\tilde{f}(\Lambda) = \min_u \left\{f(u, y),\ \text{subject to } G^p_\Lambda u = r^p,\ y = Gu\right\},$$

and denote a global minimizer for u of the inner optimization problem (3.16) as $u_\infty(\Lambda) : \Theta \to L^\ell_2[0, T]$, the optimization problem (3.16) is then equivalent to

$$\min_{\Lambda\in\Theta}\{\tilde{f}(\Lambda) := f(u_\infty(\Lambda), Gu_\infty(\Lambda))\}. \tag{3.17}$$

It follows that the point-to-point ILC with optimal tracking time allocation problem can be solved using the following two stage design framework:

- *Stage One:* Fix the tracking time allocation, Λ, and solve the inner optimal problem (3.16), i.e.

$$\begin{aligned} \underset{u,y}{\text{minimize}} \quad & f(u,y) \\ \text{subject to} \quad & r^p = G^p_\Lambda u, \\ & y = Gu. \end{aligned} \tag{3.18}$$

- *Stage Two:* Substitute the solution $u_\infty(\Lambda)$ of the problem (3.18) into the original optimization problem (3.16) and then solve the resulting optimization problem (3.17) to compute the optimal tracking time allocation, i.e.

$$\min_{\Lambda\in\Theta}\{\tilde{f}(\Lambda) := f(u_\infty(\Lambda), Gu_\infty(\Lambda))\}. \tag{3.19}$$

To exemplify the approach, the control effort is selected to be the target performance index in this chapter, so that $f(u,y) = \|u\|^2_R$. This guarantees the existence of a unique global minimizer for the inner optimization problem within (3.16), and the resulting optimization problems in Stage One and Two become

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \|u\|^2_R \\ \text{subject to} \quad & r^p = G^p_\Lambda u, \end{aligned} \tag{3.20}$$

and

$$\min_{\Lambda\in\Theta}\{\tilde{f}(\Lambda) := \|u_\infty(\Lambda)\|^2_R\} \tag{3.21}$$

respectively. Note that as the output, y, does not appear in the performance index, therefore the second constraint in problem (3.18), i.e. $y = Gu$, is not needed in optimization problem (3.20). It is worth pointing out that other performance indices rather than the input energy can also be used with no change in the form of the two stage design framework - the implementation of the resulting algorithms however will differ from those described in subsequent sections of this chapter.

As dictated by the ILC framework, the two stages must involve the use of experimental data in order to embed robustness against model uncertainties. Before this is discussed in detail in the next section, the solution of this two stage design framework is given below.

3.2.2 Solution of the Proposed Framework

3.2.2.1 Solution of Stage One Optimization Problem

For a given tracking time allocation Λ, the Stage One optimization problem is in fact a point-to-point ILC design problem with a minimum control energy requirement. This

can be solved efficiently using the point-to-point norm optimal ILC algorithm with a special initial input choice, as shown next.

Theorem 3.3. *If the system $S(A, B, C)$ is controllable and C has full row rank, the Stage One optimization problem (3.20) for a given tracking time allocation Λ can be solved by the norm-optimal point-to-point ILC algorithm*

$$u_{k+1} = \arg\min_{u} \{\|e^p\|^2_{[Q]} + \|u - u_k\|^2_R\} \tag{3.22}$$

proposed in Owens et al. (2013) with initial input $u_0 = 0$, such that

$$u_\infty = \lim_{k\to\infty} u_k.$$

The iterative solution is given by

$$u_{k+1} = u_k + G^{p*}_\Lambda (I + G^p_\Lambda G^{p*}_\Lambda)^{-1} e^p_k \tag{3.23}$$

where $G^{p*}_\Lambda : (\omega_1, ..., \omega_M) \in H \to u \in L^\ell_2[0, T]$ is the Hilbert adjoint operator of G^p_Λ defined by

$$\begin{aligned} &u(t) = R^{-1}B^\top p(t),\ \dot{p}(t) = -A^\top p(t), \\ &p(T) = 0,\ p(t_i-) = p(t_i+) + C^\top Q_i \omega_i,\ i = 1, \ldots, M \end{aligned} \tag{3.24}$$

with p denoting the costate vector and R, Q_i denoting the weighting matrices of tracking requirement, and the $Mm \times Mm$ matrix $G^p_\Lambda G^{p*}_\Lambda$ has a block structure with $(i, j)^{th}$ block

$$G_i G^*_j = \int_0^{min(t_i, t_j)} C e^{A(t_i - t)} B R^{-1} B^\top e^{A^\top (t_i - t)} C^\top Q_j \mathrm{d}t. \tag{3.25}$$

Furthermore, an analytic solution can be obtained for $u_\infty(\Lambda)$ as follows

$$u_\infty(\Lambda) = G^{p*}_\Lambda (G^p_\Lambda G^{p*}_\Lambda)^{-1} r^p. \tag{3.26}$$

Proof. The proof uses components of Owens et al. (2013) and is described in detail below. On the $(k+1)^{th}$ trial, the norm-optimal point-to-point ILC algorithm solves the optimization problem

$$\min_u \{\|e^p\|^2_{[Q]} + \|u - u_k\|^2_R :\ e^p = r^p - y^p,\ y^p = G^p_\Lambda u\} \tag{3.27}$$

to obtain the control input, u_{k+1}. The problem (3.27) has an identical structure to the norm-optimal ILC problem described in Amann et al. (1996b), with the only difference

being the definitions of the operators, signals and underlying Hilbert spaces. Therefore, the iterative solution can be expressed as

$$u_{k+1} = u_k + G_\Lambda^{p*} e_{k+1}^p \Rightarrow e_{k+1}^p = (I + G_\Lambda^p G_\Lambda^{p*})^{-1} e_k^p \tag{3.28}$$

which gives rise to (3.23).

It is proved in Owens et al. (2013) that if a system is controllable and C has full row rank, the reference, r^p, can be tracked exactly and the sequence, $\{u_k\}$, exists a limit, i.e.

$$\lim_{k\to\infty} e_k^p = 0, \quad \lim_{k\to\infty} u_k = u_\infty.$$

The algorithm converges to the minimum control energy that achieves the perfect tracking requirement if $u_0 = 0$. Hence the Stage One optimization problem (3.20) can be solved by the norm-optimal point-to-point ILC algorithm.

The relevant adjoint operator G_Λ^{p*} is obtained in Owens et al. (2013) from

$$\left\langle (\omega_1, ..., \omega_M),\ G_\Lambda^p u \right\rangle_{[Q]} = \left\langle G_\Lambda^{p*}(\omega_1, ..., \omega_M),\ u \right\rangle_R \tag{3.29}$$

which gives rise to

$$(G_i^* \omega_i)(t) = \begin{cases} R^{-1} B^\top e^{A^\top (t_i - t)} C^\top Q_i \omega_i,\ 0 \leqslant t \leqslant t_i, \\ 0,\ t > t_i. \end{cases} \tag{3.30}$$

The equation (3.30) can be further written as

$$(G_i^* \omega_i)(t) = R^{-1} B^\top p_i(t) \tag{3.31}$$

where $p_i(t) = 0$ on $(t_i,\ T]$, and on $[0,\ t_i)$

$$\dot{p}_i(t) = -A^\top p_i(t),\ p_i(t_i-) = C^\top Q_i \omega_i. \tag{3.32}$$

Adjoint operator G_Λ^{p*} is the map $(\omega_1, ..., \omega_M) \mapsto u$ defined by

$$u(t) = \sum_{i=1}^{M} (G_i^* \omega_i)(t) = R^{-1} B^\top p(t) \tag{3.33}$$

where $p(t) = \sum_{i=1}^M p_i(t)$. Due to the linearity, these equations yield the costate equation modified by 'jump conditions' at time t_i, which generates the definition (3.24) of the adjoint operator G_Λ^{p*}. Therefore, the $(i,j)^{th}$ block of the matrix $G_\Lambda^p G_\Lambda^{p*}$ can be computed as the equation (3.25).

To solve the optimization problem (3.20) analytically, the associated Lagrangian expression is given by

$$\mathcal{L}(u) = \|u\|_R^2 + 2\left\langle \lambda,\ G_\Lambda^p u - r^p \right\rangle_{[Q]}. \tag{3.34}$$

Following the method introduced in Amann (1996), let u_∞ be the global optimal u that minimize $\mathcal{L}$ and there exists

$$\mathcal{L}(u_\infty) \leqslant \mathcal{L}(u_\infty + \tau),\ \forall\ \tau \in L_2^\ell[0,\ T] \tag{3.35}$$

which can be equivalently written as

$$\begin{aligned}\mathcal{L}(u_\infty + \tau) - \mathcal{L}(u_\infty) &= \|\tau\|_R^2 + 2\left\langle u_\infty,\ \tau\right\rangle_R + 2\left\langle \lambda, G_\Lambda^p \tau\right\rangle_{[Q]} \\ &= \|\tau\|_R^2 + 2\left\langle u_\infty + G_\Lambda^{p*}\lambda,\ \tau\right\rangle_R \geqslant 0, \forall\ \tau \in L_2^\ell[0,\ T].\end{aligned} \tag{3.36}$$

Substitute $\tau = -(u_\infty + G_\Lambda^{p*}\lambda)$ into (3.36) to obtain

$$-\left\|u_\infty + G_\Lambda^{p*}\lambda\right\|_R^2 \geqslant 0. \tag{3.37}$$

It follows that the inequality condition in (3.37) only holds when $u_\infty = -G_\Lambda^{p*}\lambda$ which satisfies the condition (3.35) for all $\tau \in L_2^\ell[0,\ T]$. Then recall the tracking requirement to construct

$$\begin{cases} G_\Lambda^p u_\infty = r^p \\ u_\infty = -G_\Lambda^{p*}\lambda \end{cases} \tag{3.38}$$

which yields $G_\Lambda^p G_\Lambda^{p*}\lambda = -r^p$. As the matrix $G_\Lambda^p G_\Lambda^{p*}$ is invertible (as it does not have zero eigenvalue), it follows that $\lambda = -(G_\Lambda^p G_\Lambda^{p*})^{-1} r^p$, which together with $u_\infty = -G_\Lambda^{p*}\lambda$ give rise to the analytic solution (3.26). □

Note that the system controllability condition can be satisfied without difficulty as a controllable state space model can always be constructed for a given system and the requirement C has full row rank is not restrictive either as this simply implies no output component can be constructed from others, i.e. there is no redundant output, and is therefore assumed to hold for the rest of the thesis.

3.2.2.2 Solution of Stage Two Optimization Problem

With the analytic solution of Stage One optimization problem, the Stage Two optimization problem (3.21) can be further simplified as shown in the following lemma.

Lemma 3.4. *Based on the analytic solution (3.26) of Stage One optimization problem, the Stage Two optimization problem (3.21) can be expressed as*

$$\min_{\Lambda\in\Theta} \|u_\infty(\Lambda)\|_R^2 = \min_{\Lambda\in\Theta}\ \left\langle r^p,\ (G_\Lambda^p G_\Lambda^{p*})^{-1} r^p\right\rangle_{[Q]}. \tag{3.39}$$

Proof. Substituting the analytic solution (3.26) into the problem (3.21) and using the property of adjoint operator gives

$$\begin{aligned}\min_{\Lambda\in\Theta}\|u_\infty(\Lambda)\|_R^2 &= \min_{\Lambda\in\Theta}\ \langle (u_\infty(\Lambda),\ u_\infty(\Lambda)\rangle_R \\ &= \min_{\Lambda\in\Theta}\ \left\langle (G_\Lambda^{p*}(G_\Lambda^p G_\Lambda^{p*})^{-1}r^p, G_\Lambda^{p*}(G_\Lambda^p G_\Lambda^{p*})^{-1}r^p\right\rangle_R \\ &= \min_{\Lambda\in\Theta}\ \left\langle G_\Lambda^p G_\Lambda^{p*}(G_\Lambda^p G_\Lambda^{p*})^{-1}r^p, (G_\Lambda^p G_\Lambda^{p*})^{-1}r^p\right\rangle_{[Q]} \\ &= \min_{\Lambda\in\Theta}\ \left\langle r^p,\ (G_\Lambda^p G_\Lambda^{p*})^{-1}r^p\right\rangle_{[Q]}\end{aligned}$$

which completes the proof. □

Solving the above Stage Two optimization problem, however, is non-trivial except for the special case of $M = 1$, i.e. there is only one tracking point,in which case the solution can be obtained analytically, as shown in the following theorem.

Theorem 3.5. *When there is only one tracking point, i.e. $M = 1$, the solution of the Stage Two optimization problem (3.39) is*

$$\Lambda^* = T.$$

The corresponding minimum energy is

$$\min_{\Lambda\in\Theta}\|u_\infty(\Lambda)\|_R^2 = \left\langle r^p,\ \Psi_T^{-1}r^p\right\rangle_{[Q]}$$

where

$$\Psi_t = \int_0^t Ce^{A(t-s)}BR^{-1}(Ce^{A(t-s)}B)^\top \mathrm{d}s.$$

Proof. For $M = 1$, there is only one time-point and thus $\Lambda = t_1 \in \mathbb{R}$. Denote $\Psi_{t_1} = G_{t_1}^p G_{t_1}^{p*}$ which can be explicitly written as

$$\Psi_{t_1} = \int_0^{t_1} Ce^{A(t_1-t)}BR^{-1}(Ce^{A(t_1-t)}B)^\top \mathrm{d}t. \tag{3.40}$$

The Stage Two optimization problem becomes

$$\min_{\Lambda\in\Theta}\|u_\infty(\Lambda)\|_R^2 = \min_{\Lambda\in\Theta}\ \left\langle r^p,\ \Psi_{t_1}^{-1}r^p\right\rangle_{[Q]}. \tag{3.41}$$

For any x, it can be shown

$$\langle x,\ (\Psi_T - \Psi_{t_1})x\rangle_{[Q]} = \langle x,\ \int_{t_1}^T Ce^{A(t_1-t)}BR^{-1}(Ce^{A(t_1-t)}B)^\top \mathrm{d}t x\rangle_{[Q]} \geqslant 0,$$

so it follows that

$$\Psi_{t_1} \leqslant \Psi_T,\ \forall\ t_1^- \leqslant t_1 \leqslant t_1^+ = T.$$

The above properties of positive operators yield $\Psi_{t_1}^{-1} \geqslant \Psi_T^{-1}$, and therefore

$$\left\langle r^p,\ \Psi_{t_1}^{-1} r^p \right\rangle_{[Q]} \geqslant \left\langle r^p,\ \Psi_T^{-1} r^p \right\rangle_{[Q]}. \tag{3.42}$$

It follows that $t_1 = t_1^+ = T$ is an optimum of the Stage Two optimization problem, which completes the proof. □

Theorem 3.5 shows that when $M = 1$, $\Lambda^* = T$ is always an optimal choice in terms of minimizing the control input energy - this is not surprising as this allows the system output to change gradually to the desired position and thus less control energy can be expected. However, when $M > 1$, the performance index is generally non-linear and non-convex with respect to the tracking time allocation, Λ, leading to significant difficulties in solving Stage Two optimization problem (3.39). This is addressed in the following theorem using a gradient based algorithm.

Theorem 3.6. *For $M \geqslant 2$, Stage Two optimization problem (3.39) can be solved using the gradient based iterative method*

$$\Lambda_{j+1} = P_\Theta(\Lambda_j - \gamma_j \nabla \tilde{f}(\Lambda_j)) \tag{3.43}$$

where $j \in \mathbb{N}$ denotes the updating iteration (loop) number, $\nabla \tilde{f}(\Lambda_j) \in \mathbb{R}^M$ is the gradient of the function $\tilde{f}$, $P_\Theta(\cdot)$ denotes the projection operator, i.e.

$$P_\Theta(x) = \arg\min_{z \in \Theta} \|x - z\|^2,$$

and $\gamma_j > 0$ is a step size chosen by the generalized Armijo rule in Armijo (1966), i.e.

$$\gamma_j = \beta^{m_k} \gamma \tag{3.44}$$

where m_k is the smallest non-negative integer such that

$$\tilde{f}(\Lambda_{j+1}) - \tilde{f}(\Lambda_j) \leqslant \sigma (\nabla \tilde{f}(\Lambda_j))^\top (\Lambda_{j+1} - \Lambda_j) \tag{3.45}$$

and σ, β, γ are constant scalars with $0 < \sigma < 1$, $0 < \beta < 1$, $\gamma > 0$. The resulting sequence $\{\tilde{f}(\Lambda_j)\}$ decays to a limit $\tilde{f}^*$, i.e.

$$\tilde{f}(\Lambda_{j+1}) \leqslant \tilde{f}(\Lambda_j), \ \text{and} \ \lim_{j \to \infty} \{\tilde{f}(\Lambda_j)\} = \tilde{f}^* \tag{3.46}$$

and the sequence $\{\Lambda_j\}$ satisfies

$$\lim_{j \to \infty} \|\Lambda_j - \Lambda_{j+1}\| = 0 \tag{3.47}$$

with every limit point z of the sequence $\{\Lambda_k\}$ a stationary point for problem (3.39), i.e.

$$z = P_\Theta(z - \nabla \tilde{f}(z)). \tag{3.48}$$

Proof. To prove Theorem 3.6, the following lemma is needed.

Lemma 3.7. *Bertsekas (1976) Let* $\{\Lambda_k\}$ *be a sequence generated by* $\Lambda_{j+1} = P_\Theta(\Lambda_j - \gamma_j \nabla f(\Lambda_j))$ *where*

$$\Theta = \{\Lambda \in \mathbb{R}^M : \lambda_i \leqslant t_i \leqslant \mu_i,\ i = 1, \ldots, M\}, \tag{3.49}$$

and let γ_j *be chosen according to the generalized Armijo step size (3.44). Then every limit point of the sequence* $\{\Lambda_k\}$ *is a stationary point for problem (3.39).*

In this chapter, the admissible set Θ satisfies the constraint set requirement (3.49), and the gradient projection method (3.43) with generalized Armijo step size (3.44) is used in Theorem 3.6. Using this theorem, all the assumptions in Lemma 3.7 are satisfied, and hence the sequence $\{\Lambda_k\}$ converges to a stationary point of the problem (3.39). □

It is noted that being a stationary point satisfying (3.48) is a necessary condition of a (possibly locally) minimum point. Note that the function $\tilde{f}(\Lambda)$ is bounded below and Θ is a compact set. Therefore, the (global) minimum of the optimization problem exists and is a stationary point. When the problem only has one such point, it must be the minimum. In this case, the above algorithm converges to the global minimum solution following results in Goldstein (2012), i.e. the best result that can be achieved.

Remark 3.8. It is worth pointing out that other step size choices are also possible, e.g. constant step size (Goldstein, 2012)

$$0 < \mu \leqslant \gamma \leqslant \frac{2(1-\mu)}{L}, \tag{3.50}$$

where $L > 0$ is the Lipschitz constant of $\tilde{f}(\Lambda)$ on Θ and $\mu \in (0,\ 2/(2+L)]$ is a positive scalar, and the projected Barzilai-Borwein step size (Birgin et al., 2000)

$$\gamma_j = \frac{\langle \Delta x_j,\ \Delta g_j \rangle}{\langle \Delta g_j,\ \Delta g_j \rangle},\ \text{or}\ \gamma_j = \frac{\langle \Delta x_j,\ \Delta x_j \rangle}{\langle \Delta x_j,\ \Delta g_j \rangle} \tag{3.51}$$

where $\Delta x_j = \Lambda_j - \Lambda_{j-1}$, $\Delta g_k = \nabla \tilde{f}(\Lambda_j) - \nabla \tilde{f}(\Lambda_{j-1})$. Using these step size choices, the convergence properties will be different from those stated in the above theorem and are omitted here for brevity.

3.2.3 A Numerical Example

In this subsection, a numerical example is presented to verify the results in Theorem 3.5 for one tracking point, and to illustrate the design difficulties when there are more than one tracking point. Consider the following system model

$$G_z(s) = \frac{15.8869(s + 850.3)}{s(s^2 + 707.6s + 3.377 \times 10^5)} \tag{3.52}$$

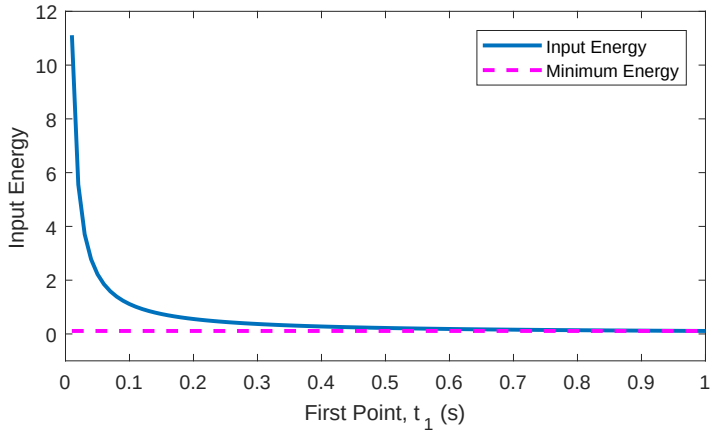


Figure 3.1: Input Energy $\tilde{f}(\Lambda)$ for a Single Point Case ($M = 1$).

which is used in Hladowski et al. (2011) with a proportional feedback gain of 100 to model the gantry robot system employed in Section 3.5.1. Firstly, it is supposed that $M = 1$, i.e. there is only one tracking point, the trial length is $T = 2s$, and the tracking reference is $r^p = 0.01$. The admissible set of tracking time allocation Θ in (3.14) is defined by the parameters $t_1^+ = 2s$ and $t_1^- = 0.01s$. The input energy at a particular time allocation t_1 can be computed analytically using equations (3.40) and (3.41), and the result is plotted in Figure 3.1. It is clear from this figure that the minimum input energy is achieved at $t_1 = T$, verifying the theoretical prediction in Theorem 3.5.

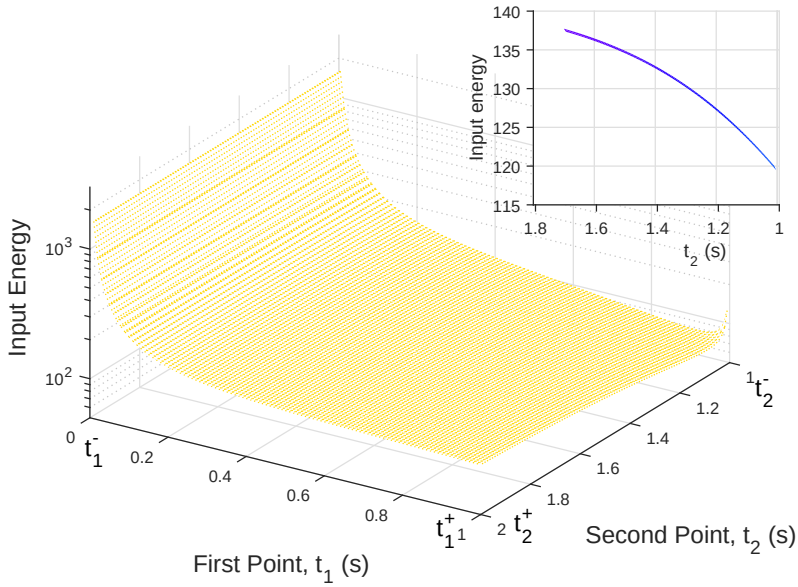


Figure 3.2: Input Energy $\tilde{f}(\Lambda)$ at Multiple Point Case ($M = 2$).

Now suppose $M = 2$, the trial length stays the same as $T = 2s$, and the tracking reference is given by $r^p = [0.01,\ 0.008]^\top$. Furthermore, the admissible set of tracking time allocation Θ in (3.14) is defined by parameters $t_1^+ = 1s$, $t_2^+ = 2s$, $t_1^- = 0.01s$, and $t_2^- = 1.01s$. The required input energy to achieve the design tracking task as a function

of tracking time instants t_1 and t_2 is plotted in Figure 3.2. It is clear from this figure (the zoomed-in cross section area) that $\tilde{f}(\Lambda)$ is non-linear and non-convex with respect to Λ, indicating the difficulties in solving Stage Two problem (3.39). In this case the algorithm in Theorem 3.6 can be applied to solve this problem, the results of which will be verified experimentally later in Section 3.5.

3.3 Implementation of the Design Approach

In the previous section, a two stage design framework was proposed. Its implementation is now discussed in detail.

3.3.1 Implementation of Stage One Design

The general update (3.23) of Stage One design can be either computed directly using the analytic solution (3.26), or implemented experimentally using the following feedback plus feedforward algorithm.

Proposition 3.9. *The Stage One update (3.23) can be implemented using the feedforward plus feedback implementation*

$$u_{k+1}(t) = u_k(t) + R^{-1}B^\top p_k(t) \tag{3.53}$$

with

$$p_k(t) = -K(t)(x_{k+1}(t) - x_k(t)) + \xi_{k+1}(t) \tag{3.54}$$

where p_k denotes the costate, $K(t)$ denotes the Riccati feedback matrix

$$\begin{aligned} 0 &= \dot{K}(t) + (A^\top - K(t)BR^{-1}B^\top)K(t) + K(t)A, \\ K(T) &= 0, \\ K(t_i-) &= K(t_i+) + C^\top Q_i C,\ i = 1, \ldots, M, \end{aligned} \tag{3.55}$$

and $\xi_{k+1}(t)$ denotes the predictive feedforward term given at the $(k+1)^{th}$ trial by

$$\begin{aligned} 0 &= \dot{\xi}_{k+1}(t) + (A^\top - K(t)BR^{-1}B^\top)\xi_{k+1}(t), \\ \xi_{k+1}(T) &= 0, \\ \xi_{k+1}(t_i-) &= \xi_{k+1}(t_i+) + C^\top Q_i e_k(t_i),\ i = 1, \ldots, M. \end{aligned} \tag{3.56}$$

Proof. The ILC update (3.23) is equivalent to

$$u_{k+1}(t) = u_k(t) + (G_\Lambda^{p*} e_{k+1}^p)(t), \tag{3.57}$$

and $(G_\Lambda^{p*} e_{k+1}^p)(t)$ can be written as

$$\begin{aligned}(G_\Lambda^{p*} e_{k+1}^p)(t) &= R^{-1}B^\top p_k(t),\\ \dot{p}_k(t) &= -A^\top p_k(t),\\ p_k(T) &= 0,\\ p_k(t_i-) &= p_k(t_i+) + C^\top Q_i e_{k+1}(t_i),\end{aligned} \tag{3.58}$$

according to the costate equation (3.24). Hence (3.57) becomes

$$u_{k+1}(t) = u_k(t) + R^{-1}B^\top p_k(t). \tag{3.59}$$

Then substitute the equation

$$p_k(t) = -K(t)(x_{k+1}(t) - x_k(t)) + \xi_{k+1}(t) \tag{3.60}$$

into the jump condition at t_i of costate equation (3.58) to give

$$-(K(t_i+)-K(t_i-))(x_{k+1}(t_i)-x_k(t_i))+(\xi_{k+1}(t_i+)-\xi_{k+1}(t_i-)) = C^\top Q_i e_{k+1}(t_i) \tag{3.61}$$

and the error $e_{k+1}(t_i)$ can be further equivalently written as

$$e_{k+1}(t_i) = r_i - y_{k+1}(t_i) = e_k(t_i) - C(x_{k+1}(t_i) - x_k(t_i)). \tag{3.62}$$

Hence (3.61) and (3.62) give rise to the jump conditions at t_i in (3.55) and (3.56). Then use the method proposed in Amann (1996) to differentiate (3.54) at any point t not in Λ and substitute for $\dot{x}_k$ and $\dot{x}_{k+1}$. These provide the Riccati and predictive differential equations given in (3.55) and (3.56). □

It it worth pointing out that although both implementation methods can solve the Stage One optimization problem, the feedback plus feedforward implementation has potential robust performance (due to the introduction of state feedback) when applied to the true plant (more details can be found in Owens et al. (2013)), and therefore is preferable in practice.

3.3.2 Implementation of Stage Two Design

The Stage Two gradient based design method (3.43) involves two steps: a gradient update step and a projection step. The gradient update step is

$$\tilde{\Lambda}_{j+1} = \Lambda_j - \gamma_j \nabla \tilde{f}(\Lambda_j)$$

where the selection of γ_j is dictated by (3.44) and $\tilde{\Lambda}_{j+1}$ denotes the intermediate solution obtained for the gradient update step at the j^{th} loop. The gradient can either be

computed analytically using (3.39), or using a computationally efficient estimation

$$\left.\frac{\partial \tilde{f}}{\partial t_i}\right|_{\Lambda_j} = \frac{\tilde{f}(\Lambda_j^{i+}) - \tilde{f}(\Lambda_j^{i-})}{2\Delta T} \tag{3.63}$$

where $\Lambda_j^{i+} = [t_1^j,\ t_2^j, \ldots,\ t_i^j + \Delta T, \ldots,\ t_M^j]^\top$ and $\Lambda_j^{i-} = [t_1^j,\ t_2^j, \ldots,\ t_i^j - \Delta T, \ldots,\ t_M^j]^\top$, and $\Delta T \in \mathbb{R}$ is a sufficiently small number. It should be noted that both analytic calculation and experimental testing can be used to compute the fixed tracking time allocation's optimal energy $\tilde{f}(\Lambda_j^{i+})$ and $\tilde{f}(\Lambda_j^{i-})$ in (3.63). The projection step is performed to obtain the tracking time allocation Λ_{j+1} for the next loop based on $\tilde{\Lambda}_{j+1}$, i.e.

$$\Lambda_{j+1} = P_\Theta(\tilde{\Lambda}_{j+1}) = \arg\min_{\Lambda \in \Theta} \|\Lambda - \tilde{\Lambda}_{j+1}\|.$$

Note that this can be formulated into the following quadratic programming (QP) problem

$$\begin{aligned} \underset{\Lambda}{\text{minimize}} \quad & \|\Lambda - \tilde{\Lambda}_{j+1}\|^2 \\ \text{subject to} \quad & \hat{A}\Lambda - b \preceq 0 \end{aligned}$$

where $\hat{A} = [I, -I]^\top$, $b = [t_1^+, ..., t_M^+, -t_1^-, ..., -t_M^-]^\top$ and the symbol $\preceq$ denotes the component-wise inequality. This QP problem can be solved efficiently using standard QP solvers, e.g. using Matlab function `quadprob`.

As an iterative algorithm, the choice of initial tracking time allocation Λ_0 may affect the algorithm's convergence performance, as in most non-linear and non-convex optimization problems. Therefore, three methods are now proposed to provide an appropriate initial tracking time allocation for the algorithm to get better system performance.

3.3.2.1 Central initial tracking time allocation

In this method, all the initial tracking points are specified in the center of their time intervals, and the initial tracking time allocation is hence

$$\Lambda_0^c = [t_1^c,\ t_2^c, \ldots,\ t_M^c]^\top \tag{3.64}$$

where $t_i^c = (t_i^- + t_i^+)/2$.

3.3.2.2 Greedy initial tracking time allocation

This method is defined by Algorithm 3.10 which takes M 'greedy' steps to obtain the greedy initial tracking time allocation

$$\Lambda_0^g = [t_1^g,\ t_2^g, \ldots,\ t_M^g]^\top \tag{3.65}$$

based on the central initial tracking time allocation, Λ_0^c. Each 'greedy' step only computes a single optimal time-point, t_i^*, and the other time-points are treated as constants, i.e. $t_1^*, \ldots, t_{i-1}^*$ and $t_{i+1}^c, \ldots, t_M^c$. Algorithm 3.10 requires some additional computations but tries to provide a better initial tracking time allocation, Λ_0^g, compared to Λ_0^c.

Algorithm 3.10. Given system state space model, $S(A, B, C)$, desired tracking reference, r^p, central initial tracking time allocation, Λ_0^c and admissible set of tracking time allocation, Θ, the following steps provide the solution to the greedy initial tracking time allocation, Λ_0^g, i.e.

1: **for** $i = 1$ to M **do**
2: Let $\Lambda_i = [t_1^*,\ t_2^*, \ldots,\ t_i, \ldots,\ t_M^c]^\top$.
3: Solve the following problem using Theorem 3.6

$$t_i^* = \arg\min_{t_i} \tilde{f}(\Lambda_i). \tag{3.66}$$

4: **end for**
5: **return** $\Lambda_0^g = [t_1^*,\ t_2^*, \ldots,\ t_M^*]^\top$

3.3.2.3 Low resolution initial tracking time allocation

Low resolution initial tracking time allocation can be implemented by using Algorithm 3.11, which involves performing a grid search in order to approximate the optimal tracking time allocation based on the nominal plant model. The solution is denoted as

$$\Lambda_0^l = [t_1^l,\ t_2^l, \ldots,\ t_M^l]^\top \tag{3.67}$$

which minimizes the performance index. The term 'low resolution' implies that the sampling time, T_s, is suitably large, and hence the total number of time-point combinations, i.e. number of elements in $\tilde{\Theta}$, should not be excessive. Therefore this method can balance computation time and accuracy in approximating the global solution. However, this method may require a significant amount of time to carry out the grid search procedure when the number of time-points is large.

Algorithm 3.11. Given system state space model, $S(A, B, C)$, desired tracking reference, r^p, admissible set of tracking time allocation, Θ, and sample time, T_s, the following steps provide the solution to the low resolution initial tracking time allocation, Λ_0^l, i.e.

1: Discretize the infinite set Θ at a sample rate of T_s to obtain

$$\tilde{\Theta} = \{\Lambda \in \mathbb{R}^M : 0 < t_1^- \leqslant t_1 \leqslant t_1^+ \leqslant t_2^- \leqslant t_2 \leqslant t_2^+ \\ \leqslant \ldots \leqslant t_M^+ = T,\ t_i = n_i T_s,\ n_i \in \mathbb{N},\ i = 1, \ldots, M\}$$

which is a finite subset of Θ.

2: Solve the optimization problem below using a blind search

$$\Lambda^* = \arg\min_{\Lambda \in \tilde{\Theta}} \tilde{f}(\Lambda). \tag{3.68}$$

3: **return** $\Lambda_0^l = \Lambda^*$

3.3.3 An Iterative Implementation Algorithm

Combining the implementation of Stage One and Stage Two designs leads to an iterative implementation of the two stage design framework - Algorithm 3.12. Note that Λ_0 is a suitably chosen initial tracking time allocation, and $\epsilon > 0$, $\delta > 0$ are small scalars which depend on the tracking precision requirement and performance requirement, respectively.

Algorithm 3.12. Given initial tracking time allocation, Λ_0, system state space model, $S(A, B, C)$, desired tracking reference, r^p, an admissible set of tracking time allocation, Θ, weighting matrices, R and Q_i, the following steps provide the solution to the optimal tracking time allocation, Λ_{opt}, and input, u_{opt}, i.e.

1: **initialization:** Loop number $j = 0$

2: Repeatedly implement Stage One update (3.23) with $\Lambda = \Lambda_0$ experimentally using feedback plus feedforward update (3.53) until convergence, i.e. $\|e_k^p\| < \epsilon \|r^p\|$; record converged input, $u_\infty^{ex}(\Lambda_0)$, and input energy, $\tilde{f}(\Lambda_0)$.

3: **repeat**

4: Compute the gradient using (3.39) or (3.63) with $r^p = G_{\Lambda_j}^p u_\infty^{ex}(\Lambda_j)$; implement Stage Two update (3.43).

5: Set $j \to j + 1$.

6: Repeatedly implement Stage One update (3.23) with $\Lambda = \Lambda_j$ experimentally using feedback plus feedforward update (3.53) until convergence, i.e. $\|e_k^p\| < \epsilon \|r^p\|$; record converged input, $u_\infty^{ex}(\Lambda_j)$, and input energy, $\tilde{f}(\Lambda_j)$.

7: **until** $\left|\tilde{f}(\Lambda_j) - \tilde{f}(\Lambda_{j-1})\right| < \delta \left|\tilde{f}(\Lambda_{j-1})\right|$

8: **return** $\Lambda_{opt} = \Lambda_j$ and $u_{opt} = u_\infty^{ex}(\Lambda_j)$

It is essential to note that in Algorithm 3.12, Step 2 and 6 (i.e. the norm-optimal point-to-point ILC algorithm) are required to be implemented experimentally on the test platform and Step 4 involves the usage of experimental data, $u_\infty^{ex}(\Lambda_j)$. These requirements are not necessary when an accurate system model is known. However when there exists model mismatch/uncertainty, the proposed algorithm embeds appealing robustness properties as the algorithm 'learns' information concerning the real plant dynamics through exploitation of experimental data. This will be further demonstrated in subsequent experimental results.

3.4 Constrained Input Condition Handling

The previous sections propose a two stage design approach for point-to-point ILC with optimal tracking time allocation. This section further extends the proposed method to incorporate system constraints into the design.

3.4.1 Optimal Tracking Time Allocation with System Constraints

In practice, constraints exist widely in control systems due to physical limitations or performance requirements. For example, input constraints typically assume one of the forms introduced in Section 2.4.1. With the addition of input constraints, the optimization problem (3.15) becomes

$$\begin{aligned} \underset{u,y,\Lambda}{\text{minimize}} \quad & f(u,y) \\ \text{subject to} \quad & r^p = G_\Lambda^p u, \\ & y = Gu, \\ & \Lambda \in \Theta,\ u \in \Omega. \end{aligned} \tag{3.69}$$

As will be seen later, the constraints add significant difficulties into the algorithm design. In this section, only input constraints are considered. Note that in principle, the design developed in the following section can handle output constraints as well, but the details will be different and are omitted here for brevity.

3.4.2 Modified Two Stage Design Framework with Input Constraints

Following a similar procedure to that in Section 3.2, the constrained optimization problem (3.69) becomes

$$\min_{\Lambda\in\Theta} \left\{ \min_u f(u,y),\ \text{s.t. } G_\Lambda^p u = r^p,\ y = Gu,\ u \in \Omega \right\} \tag{3.70}$$

suggesting a possible two stage design framework:

- *Stage One:*

$$\begin{aligned} \underset{u\in\Omega}{\text{minimize}} \quad & \|u\|_R^2 \\ \text{subject to} \quad & r^p = G_\Lambda^p u \end{aligned} \tag{3.71}$$

  whose solution is denoted as $\hat{u}_\infty(\Lambda)$.

- *Stage Two:*

$$\min_{\Lambda\in\Theta}\{\tilde{f}(\Lambda) := \|\hat{u}_\infty(\Lambda)\|_R^2\}. \tag{3.72}$$

With the presence of input constraints, the problem becomes significantly more difficult, as the Stage One inner optimization problem needs to solve a constrained optimization problem, which unfortunately is inherently challenging and does not admit an analytic solution that is essential to the Stage Two optimization problem. To address this difficulty, a modified two stage design is proposed as follows.

Note that now Stage One does not have a direct analytic solution, but the norm-optimal ILC algorithm with successive projection proposed in Chu et al. (2015) can be applied to solve the modified Stage One optimization problem (3.71). The update (3.23) is accordingly replaced by two alternative update methods.

- *Method 1:* Solve the constrained input norm-optimal point-to-point ILC optimization problem

$$u_{k+1} = \arg\min_{u\in\Omega}\{\|e^p\|_{[Q]}^2 + \|u - u_k\|_R^2\}. \tag{3.73}$$

  This algorithm converges to the minimum error norm. The constrained QP problem (3.73) becomes computationally demanding which might introduce problems in some applications, especially when the trial length is large. A number of methods have been proposed to address this problem, see Chu et al. (2015); Chu and Owens (2009) for more information.

- *Method 2:* Solve the unconstrained input norm-optimal point-to-point ILC optimization problem

$$\tilde{u}_{k+1} = \arg\min_{u}\{\|e^p\|_{[Q]}^2 + \|u - u_k\|_R^2\} \tag{3.74}$$

  and then perform a simple input projection

$$u_{k+1} = \arg\min_{u\in\Omega} \|u - \tilde{u}_{k+1}\|. \tag{3.75}$$

  It is clear that the first step has an analytic solution and the solution of the second step is straightforward as the input constraint, Ω, is usually of a pointwise form in practice. This method is computationally simpler than Method 1 and can be carried out for large scale applications. Its convergence performance property, however, is different from that of Method 1.

The solution of Stage Two optimization problem (3.72), i.e. Step 6 in Algorithm 3.12, is modified as

$$\Lambda_{j+1} = \arg\min_{\Lambda\in\Theta} \tilde{f}_j(\Lambda) \tag{3.76}$$

where

$$\tilde{f}_j(\Lambda) = \|u_\infty(\Lambda)\|_R^2 + \rho\,\|u_\infty(\Lambda) - \hat{u}_\infty(\Lambda_j)\|_R^2,\ \rho \geqslant 0, \tag{3.77}$$

and $\hat{u}_\infty(\Lambda_j)$ is the converged input of the Stage One design for tracking time allocation, Λ_j. Note that in this modified Stage Two design, the input constraint is decoupled from the optimization problem and thus can be solved analytically (using the algorithm in Theorem 3.6).

These new solution forms combine to generate Algorithm 3.14 for the optimal tracking time allocation problem in point to point ILC with system constraints.

Remark 3.13. It is also possible to estimate the gradient, $\nabla\tilde{f}(\Lambda)$, in (3.72) experimentally following similar procedures to those discussed in Section 3.3.2. This is an alternative to computing it analytically using the above modified Stage Two design, the details of which are omitted here for brevity.

Algorithm 3.14. Given initial tracking time allocation, Λ_0, system state space model, $S(A, B, C)$, desired tracking reference, r^p, an admissible set of tracking time allocation, Θ, an input constraint set, Ω, weighting matrices, R and Q_i, the followings steps provide the solution to the optimal tracking time allocation, Λ_{opt}, and input, u_{opt}, i.e.

1: **initialization:** Loop number $j = 0$
2: Repeatedly implement Stage One update (3.73) or (3.74)-(3.75) with $\Lambda = \Lambda_0$ experimentally using the update (3.53) until convergence, i.e. $\|e_k^p\| < \epsilon\|r^p\|$; record converged input, $\hat{u}_\infty^{ex}(\Lambda_0)$, input energy, $\tilde{f}(\Lambda_0)$.
3: **repeat**
4: Compute the gradient using (3.26) or (3.63) with $r^p = G_{\Lambda_j}^p \hat{u}_\infty^{ex}(\Lambda_j)$; implement Stage Two update (3.76).
5: Set $j \to j + 1$.
6: Repeatedly implement Stage One update (3.73) or (3.74)-(3.75) with $\Lambda = \Lambda_j$ experimentally using the update (3.53) until convergence, i.e. $\|e_k^p\| < \epsilon\|r^p\|$; record converged input, $\hat{u}_\infty^{ex}(\Lambda_j)$, input energy, $\tilde{f}(\Lambda_j)$.
7: **until** $\left|\tilde{f}_j(\Lambda_j) - \tilde{f}(\Lambda_{j-1})\right| < \delta\left|\tilde{f}(\Lambda_{j-1})\right|$
8: **return** $\Lambda_{opt} = \Lambda_j$ and $u_{opt} = \hat{u}_\infty^{ex}(\Lambda_j)$

3.4.3 Convergence Properties of the Algorithm

Algorithm 3.14 has the following convergence properties:

Theorem 3.15. *Suppose perfect tracking is achievable and $\kappa \leqslant 1$, then the analytic input energy resulting from (3.76) satisfies*

$$\|u_\infty(\Lambda_{j+1})\|_R^2 \leqslant \|\hat{u}_\infty(\Lambda_j)\|_R^2 . \tag{3.78}$$

Proof. As Λ_{j+1} is the solution of the gradient projection, it is clear from Theorem 3.6 that the inequality $\tilde{f}_j(\Lambda_{j+1}) \leqslant \tilde{f}_j(\Lambda_j)$ holds and hence it follows that

$$\begin{aligned}
\|u_\infty(\Lambda_{j+1})\|_R^2 &\leqslant \kappa \|u_\infty(\Lambda_{j+1}) - \hat{u}_\infty(\Lambda_j)\|_R^2 + \|u_\infty(\Lambda_{j+1})\|_R^2 \\
&\leqslant \|u_\infty(\Lambda_j)\|_R^2 + \kappa \|u_\infty(\Lambda_j) - \hat{u}_\infty(\Lambda_j)\|_R^2 \\
&= \kappa \|\hat{u}_\infty(\Lambda_j)\|_R^2 + (1+\kappa) \|u_\infty(\Lambda_j)\|_R^2 - 2\kappa \langle \hat{u}_\infty(\Lambda_j),\ u_\infty(\Lambda_j)\rangle_R .
\end{aligned} \tag{3.79}$$

Then, recall the analytic solution (3.26) for $u_\infty(\Lambda)$ and the perfect tracking assumption $G_\Lambda^p \hat{u}_\infty(\Lambda) = r^p$ to give

$$\begin{aligned}
\langle \hat{u}_\infty(\Lambda),\ u_\infty(\Lambda)\rangle_R &= \langle \hat{u}_\infty(\Lambda),\ G_\Lambda^{p*}(G_\Lambda^p G_\Lambda^{p*})^{-1} r^p \rangle_R \\
&= \langle G_\Lambda^p \hat{u}_\infty(\Lambda), (G_\Lambda^p G_\Lambda^{p*})^{-1} r^p \rangle_{[Q]} \\
&= \langle r^p, (G_\Lambda^p G_\Lambda^{p*})^{-1} r^p \rangle_{[Q]} \\
&= \langle G_\Lambda^{p*}(G_\Lambda^p G_\Lambda^{p*})^{-1} r^p,\ G_\Lambda^{p*}(G_\Lambda^p G_\Lambda^{p*})^{-1} r^p \rangle_R \\
&= \langle u_\infty(\Lambda),\ u_\infty(\Lambda)\rangle_R .
\end{aligned} \tag{3.80}$$

Substitute (3.80) into (3.79) to give

$$\|u_\infty(\Lambda_{j+1})\|_R^2 \leqslant (1-\kappa) \|u_\infty(\Lambda_j)\|_R^2 + \kappa \|\hat{u}_\infty(\Lambda_j)\|_R^2 . \tag{3.81}$$

It is clear that the unconstrained converged input energy is no larger than the constrained converged input energy i.e.

$$\|u_\infty(\Lambda_j)\|_R^2 \leqslant \|\hat{u}_\infty(\Lambda_j)\|_R^2 ,$$

and it follows that

$$(1-\kappa) \|u_\infty(\Lambda_j)\|_R^2 \leqslant (1-\kappa) \|\hat{u}_\infty(\Lambda_j)\|_R^2 \tag{3.82}$$

as $(1-\kappa)$ is non-negative. Hence combine (3.82) and (3.81) together to generate the inequality (3.78). □

Although Theorem 3.15 only states that the next loop's unconstrained minimum energy is no larger than the constrained minimum energy of the current loop, it still provides useful information about the convergence properties of Algorithm 3.14. A series of simulations using different models and input constraints have been undertaken to examine the convergence properties of Stage Two update (3.76). The results demonstrate that although not proved, the proposed algorithm achieves monotonic convergence of the constrained minimum input energy, which is very appealing in practice. One representative simulation result is shown in the next subsection.

3.4.4 A Numerical Example

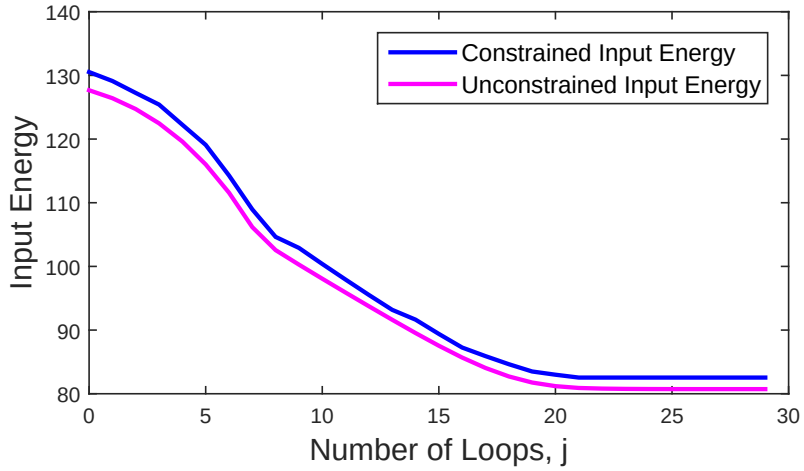


Figure 3.3: Convergence Performance Comparison between Constrained and Unconstrained Minimum Input Energy at Each Loop.

The same design objectives as the multiple case ($M = 2$) in Section 3.2.3 are considered as well as the input saturation constraint (2.55) with $M(t) = 1.5$. Perform Algorithm 3.14 for a total of 30 loops with the Stage Two update (3.76) in simulation assuming $\sigma = 0.1$, $\beta = 0.8$ and $\gamma = 0.08$. The corresponding constrained and unconstrained minimum input energy $\|\hat{u}_\infty(\Lambda_j)\|_R^2$ and $\|u_\infty(\Lambda_j)\|_R^2$ are plotted for each loop in Figure 3.3. The results in Figure 3.3 not only verify the property in (3.78), but also demonstrate that the proposed algorithm achieves monotonic convergence of the constrained minimum input energy, which is very appealing in practice.

3.5 Experimental Verification on a Gantry Robot

The proposed design framework is now validated experimentally on a three-axis gantry robot test facility to demonstrate its effectiveness on a widely used industrial platform.

3.5.1 Test Platform Specification

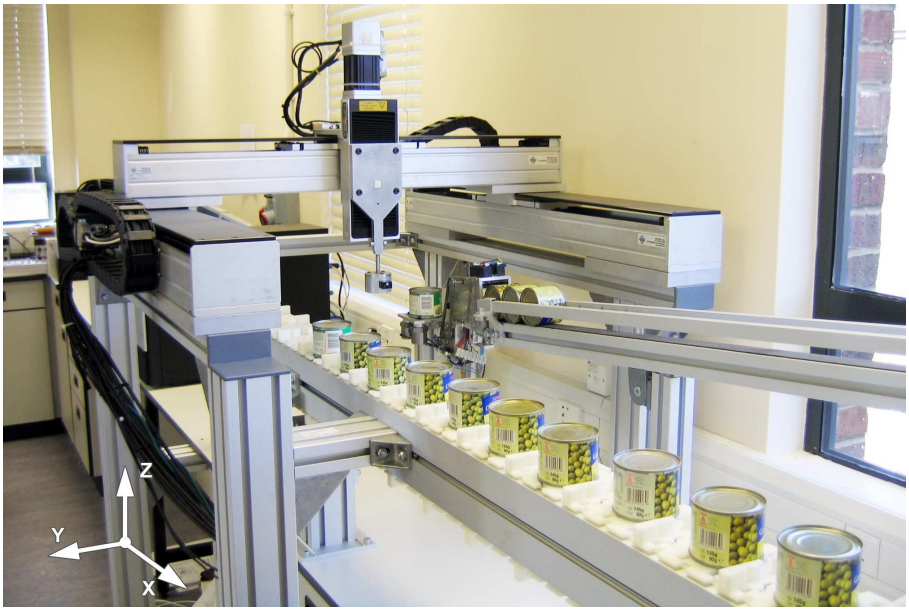


Figure 3.4: Multi-axis Gantry Robot Test Platform.

The multi-axis gantry robot shown in Figure 3.4 is employed as a test platform, which comprises three perpendicular axes. The x-axis and the y-axis move in the horizontal plane and are driven by linear brush-less dc motors. The vertical z-axis is placed on the top of the other two axes, and has a linear ball-screw stage driven by a rotary brush-less dc motor. The optical incremental encoders are used to measure axis displacement data. The combined motion of the three axes enable the gantry robot's end-effector to move within the 3D space. The three axes have been modelled based on frequency response tests in Ratcliffe (2005) with resulting transfer functions

$$G_x(s) = \frac{1.67 \times 10^{-5}(s+500.2)(s+4.9\times10^5)(s^2+10.58s+1.145\times10^4)\dots}{s(s^2+24s+6401)(s^2+21.38s+2.017\times10^4)\dots}$$
$$\frac{\dots(s^2+21.98s+2.9\times10^4)}{\dots(s^2+139.5s+2.162\times10^5)},\ G_y(s) = \frac{0.59s^4-19.86s^3+1.05s^2+15.92s-8.58}{s^5-4.19s^4+7.24s^3-6.45s^2+2.96s-0.56}$$
$$\times 10^{-6},\ \text{and}\ G_z(s) = \frac{15.8869(s+850.3)}{s(s^2+707.6s+3.377\times10^5)}. \tag{3.83}$$

The powerful dSPACE device is chosen to build up the interface between the software, i.e. host computer, and the hardware, i.e. gantry robot. For each axis, a BNC channel is used to send the input voltage to the motor amplifier, and an incremental encoder channel is used to receive the position feedback of the motor. The overall gantry robot control loop is shown in Figure 3.5.

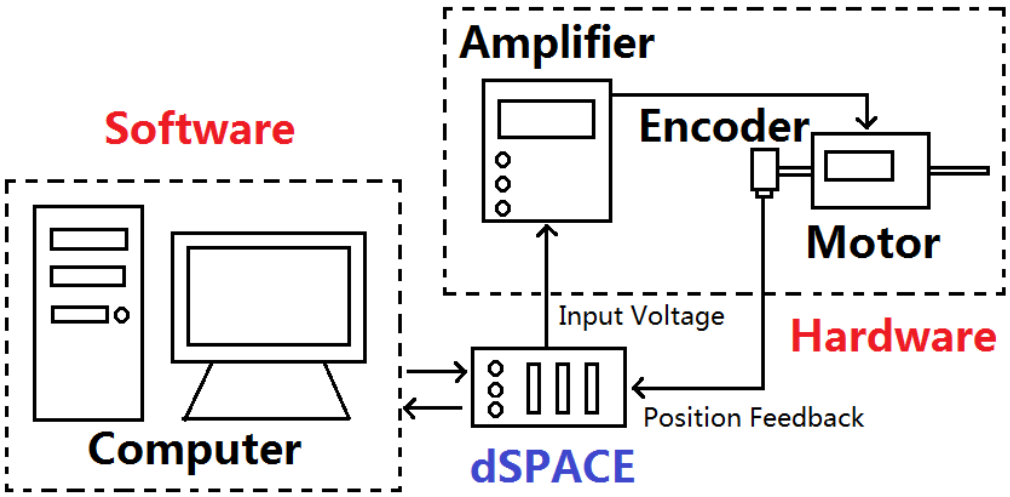


Figure 3.5: Gantry Robot Test Platform Control Loop.

3.5.2 Experimental Results

The control design objective is to perform a pick-and-place task with two special tracking points ($M = 2$), which correspond to the 'pick' position and the 'place' position as shown in Figure 1.2 from an exemplary tracking trajectory. For simplicity, only the z-axis is considered in this chapter with system model, $G_z(s)$, and take the time index to two decimal place accuracy. The total trial length is $T = 1.99s$ and the reference for z-axis, r^p, is $[0.01, 0.008]^\top$. The parameter choice $t_1^+ = 0.99s$, $t_2^+ = 1.99s$, $t_1^- = 0.01s$, and $t_2^- = 1.01s$ ensures that the robot moves to the pick position first and then to the place position. To provide baseline tracking, disturbance rejection and smooth tracking at turning edge, a proportional controller with gain K, is applied around the system, yielding

$$G(s) = \frac{G_z(s)}{1 + KG_z(s)}. \tag{3.84}$$

The transfer function system model, $G(s)$, can be equivalently written in minimal state space form $S(A, B, C)$. Note that for this problem, previous studies used a predefined (*a priori*) tracking time allocation $\Lambda_r = [0.5, 1.35]^\top$, with a corresponding control input energy obtained by implementing the Stage One update only. The central initial tracking time allocation $\Lambda_0 = [0.5, 1.5]^\top$ is used in both Algorithm 3.12 and 3.14, the weighting matrices are taken as $Q_i = qI$ for $i = 1, \ldots, M$, $R = rI$ where q and r are positive scalars, and the gradient is obtained using the estimation (3.63) via analytic calculation. Furthermore, appropriate weighting matrices are chosen according to the theoretical predictions in Owens et al. (2013) to balance convergence speed and robust performance, i.e. $q/r = 500,000$. Note that the choice of Q_i and R is not restricted to the values used in this section and have a wide range of other options.

First assume that only an approximate system model of the z-axis is available as follows:

$$\hat{G}_z(s) = \frac{0.03}{s} \tag{3.85}$$

with a feedback gain $K = 30$. A 60 loop updating procedure of Algorithm 3.12 is performed using the gantry robot. In Step 4, the generalized Armijo step size (3.44) is applied with $\sigma = 0.1$, $\beta = 0.8$ and $\gamma = 0.03, 0.04, 0.05$ respectively.

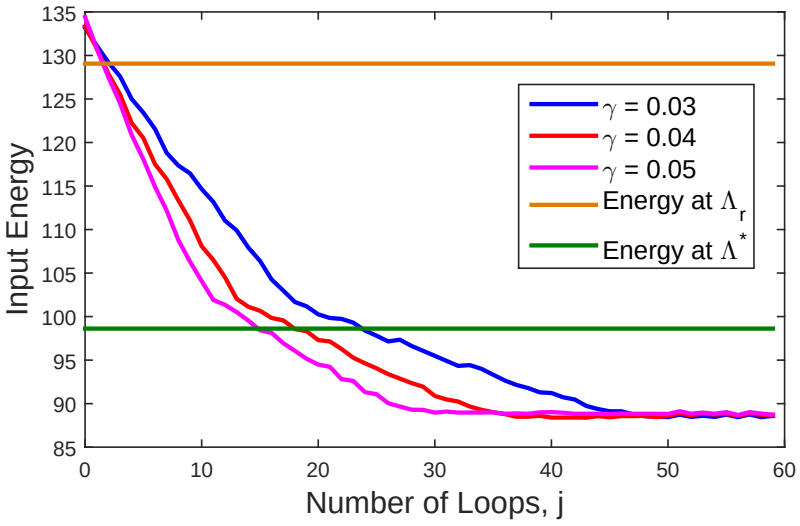


Figure 3.6: Input Energy Results using an Inaccurate Model without Input Constraints.

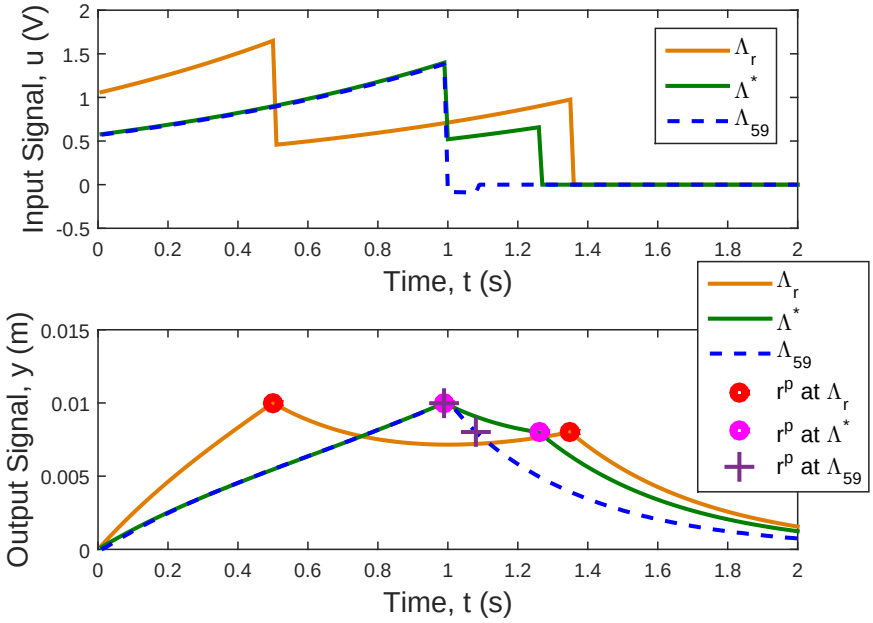


Figure 3.7: Converged Input and Output Trajectory Comparison using an Inaccurate Model without Input Constraints.

The experimental optimal input energy $\tilde{f}(\Lambda_k)$ at each loop is plotted in Figure 3.6 for a step size chosen under different values of γ. The optimal energy $\tilde{f}(\Lambda_r) = 129.06$ required for the gantry robot to track at the *a priori* tracking time allocation is also plotted for comparison. It should be noted that the proposed algorithm provides an experimental final converged energy norm of 88.76, which is a 31% reduction in input energy compared to the operating energy, $\tilde{f}(\Lambda_r)$. This is further compared with normal practice by computing the optimal tracking time allocation in simulation using the nominal model, and then using Stage One update alone to track them experimentally. This yields $\Lambda^* = [0.99, 1.26]^\top$, and $\tilde{f}(\Lambda^*) = 98.62$. It is clear that the experimental final converged energy is approximately 10% less. This means that experimental implementation of Algorithm 3.14 is far superior to optimization using the nominal model in simulation. This confirms that the algorithm displays satisfactory robustness against model uncertainty. Furthermore, the converged input and output trajectories at the original (*a priori*), theoretical optimal and experimental optimal allocations are plotted in Figure 3.7 and illustrate how the experimentally obtained optimal allocation outperforms the other two allocations.

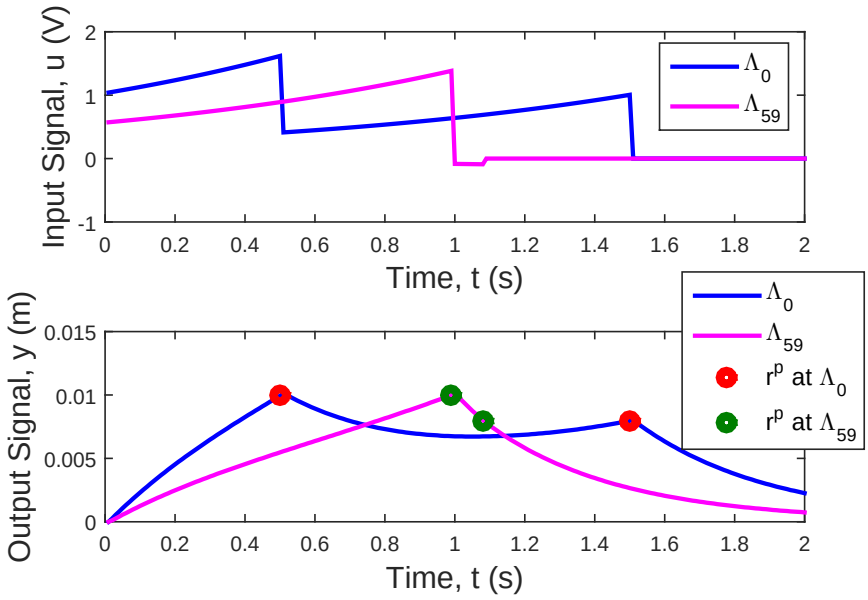


Figure 3.8: Experimental Converged Input and Output Trajectories for Initial and Final Loops using an Inaccurate Model without Input Constraints.

The experimental final converged input and output trajectories for the initial and final loops of the algorithm are compared in Figure 3.8, and it is clear that the input signal immediately becomes zero after finishing tracking the last point in both figures as there is no tracking requirement along the remaining finite time interval. The reference at the special tracking times is marked with red and green circles. It is clear that the final converged output accurately tracks the special tracking points, so the algorithm not only optimizes the input energy but also maintains high tracking performance even with significant model uncertainty. For example, while using the experimental estimation

with a step size chosen to be under $\gamma = 0.03$, the final mean square error is 0.00032mm^2 at Step 6 of the 60^{th} loop.

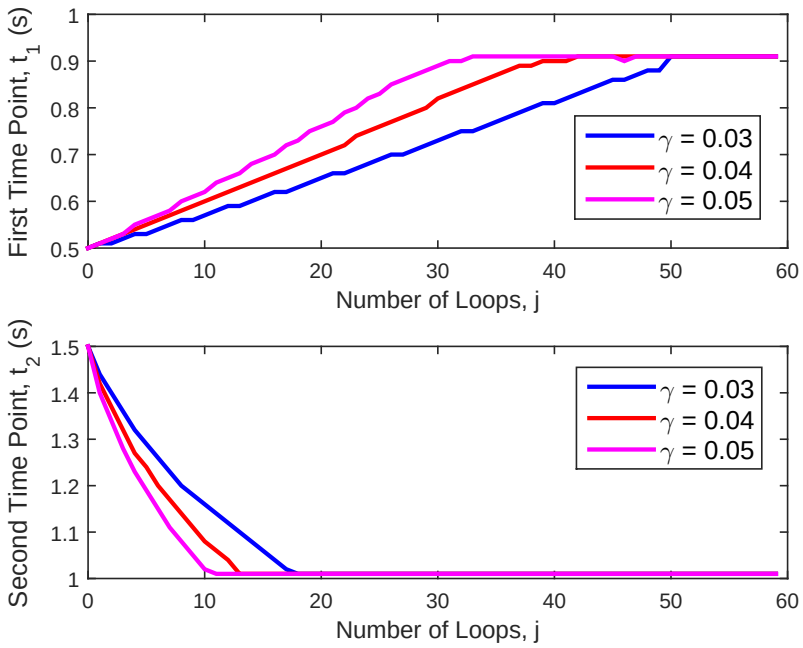


Figure 3.9: Experimental Time-Point Position Results at Each Loop using an Inaccurate Model without Input Constraints.

To further illustrate the performance of Algorithm 3.12, the convergence of the tracking time allocation is shown in Figure 3.9. For each value of γ, the tracking time allocation at each loop is plotted in this figure, and all converge to an identical tracking time allocation of $[0.91, 1.01]^\top$. The automatic tracking time allocation adjustment has meant that the speed of the gantry is slower and the distance the gantry moves is shorter (as can be seen from Figure 3.6), and thus leads to a lower input energy consumption. Experiments using the other initial tracking time allocations, e.g. the low resolution initial tracking time allocation, yield similar levels of performance.

The proposed algorithm with input saturation constraint (2.55) with $M(t) = 1.8$ has also been tested using different system models and parameter choices. A representative result for the input energy convergence is shown in Figure 3.10. From this figure, the optimal energy of 89.53 obtained by the algorithm is 28% less than the energy $\tilde{f}(\Lambda_r) = 129.12$ at the *a priori* allocation and 10% less than the energy $\tilde{f}(\Lambda^*) = 98.51$ at the theoretically obtained optimal allocation using the inaccurate model. This again confirms the superiority of the proposed design method. Furthermore, inspection of the converged input shows that input constraint satisfaction is guaranteed by the proposed algorithm, a typical result of which is show in Figure 3.11 that clearly demonstrates this.

The above tests have also been repeated using the relatively accurate system model shown in (3.83) with a feedback gain of $K = 100$. The results are summarized in

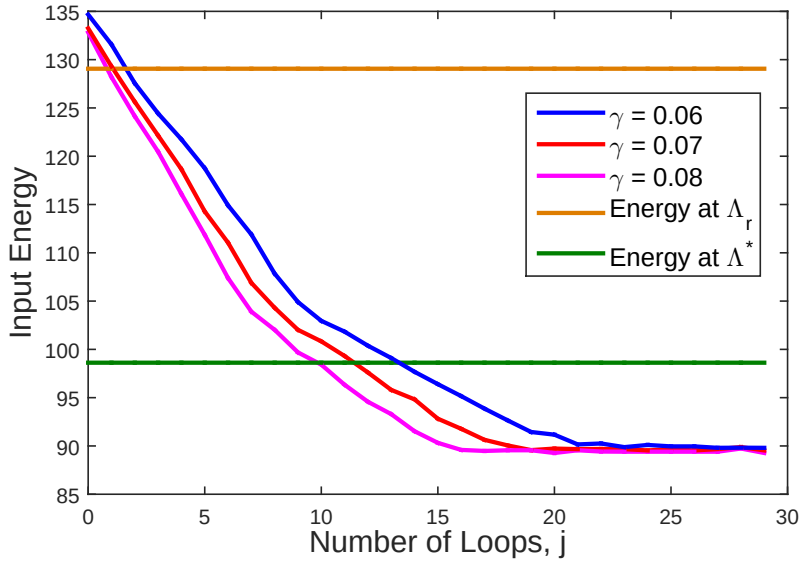


Figure 3.10: Exemplary Input Energy Convergence with Input Constraints.

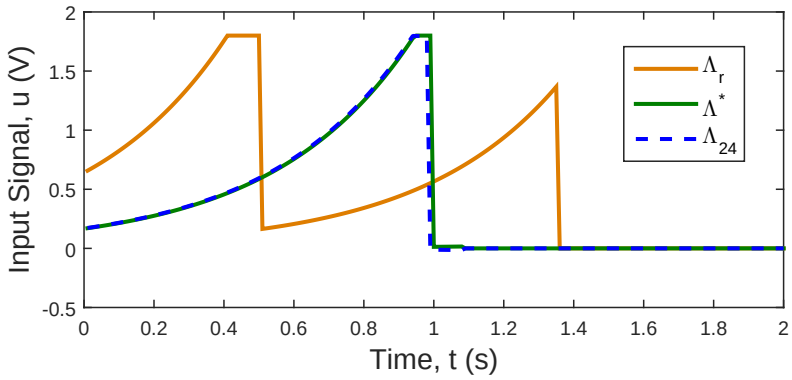


Figure 3.11: Exemplary Converged Input Trajectories with Input Constraints.

Table 3.1: Summary of Experimental Results without Constraints.

	$\gamma = 0.03$	$\gamma = 0.04$	$\gamma = 0.05$
Λ_{49}	$[0.99, 1.09]^\top$	$[0.99, 1.09]^\top$	$[0.99, 1.09]^\top$
$\tilde{f}(\Lambda_{49})$	78.69	78.67	78.66
Reduction from $\tilde{f}(\Lambda_r)$	37.10%	37.11%	37.12%
Difference from $\tilde{f}(\Lambda^*)$	0.63%	0.66%	0.67%
$\|e^p\|^2/2$ at Λ_{49}	0.00051mm^2	0.00006mm^2	0.00026mm^2

Table 3.2: Summary of Experimental Results with Constraints.

	$\gamma = 0.05$	$\gamma = 0.06$	$\gamma = 0.07$
Λ_{24}	$[0.99, 1.08]^\top$	$[0.99, 1.08]^\top$	$[0.99, 1.08]^\top$
$\tilde{f}(\Lambda_{24})$	78.78	79.10	78.17
Reduction from $\tilde{f}(\Lambda_r)$	37.03%	36.77%	37.51%
Difference from $\tilde{f}(\Lambda^*)$	0.52%	0.11%	1.29%
$\|e^p\|^2/2$ at Λ_{24}	0.00013mm^2	0.00002mm^2	0.00003mm^2

Table 3.1 and Table 3.2 for the unconstrained and constrained cases. From the tables, the tracking time allocations obtained by the proposed algorithms all converge, and are close to the corresponding theoretical obtained one, Λ^*, as the system model is relatively accurate. It is clear that the obtained input energy results at each case are approximately 37% less than the *a priori* one, $\tilde{f}(\Lambda_r)$, and are almost the same as the theoretical one, $\tilde{f}(\Lambda^*)$. Also, the tracking errors are within the practical tolerance, confirming that the algorithm not only optimizes the input energy, but also maintains satisfactory tracking performance. The detailed results are omitted here for brevity.

3.6 Summary

Tracking time allocation plays an important role in point-to-point ILC and can significantly affect the system performance. This chapter has developed an optimization framework to fully exploit the flexibility in choosing tracking time allocation to optimize some cost function of interest, in addition to high accuracy reference tracking. The problem has been formulated into an optimization problem in an abstract Hilbert space and a two stage design framework has been developed. A solution to the Stage One design problem has been derived using a well-known norm optimal point-to-point ILC algorithm. For the Stage Two design problem there are no direct analytic solutions of the optimization problem, and an iterative update based on the gradient projection method has therefore been proposed. The solutions of the two stages are combined to yield the first algorithm to solve the high performance point-to-point tracking problem with an additional optimal cost function. The implementation procedures are discussed in detail and the proposed design framework is further extended to embed system constraints into the design.

The proposed algorithm is verified experimentally on a gantry robot test platform. When the system model is inaccurate, significant reduction of the input energy can be achieved. When an accurate system model is available, the input energy converges to the theoretical optimal solution. In both scenarios, the proposed algorithm guarantees high performance tracking. In addition, the proposed Two Stage design framework can also

be extended to handle the same problem discussed in this chapter within the class of discrete time systems with some modification - see Chapter 4 for details.

Chapter 4

Point-to-Point ILC with Optimal Tracking Time Allocation -A Coordinate Descent Approach

Note that the results presented in Chapter 3 employ the gradient method to address the optimal tracking time allocation problem within continuous time systems. However, a number of practical systems incorporate digital processors receiving and sending signals at unit sample rate, and it is impossible to apply the previous algorithms to these systems as the gradient cannot be obtained in discrete time systems. This section uses a coordinate descent approach to address the full optimal tracking time allocation problem for discrete time systems in which dynamic interaction occurs between the critical points.

4.1 Formulation of the Problem

This section first introduces the discrete time system dynamics and defines the corresponding point-to-point ILC framework. Then the design problem of point-to-point ILC with optimal tracking time allocation is formulated into an optimization problem.

4.1.1 Discrete Time System Dynamics

Consider an ℓ-input, m-output discrete linear time-invariant system given in state space form by $S(A, B, C)$ with unit sample time

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t), \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{4.1}$$

where $t \in [0,\ N]$ is the time index (e.g. sample number), $x_k(t) \in \mathbb{R}^n$, $u_k(t) \in \mathbb{R}^\ell$ and $y_k(t) \in \mathbb{R}^m$ are the state, input and output respectively; A, B and C are system matrices of compatible dimension; $0 < N < \infty$ is the trial length, the subscript $k \in \mathbb{N}$ denotes the ILC trial number. At the end of each trial, the state is reset to initial value x_0. The system can be represented in an equivalent operator form

$$y_k = G_d u_k + d \tag{4.2}$$

where $G_d : l_2^\ell[0,\ N] \to l_2^m[0,\ N]$, $y_k,\ d \in l_2^m[0,\ N]$ and $u_k \in l_2^\ell[0,\ N]$, and the input and output Hilbert spaces $l_2^\ell[0,\ T]$ and $l_2^m[0,\ T]$ are defined with inner products and associated induced norms

$$\langle u,\ v\rangle_R = \sum_{i=0}^{N} u^\top(i) R v(i), \quad \|u\|_R = \sqrt{\langle u,\ u\rangle_R} \tag{4.3}$$

$$\langle x,\ y\rangle_S = \sum_{i=0}^{N} x^\top(i) S y(i), \quad \|y\|_S = \sqrt{\langle y,\ y\rangle_S} \tag{4.4}$$

in which $R \in \mathbb{S}_{++}^\ell$ and $S \in \mathbb{S}_{++}^m$. The convolution operator G_d and signal d (representing the effect of initial condition) take the form

$$(G_d u)(t) = \sum_{i=0}^{t-1} C A^{t-i-1} B u(i),\ d(t) = C A^t x_0 \tag{4.5}$$

where, without loss of generality, the constant $d(t)$ can be absorbed into the reference to give $x_0 = 0$, $d(t) = 0$.

4.1.2 Point-to-Point ILC Framework

The point-to-point ILC design objective is to update the input signal, u_k, such that the input signal, u_k, converges to a unique value and the associated output, y_k, ultimately tracks the given reference positions, r_i, $i = 1, \ldots, M$, at a sub set of time instants, t_i, $i = 1, \ldots, M$, i.e.

$$\lim_{k\to\infty} y_k(t_i) = r_i,\ i = 1, \ldots, M, \quad \lim_{k\to\infty} u_k = u^*. \tag{4.6}$$

From the design objective (4.6), only the particular output at the tracking time allocation

$$\Lambda = [t_1,\ t_2, \ldots,\ t_M]^\top \in \Theta \tag{4.7}$$

is of interest where

$$\Theta = \{\Lambda \in \mathbb{R}^M : 0 < t_1 < t_2 < \ldots < t_M \leqslant N\} \tag{4.8}$$

is the admissible set of allocated tracking time instants representing the requirements on enforcing process timing constraints necessary to complete the task. Hence a linear mapping $\zeta \in l_2^m[0,\ N] \mapsto \zeta^p \in H$ defined by

$$\zeta^p = \begin{bmatrix} \zeta(t_1) \\ \vdots \\ \zeta(t_M) \end{bmatrix} \tag{4.9}$$

is introduced to extract the output values at the tracking time allocation. Note that H is the Hilbert space denoted by

$$H = \underbrace{\mathbb{R}^m \times \cdots \times \mathbb{R}^m}_{\text{M times}} \tag{4.10}$$

with inner product and associated induced norm

$$\langle \omega,\ \mu \rangle_{[Q]} = \sum_{i=1}^{M} \omega_i^\top Q_i \mu_i, \quad \|\omega\|_{[Q]} = \sqrt{\langle \omega,\ \omega \rangle_{[Q]}} \tag{4.11}$$

where

$$\omega = [\omega_1, \ldots, \omega_M]^\top \in H,\ \mu = [\mu_1, \ldots, \mu_M]^\top \in H;$$

$[Q]$ denotes the set $\{Q_1, \ldots, Q_M\}$, and $Q_i \in \mathbb{S}_{++}^m$.

From definition (4.9), it follows that the 'point-to-point output', y^p, comprises a subset of plant outputs defined over the tracking time allocation Λ. The dynamics of the point-to-point system can therefore be modelled by

$$y^p = G_\Lambda^p u = (Gu)^p = \begin{bmatrix} (Gu)(t_1) \\ \vdots \\ (Gu)(t_M) \end{bmatrix} \tag{4.12}$$

where $G_\Lambda^p : l_2^\ell[0,\ N] \to H$ is a linear operator.

Therefore, the point-to-point ILC design objective design objective (4.6) can be equivalently described as iteratively finding a sequence of input $\{u_k\}$ such that

$$\lim_{k\to\infty} y_k^p = r^p, \quad \lim_{k\to\infty} u_k = u^* \tag{4.13}$$

where

$$r^p = [r_1,\ r_2, \ldots,\ r_M]^\top \in H \tag{4.14}$$

To solve the problem, the point-to-point tracking error, $e_k^p = r^p - y_k^p$, is employed within the following updating law:

$$u_{k+1} = \mathcal{F}(u_k, e_k^p) \tag{4.15}$$

where $\mathcal{F}$ is an updating function involving the previous input and point-to-point tracking error.

4.1.3 Optimal Tracking Time Allocation Problem

The **Point-to-Point ILC with Optimal Tracking Time Allocation Problem** in discrete time case can be addressed by proposing a design framework which automatically provides a tracking time allocation to optimize some desired cost functions, and meanwhile ensures high performance tracking at the tracking time allocation.

The problem design objective is to iteratively find a tracking time allocation, Λ_k, and an input, u_k, with the asymptotic property that the output values, y_k^p, at the tracking time allocation accurately pass through a set of points, r^p, i.e.

$$\lim_{k\to\infty} y_k^p = r^p,$$

at the same time optimizing a target cost function $f(u, y)$ with respect to the system input, u, and output, y, i.e.

$$\lim_{k\to\infty} (u_k,\ y_k,\ \Lambda_k) = (u_k^*,\ y_k^*,\ \Lambda_k^*)$$

where u_k^*, y_k^* and Λ_k^* are optimal solutions of the problem

$$\begin{aligned} &\underset{u,y,\Lambda}{\text{minimize}} && f(u, y) \\ &\text{subject to} && r^p = G_\Lambda^p u,\ y = Gu,\ \Lambda \in \Theta. \end{aligned} \tag{4.16}$$

4.2 A Two Stage Design Framework

While the tracking time allocation Λ does not appear in the performance function $f(u, y)$, they are connected by the constraint, $G_\Lambda^p u = r^p$, making the problem (4.16) non-trivial. In this section, the Two Stage design framework proposed in Chapter 3 is used to solve this optimization problem.

4.2.1 Framework Description

Optimization problem (4.16) can be equivalently written as

$$\min_{\Lambda\in\Theta}\{\min_u f(u, y), \text{subject to } r^p = G_\Lambda^p u,\ y = Gu\} \tag{4.17}$$

which optimizes over u first and then optimizes over Λ. Define the function $\tilde{f}(\Lambda) : \mathbb{R}^M \to \mathbb{R}$ by

$$\tilde{f}(\Lambda) = \{\min_u f(u, y), \text{subject to } r^p = G^p_\Lambda u,\ y = Gu\} \tag{4.18}$$

and denote a global minimizer for u of the inner optimization problem as $u_\infty(\Lambda) : \mathbb{R}^M \to l_2^\ell[0,\ N]$. Hence the problem (4.17) can be equivalently written as

$$\min_{\Lambda\in\Theta}\{\tilde{f}(\Lambda) := f(u_\infty(\Lambda))\}. \tag{4.19}$$

The above suggests that the optimization problem (4.16) can be solved by applying a two stage design framework as:

- *Stage One*: Keep the tracking time allocation, Λ, fixed and solve the optimization problem
$$\min_u f(u, y),\ \text{subject to } r^p = G^p_\Lambda u,\ y = Gu. \tag{4.20}$$
- *Stage Two*: Substitute the solution, $u_\infty(\Lambda)$, into the original problem and then find the optimal solution
$$\min_{\Lambda\in\Theta}\{\tilde{f}(\Lambda) := f(u_\infty(\Lambda))\}. \tag{4.21}$$

In this appendix, the control effort is selected to be the target cost function to exemplify the approach, i.e. $f(u, y) = \|u\|^2_R$. This guarantees the existence of a unique global minimizer for u within (4.20).

4.2.2 Solution of the Proposed Framework

1). Solution of Stage One: For a given Λ, Stage One is a point-to-point ILC design problem with minimum control effort requirement. This can be solved using a norm optimal point-to-point ILC algorithm as shown in the next theorem.

Theorem 4.1. *If the system $S(A, B, C)$ is controllable and C has full row rank, the solution of Stage One problem (4.20) is given by u_∞, which can be found using the norm-optimal point-to-point ILC algorithm*

$$u_{k+1} = u_k + G^{p*}_\Lambda (I + G^p_\Lambda G^{p*}_\Lambda)^{-1} e^p_k \tag{4.22}$$

proposed in Owens et al. (2013) with initial input $u_0 = 0$ such that

$$u_\infty(\Lambda) = \lim_{k\to\infty} u_k. \tag{4.23}$$

Furthermore, an analytic solution can be obtained for $u_\infty(\Lambda)$ *as follows:*

$$u_\infty(\Lambda) = G_\Lambda^{p*}(G_\Lambda^p G_\Lambda^{p*})^{-1} r^p. \tag{4.24}$$

Note that $G_\Lambda^{p*} : \omega \in H \to u \in l_2^\ell[0,\ N]$ *is the Hilbert adjoint operator of* G_Λ^p *given by*

$$\begin{aligned} &u(t) = R^{-1}B^\top p(t), \\ &p(t) = A^\top p(t+1), \\ &p(N) = 0, \\ &p(t_i - 1) = p(t_i) + C^\top Q_i \omega_i,\ i = 1, \dots, M. \end{aligned} \tag{4.25}$$

Proof. The proof follows from the proof of Theorem 3.3. □

Remark 4.2. Note that the system's state controllable condition is not restrictive as a state controllable model can always be constructed for a given system.

2). Solution of Stage Two: Similar to Lemma 3.4, the Stage Two optimization problem (4.21) can be expressed as

$$\min_{\Lambda\in\Theta} \|u_\infty(\Lambda)\|_R^2 = \min_{\Lambda\in\Theta} \left\langle r^p,\ (G_\Lambda^p G_\Lambda^{p*})^{-1} r^p \right\rangle_{[Q]}. \tag{4.26}$$

This optimization problem, however, is non-trivial except for the special case of $M = 1$, i.e. there is only one tracking point, where the solution can be obtained analytically, as shown in the following theorem.

Theorem 4.3. *When there is only one tracking point, optimization problem (4.26) has analytical solution*

$$\Lambda^* = N.$$

The corresponding minimum energy is

$$\min_{\Lambda\in\Theta} \|u_\infty(\Lambda)\|_R^2 = \left\langle r^p,\ \Psi_N^{-1} r^p \right\rangle_{[Q]}$$

where

$$\Psi_t = \sum_{i=1}^{t} CA^{t-i}BR^{-1}(CA^{t-i}B)^\top.$$

Proof. The proof follows from the proof of Theorem 3.5. □

Theorem 4.3 shows that when $M = 1$, $\Lambda^* = N$ is always the optimal choice in terms of minimizing the control input energy - this is not surprising as this allows the system output to change gradually to the desired position. However, when $M > 1$, i.e. there is more than one tracking point, the cost function is generally non-linear and non-convex

with respect to the time point set, Λ. These aspects lead to the difficulties in obtaining an analytical solution of the problem (4.26) at general cases.

In the case of discrete time systems, however, the gradient of the cost function in 4.26 cannot obtained. The gradient projection method cannot be used to solve the Stage Two problem. It is noticed that the set, Θ, has a finite number of elements and this permits the use of a blind search over the whole set. However, this carries a high computational load especially when the number of tracking points M becomes large. Therefore, an efficient alternative method is proposed in the next theorem to give an approximate solution of the Stage Two problem (4.26).

Theorem 4.4. *Consider the coordinate descent method with initial estimate* Λ_0

$$\Lambda_{j+1} = \mathcal{C}(\Lambda_j) \tag{4.27}$$

with $\Lambda_j = [t_1^j, t_2^j, \ldots, t_M^j]^\top$, $j \in \mathbb{N}$ *denotes the coordinate descent trial number and each time point is updated by the function* $\mathcal{C}$ *as*

$$t_{i,j+1} = \begin{cases} t_i^{j*}, & i = (j+1) \bmod M \\ t_i^j, & else \end{cases} \tag{4.28}$$

where t_i^{j*} *is the optimizer of the optimization problem*

$$\begin{aligned} \underset{t}{minimize} \quad & \left\langle r^p, \ (G_\Lambda^p G_\Lambda^{p*})^{-1} r^p \right\rangle_{[Q]} \\ subject\ to \quad & \Lambda = [t_1^j, \ldots, \ t_{i-1}^j, \ t, \ t_{i+1}^j, \ldots, \ t_M^j]^\top, \\ & t \in (t_{i-1}^j, \ t_{i+1}^j). \end{aligned} \tag{4.29}$$

The sequence, $\{\tilde{f}(\Lambda_j)\}$*, based on the coordinate descent update (4.27) converges downward to a limit* $\tilde{f}^*$*.*

Proof. The coordinate descent method divides the optimization problem (4.26) into a number of intermediate trials. At each trial, it performs a local optimization (4.29) to update a single time point $t_{i,j}$ with other points keeping the same values. Therefore, it follows that the sequence $\{\tilde{f}(\Lambda_j)\}$ monotonically decreases as

$$\tilde{f}(\Lambda_{j+1}) \leqslant \tilde{f}(\Lambda_j). \tag{4.30}$$

In addition, as the function $\tilde{f}(\Lambda)$ is bounded below, the sequence, $\{\tilde{f}(\Lambda_j)\}$, converges to a non-negative value $\tilde{f}^*$. □

Remark 4.5. Although blind search is still used in each coordinate descent trial to update a single time point, t_i^j, the coordinate descent method requires much less computation time than the pure blind search method over the whole set, Θ, to provide an optimal solution to the same optimization problem. This is because of the desirable

property of the coordinate descent method which splits the original problem into several sub-problems.

Remark 4.6. In most optimization problems the coordinate descent method does not guarantee a global optimal solution, but yields a suitable local optimal solution which approximates the global optimal solution Cormen et al. (2009).

Remark 4.7. Gradient method can be applied to problem (4.26) in the case of continuous time systems as shown in Chapter 3. However, the gradient is unavailable in discrete time systems, so this approach is infeasible.

Remark 4.8. The coordinate descent method described in (4.27) updates a single time instant at each trial, however, multiply number of time instants can be also considered for updating.

Remark 4.9. As the solution of the coordinate descent method is a local solution, it is necessary to choose a suitable initial tracking time allocation, Λ_0, to give a solution which approximates the global one, especially when the system is complex. If no information is available, Λ_0 can be chosen arbitrarily, and alternatively it can be selected using different methods such as performing grid search with a large sample time.

4.3 Implementation of the Design Approach

This section develops an algorithm to efficiently implement the two stage framework.

4.3.1 Implementation of Stage One

The general solution of Stage One using (4.22) can be either computed directly using the analytical solution (4.24) or implemented experimentally using a combined feedback and feedforward solution illustrated in the next proposition.

Proposition 4.10. *The ILC update (4.22) can be implemented using the feedforward plus feedback implementation*

$$u_{k+1}(t) = u_k(t) + R^{-1}B^\top p_k(t),\ t = 0, \dots, N \tag{4.31}$$

with

$$p_k(t) = -K(t)(I + BR^{-1}B^\top K(t))^{-1}A(x_{k+1}(t) - x_k(t)) + \xi_{k+1}(t) \tag{4.32}$$

where $K(t)$ denotes the Riccati feedback matrix

$$\begin{aligned} &K(t) = A^\top K(t+1)(I + BR^{-1}B^\top K(t+1))^{-1}A, \\ &K(N) = 0,\ K(\tilde{t}_i^-) = K(\tilde{t}_i^+) + C^\top Q_i C \end{aligned} \tag{4.33}$$

and $\xi_{k+1}(t)$ *denotes the predictive feedforward term at the* $(k+1)^{th}$ *ILC trial*

$$\begin{aligned}&\xi_{k+1}(t) = (I + K(t)BR^{-1}B^\top)^{-1}A^\top\xi_{k+1}(t+1),\\&\xi_{k+1}(N) = 0,\ \xi_{k+1}(\tilde{t}_i^-) = \xi_{k+1}(\tilde{t}_i^+) + C^\top Q_i e_k(t_i).\end{aligned} \tag{4.34}$$

Proof. The proof follows from the proof of Proposition 3.9. □

The experimental implementation of norm-optimal ILC using feedback and feedforward solution provides an optimal solution to Stage One problem (4.20) based on real plant dynamics. Due to the real time state feedback at the current ILC trial, this implementation method has a certain degree of robustness against model uncertainties.

4.3.2 Implementation of Stage Two

The coordinate descent method introduced in Theorem 4.4 re-arranges the tracking time allocation to minimize the control effort. It starts from an initial tracking time allocation

$$\Lambda_0 = [t_1^0, t_2^0, \dots,\ t_M^0]^T \in \Theta. \tag{4.35}$$

At each coordinate descent trial, it only updates a single time point, t_i^j, by solving the optimization problem (4.29). This is equivalent to finding the optimal element along the finite interval $(t_{i-1}^j,\ t_{i+1}^j)$ with respect to a cost function. Therefore, blind search methods can be applied to this problem with a total computation number $\eta_i = (t_{i+1}^j - t_{i-1}^j - 1)$.

Remark 4.11. As the solution of the coordinate descent method is a local solution, it is necessary to choose a suitable initial tracking time allocation, Λ_0, to give a solution which approximates the global one, especially when the system is complex. If no information is available, Λ_0 can be chosen arbitrarily, and alternatively it can be selected using different methods such as performing grid search with a large sample time.

4.3.3 An iterative implementation algorithm

Combining the implementation of Stage One and Stage Two designs leads to an iterative implementation of the two stage design framework - Algorithm 4.12. Note that Λ_0 is a suitably chosen initial tracking time allocation, and $\epsilon > 0$, $\delta > 0$ are small scalars which depend on the tracking precision requirement and performance requirement, respectively.

Algorithm 4.12. Given initial allocation, Λ_0, system state space model, $S(A, B, C)$, desired tracking reference, r^p, admissible set of tracking time allocation, Θ, weighting

matrices, R and Q_i, the followings steps provide the solution to the optimal tracking time allocation, Λ_{opt}, and input, u_{opt}, i.e.

1: **initialization:** Coordinate descent trial number $j = 0$
2: Implement Stage One update (4.22) with $\Lambda = \Lambda_0$ experimentally until convergence, i.e. $\|e_k^p\| < \epsilon\|r^p\|$; record converged input, $u_\infty^{ex}(\Lambda_0)$, and input energy, $\tilde{f}(\Lambda_0)$.
3: **repeat**
4: Implement Stage Two update (4.27) with $r^p = G_{\Lambda_j}^p u_\infty^{ex}(\Lambda_j)$.
5: Set $j \to j + 1$.
6: Implement Stage One update (4.22) with $\Lambda = \Lambda_j$ experimentally until convergence, i.e. $\|e_k^p\| < \epsilon\|r^p\|$; record converged input, $u_\infty^{ex}(\Lambda_j)$, and input energy, $\tilde{f}(\Lambda_j)$.
7: **until** $\left|\tilde{f}(\Lambda_j) - \tilde{f}(\Lambda_{j-1})\right| < \delta \left|\tilde{f}(\Lambda_{j-1})\right|$
8: **return** $\Lambda_{opt} = \Lambda_j$ and $u_{opt} = u_\infty^{ex}(\Lambda_j)$

In Algorithm 4.12, Step 2 and 6 (i.e. norm-optimal ILC algorithm) are required to be implemented experimentally and Step 4 uses real data, $u_\infty(\Lambda_j)$, obtained from experiments. These requirements are not necessary when an accurate system model is known. However when there exists model mismatches/uncertainties, the proposed algorithm will have attractive robustness properties as the algorithm 'learns' information about the real plant dynamics through use of experimental data. This will be demonstrated using experimental results in the next section.

4.4 Experimental Verification on a Gantry Robot

In this section, the proposed algorithm is validated experimentally on a three-axis gantry robot test platform to demonstrate its effectiveness. Consider the multi-axis gantry robot shown in Chapter 3.5.1 as the test platform. The control design objective is to use the z-axis ($m = 1$) to perform a point-to-point ILC tracking task during the given tracking time $T = 2s$ with only five special tracking points ($M = 5$) given as

$$r^p = [0.0048,\ 0.0029,\ -0.0029,\ -0.0048,\ 0]^\top \tag{4.36}$$

which are shown in Figure 4.3. The *a priori* tracking time allocation is given as $\Lambda_r = [20, 60, 100, 140, 180]^\top$. Again, it is assumed that only an approximate model (3.85) is available for the z-axis as follows with a 150 proportional feedback controller which is sampled with a zero-order hold at $0.01s$.

A 30 coordinate descent trial updating procedure of Algorithm 4.12 is performed on the gantry robot platform with initial tracking time allocation $\Lambda_0 = \Lambda_r$. For implementational simplicity, the weighting matrices in Step 2 and 6 are taken as $Q_i = qI$ and $R = rI$

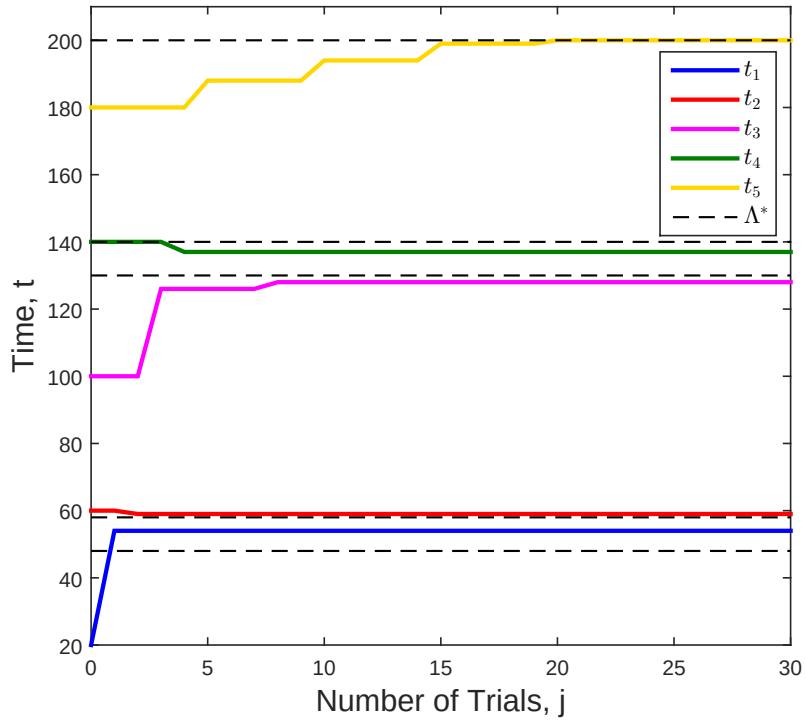


Figure 4.1: Experimental Time-Point Position Results at Each Trial.

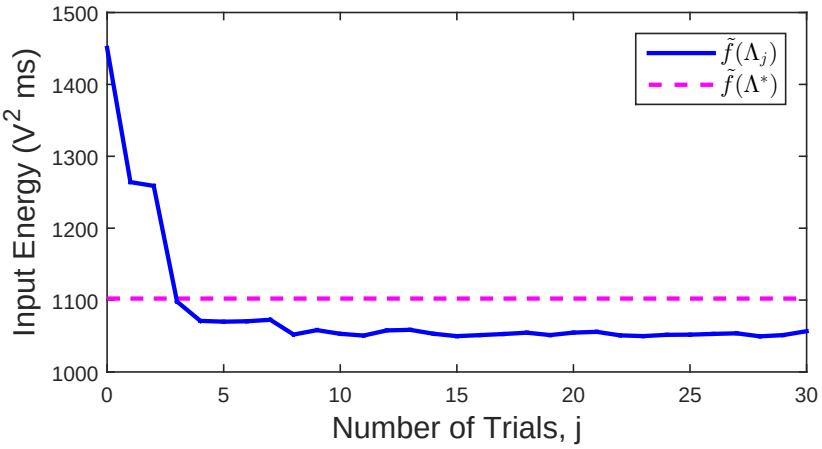


Figure 4.2: Experimental Input Energy Results at Each Trial.

where q and r are positive scalars which satisfy $q/r = 1,000,000$. The tracking time allocation at each coordinate descent trial is plotted in Figure 4.1, and it is clear from the figure that an experimental optimal tracking time allocation $\Lambda_{30} = [54, 59, 128, 137, 200]^\top$ is obtained.

The input energy $\tilde{f}(\Lambda_j)$ at each trial is plotted in Figure 4.2, which shows that the input energy converges to a limit energy $\tilde{f}(\Lambda_{30}) = 1053.4$ along the trial. From the figure, the limit input energy $\tilde{f}(\Lambda_{30})$ is 27.4% less than the input energy $\tilde{f}(\Lambda_r)$ at Λ_r, which confirms

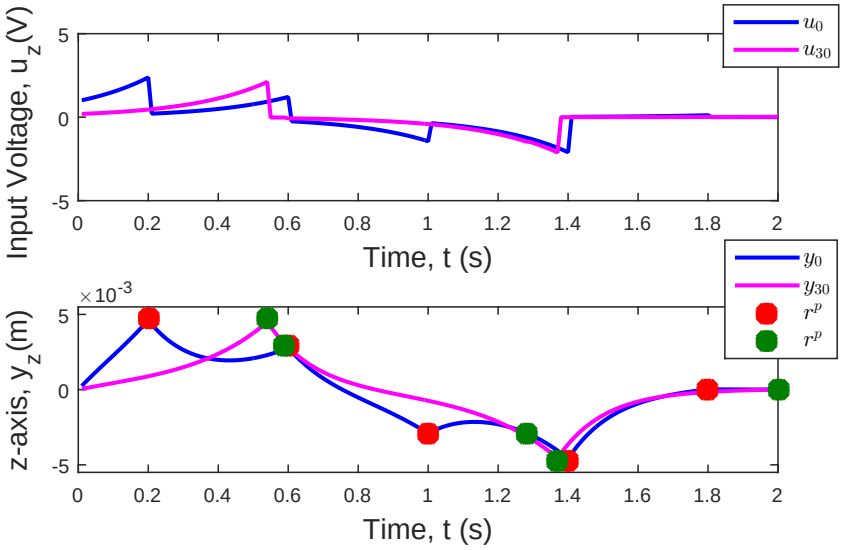


Figure 4.3: Experimental Converged Input and Output Trajectories for Initial and Final Trials.

the robustness of the algorithm against model uncertainties. The theoretical optimal tracking time allocation $\Lambda^* = [48, 58, 130, 140, 200]^\top$ is computed in simulation using the system model, and plotted as the dashed black lines in Figure 4.1 for comparison. The corresponding operation energy $\tilde{f}(\Lambda^*) = 1102.1$ at Λ^* is also plotted in Figure 4.2 as the dashed magenta line. Note that the experimental optimal tracking time allocation Λ_{30} is slightly different from the theoretically obtained Λ^*, and the experimental energy solution $\tilde{f}(\Lambda_{30})$ is 4.4% less than the theoretical one $\tilde{f}(\Lambda^*)$, which demonstrates the advantage of implementing the algorithm experimentally rather than using the nominal model in simulation.

The experimental final converged results for the initial and final trials are compared in Figure 4.3 with the reference, r^p, marked as red and green dots. It is obvious that the final converged output perform perfect point-to-point tracking at the critical time points, e.g. the mean square error is 0.00053 at the final trial. So the algorithm not only optimizes the input energy but also maintains high tracking performance.

Experiments with other initial tracking time allocations and other R_i, Q values provide similar convergence performance to the results in Figure 4.2. For brevity, these results are omitted.

4.5 Summary

This appendix exploits the flexibility of choosing the tracking time allocation in the point-to-point ILC framework within the discrete time case, which affects system performance. An optimization problem is formulated, and the Two Stage design framework

proposed in Chapter 3 is used to solve this problem with minimum control effort. Distinct to the work in Chapter 3, this problem setup is suitable for discrete time systems. Stage One solution is obtained via a norm optimal point-to-point ILC algorithm, and Stage Two solution is computed using a coordinate descent method. The solutions yield an iterative algorithm, which is verified on a gantry robot test platform, whose results reveals practical efficacy.

In the problem setup of Chapter 3, only a finite number of critical positions have been considered, however the ideas in this chapter can be further extended to optimize the tracking time allocation of all the positions along a given path. This extension naturally meets the task description of spatial ILC, whose design objective is to follow a given path defined in space with no *a priori* temporal constraints. The details of this extension will be developed in the next two chapters.

Chapter 5

Generalized ILC with Application to Spatial Path Tracking

In order to extend the idea of optimizing the tracking time allocation proposed in the previous chapter to address the class of spatial ILC tasks, the ILC framework must be generalized to wide classes of tracking tasks. In addition, the framework must be further expanded by introducing a range of system constraints that have significant relevance to industrial manufacture, e.g. input constraints represent the limit of the system's input load and output constraints represent the boundary of the acceptable moving space preventing potential overshoot.

To do this, this chapter first formulates a generalized ILC problem embedding both intermediate point tracking at time instants, t_i, and linear constraints between outputs on defined sub-intervals, $[t_{i-1},\ t_i]$, enforcing tracking along lines or planes with no *a priori* timing constraints. It then incorporate a mixed form of system constraints to the proposed framework. The successive projection method is used to design a control algorithm which can be efficiently implemented in practice and automatically provides the solution of the generalized ILC problem. In particular, this algorithm can be applied to spatial path tracking problem by setting the coefficients appropriately.

5.1 Problem Formulation

This section introduces the system dynamics and defines the general tracking requirement. Then input and output constraints are incorporated to yield a generalized ILC problem formulation.

5.1.1 Generalized ILC Design Objective

A generalized control design objective is described next. In traditional ILC the system is required to repeatedly track a desired reference defined over the whole horizon; in point to point ILC the system is required to track a given reference defined on a finite set of intermediate points. The general control design objective subsumes both the intermediate point tracking requirement at time instants, t_i, $i = 1, \ldots, M$, where

$$0 < t_1 < \cdots < t_M = T, \tag{5.1}$$

and the linear tracking requirement at each sub-interval, $[t_{i-1},\ t_i]$, $i = 1, \ldots, M$, where $t_0 = 0$ is used for ease of notation. To extract the intermediate point and sub-interval tracking requirements, a linear mapping is defined as

$$\zeta \in L_2^m[0,\ T] \mapsto \zeta^e \in \breve{H} : \zeta^e = \begin{bmatrix} F\zeta \\ P\zeta \end{bmatrix}. \tag{5.2}$$

where $\breve{H}$ is the Hilbert space defined as

$$\breve{H} = \mathbb{R}^{f_1} \times \cdots \times \mathbb{R}^{f_M} \times L_2^{p_1}[t_0,\ t_1] \times \cdots \times L_2^{p_M}[t_{M-1},\ t_M]$$

with inner product and associated induced norm

$$\begin{aligned} \langle (\omega, \nu),\ (\mu, \lambda) \rangle_{\tilde{Q}} &= \sum_{i=1}^{M} \{ \omega_i^\top Q_i \mu_i + \sum_{j=t_{i-1}}^{t_i} \nu_i^\top(j) \hat{Q}_i \lambda_i(j) \}, \\ \|(\omega, \nu)\|_{\tilde{Q}} &= \sqrt{\langle (\omega, \nu), (\omega, \nu) \rangle_{\tilde{Q}}}, \end{aligned} \tag{5.3}$$

in which $(\omega, \nu),\ (\mu, \lambda) \in \breve{H}$ have the following forms

$$\begin{aligned} &\omega = (\omega_1,\ \omega_2, \ldots,\ \omega_M),\ \mu = (\mu_1,\ \mu_2, \ldots,\ \mu_M), \\ &\nu = (\nu_1,\ \nu_2, \ldots,\ \nu_M),\ \lambda = (\lambda_1,\ \lambda_2, \ldots,\ \lambda_M), \end{aligned} \tag{5.4}$$

where $\omega_i,\ \mu_i \in \mathbb{R}^{f_i},\ \nu_i,\ \lambda_i \in L_2^{p_i}[t_{i-1},\ t_i],\ i = 1, \ldots, M$ and $\tilde{Q}$ denotes the data set $\{Q_1, \ldots, Q_M,\ \hat{Q}_1, \ldots, \hat{Q}_M\}$ in which $Q_i \in \mathbb{S}_{++}^{f_i}$, $\hat{Q}_i \in \mathbb{S}_{++}^{p_i}$, for $i = 1, \ldots, M$.

In the mapping defined above, the operator F selects the important elements or linear combination of elements of ζ at the intermediate time instants, t_i, $i = 1, \ldots, M$, and is defined as

$$F\zeta = \begin{bmatrix} F_1\zeta(t_1) \\ \vdots \\ F_M\zeta(t_M) \end{bmatrix}, \tag{5.5}$$

where

$$F_i\zeta(t_i) \in \mathbb{R}^{f_i}$$

with $F_i \in \mathbb{R}^{f_i \times m}$ a full row rank matrix for $i = 1, \ldots, M$. The operator P extracts a linear combination of elements of ζ at each sub-interval, $[t_{i-1},\ t_i]$, $i = 1, \ldots, M$, as follows

$$P\zeta = \begin{bmatrix} (P\zeta)_1 \\ \vdots \\ (P\zeta)_M \end{bmatrix}, \tag{5.6}$$

where

$$(P\zeta)_i \in L_2^{p_i}[t_{i-1},\ t_i],$$

is defined as

$$(P\zeta)_i(t) = P_i\zeta(t),\ t \in [t_{i-1},\ t_i],$$

in which $P_i \in \mathbb{R}^{p_i \times m}$ is a full row rank matrix for $i = 1, \ldots, M$.

From definitions (5.5) and (5.6), it follows that the 'extended output' y^e comprises a subset of outputs at distinct intermediate points, together with a subset of plant outputs defined over sub-intervals of the task duration. The dynamic relationship between the system input u and the extended output y^e can therefore be modelled by

$$y^e = G_\Lambda^e u = (Gu)^e = \begin{bmatrix} FGu \\ PGu \end{bmatrix} \tag{5.7}$$

where $G_\Lambda^e : L_2^\ell[0,\ T] \to H$ is a linear operator with its subscript denoting dependence on the tracking time allocation, Λ, of transition positions defined as

$$\Lambda = [t_1, \ldots, t_M]^\top. \tag{5.8}$$

For the generalized control design, the system is required to meet the tracking requirement that the system extended output, y^e, (repeatedly) follows a desired reference, $r^e \in H$, i.e. $y^e = r^e$.

Remark 5.1. It is worth mentioning that as a general and powerful ILC design framework, the generalized ILC framework collapses to most types of ILC framework previously considered by setting appropriate values of parameters Q, $\hat{Q}$, F and P, e.g. $Q_i = 0$, $P_i = I$, classical ILC; $\hat{Q}_i = 0$, $F_i = I$, point-to-point ILC.

5.1.2 Input and Output Constraints

In practice, input and output constraints exist widely in control systems due to physical limitations or performance requirements. For example, the input constraint set, Ω, typically assumes one of the following forms:

- Input saturation constraint

$$\Omega = \{u \in L_2^\ell[0,\ T] : |u(t)| \preceq M(t),\ 0 \leqslant t \leqslant T\}, \tag{5.9}$$

  where $M(t) \succeq 0, 0 \leqslant t \leqslant N$ are the (possible time varying) saturation limits,

- Input energy constraint

$$\Omega = \{u \in L_2^\ell[0,\ T] : \sum_0^N u^\top(t)u(t) \leqslant M\}, \tag{5.10}$$

where $M > 0$ is the total energy limit. Similarly the output constraint set, Φ, usually has the forms:

- Output saturation constraint

$$\Phi = \{y \in L_2^m[0,\ T] : |y(t)| \preceq N(t),\ 0 \leqslant t \leqslant T\}, \tag{5.11}$$

  where $N(t) > 0$ represent the output saturation limit,

- Output polyhedral constraint

$$\Phi = \{y \in L_2^m[0,\ T] : a_i^\top y(t) \leqslant b_i,\ a_i \in \mathbb{R}^m,\ b_i \in \mathbb{R},\ i = 1, \ldots, M,\ 0 \leqslant t \leqslant T\}. \tag{5.12}$$

In particular, the latter constraint restricts the system output to a specified convex region, and can be used to solve the potential overshoot problem.

5.1.3 Generalized ILC Design problem

Using the extended output (5.7) combined with the above constraints, the generalized ILC design problem can be stated clearly.

The generalized ILC design problem is to find an input updating law based on a function of the previous trial's input and tracking error in the following form

$$u_{k+1} = \mathcal{F}(u_k, e_k^e) \tag{5.13}$$

where $e_k^e = r^e - y_k^e$ is the extended tracking error such that the tracking error converges to zero as $k \to \infty$, i.e.

$$\lim_{k\to\infty} e_k^e = 0$$

and that the converged input and output satisfy the constraints, i.e.

$$\lim_{k\to\infty} u_k = u^* \in \Omega,\ \lim_{k\to\infty} y_k = y^* \in \Phi. \tag{5.14}$$

5.2 Generalized ILC using Successive Projection

In this section, the above generalized ILC design problem is formulated using the successive projection framework which was used previously to derive classical and point-to-point ILC algorithms (Chu and Owens, 2010; Chu et al., 2015). Based on this formulation, a novel ILC algorithm is proposed to solve the generalized ILC design problem.

5.2.1 Successive Projection Interpretation

The design objective of the 'generalized' ILC problem is to iteratively find an input, u^*, such that: i) the extended output, $y^{e*} = G^e u^*$, tracks the desired reference, r^e, i.e. $y^{e*} = r^e$, ii) the output, $y^* = Gu^*$, satisfies the constraint, i.e. $y^* \in \Phi$ and iii) the input u^* meets the constraint requirement, i.e. $u^* \in \Omega$. This is equivalent to iteratively finding a point (y^{e*}, y^*, u^*) in the intersection of the two following convex sets:

$$S_1 = \{(y^e, y, u) \in \hat{H} : y^e = G^e u,\ y = Gu\} \tag{5.15}$$

$$S_2 = \{(y^e, y, u) \in \hat{H} : y^e = r^e,\ y \in \Phi,\ u \in \Omega\} \tag{5.16}$$

where the set S_1 represent the plant dynamics and S_2 represents the tracking requirements and system constraints; $\hat{H}$ is the Hilbert space defined by

$$\hat{H} = \mathbb{R}^{f_1} \times \cdots \times \mathbb{R}^{f_M} \times L_2^{p_1}[t_0,\ t_1] \times \cdots \times L_2^{p_M}[t_{M-1},\ t_M] \times L_2^m[0,\ T] \times L_2^\ell[0,\ T] \tag{5.17}$$

whose inner product and associated induced norm are derived naturally from (3.3), (3.4) and (5.3).

The problem of finding a point in the intersection of two sets can be solved by the method of successive projection. The basic successive projection scheme from Owens and Jones (1978) is shown in Figure 5.1 with guaranteed convergence performance as shown in the following theorem.

Theorem 5.2. *Owens and Jones (1978); Gubin et al. (1967). Let S_1 and S_2 be two closed convex sets in a Hilbert space X. Define projection operators $P_{S_1}(\cdot)$ and $P_{S_2}(\cdot)$ as*

$$P_{S_1}(x) = \arg\min_{\hat{x} \in S_1} \|\hat{x} - x\|_X^2 , \tag{5.18}$$

$$P_{S_2}(x) = \arg\min_{\hat{x} \in S_2} \|\hat{x} - x\|_X^2 , \tag{5.19}$$

where $\|\cdot\|$ is the induced norm in X. Then given the initial estimate $x_0 \in X$, the sequences $\{\tilde{x}_k\}_{k \geqslant 0}$ and $\{x_k\}_{k \geqslant 0}$ generated by

$$\tilde{x}_{k+1} = P_{S_1}(x_k),\ x_{k+1} = P_{S_2}(\tilde{x}_{k+1}),\ k \geqslant 0 \tag{5.20}$$

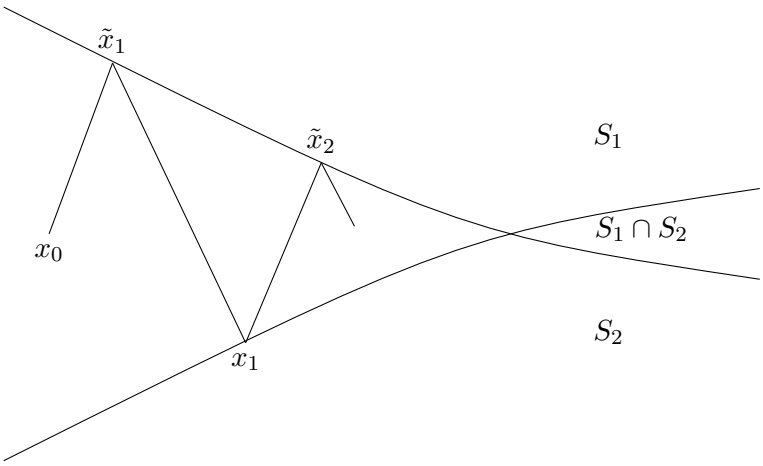


Figure 5.1: Illustration of the Successive Projection Algorithm.

are uniquely defined for each $x_0 \in X$ *and satisfy the following monotonic convergence conditions*

$$\|\tilde{x}_{k+2} - x_{k+1}\|_X^2 \leqslant \|\tilde{x}_{k+1} - x_k\|_X^2 . \tag{5.21}$$

For each $\epsilon > 0$*, there exists an integer* N *such that for* $k > N$

$$\|\tilde{x}_{k+1} - x_k\|_X^2 < \epsilon. \tag{5.22}$$

and the minimum distance between two sets is guaranteed, i.e.

$$\lim_{k\to\infty} \|\tilde{x}_k - x_k\|_X^2 = \inf_{\tilde{x}\in S_1, x\in S_2} \|\tilde{x} - x\|_X^2 . \tag{5.23}$$

Furthermore, if $S_1 \cap S_2 \neq \varnothing$*, the following convergence condition is satisfied*

$$\|x_{k+1} - x\|_X^2 \leqslant \|x_k - x\|_X^2 , \ \forall x \in S_1 \cap S_2, k \geqslant 0. \tag{5.24}$$

Proof. See Owens and Jones (1978); Gubin et al. (1967) for the detailed proof. □

5.2.2 Generalized ILC with Constraint Handling

Direct application of Theorem 5.2 to the generalized ILC design problem (5.14) with $X = \hat{H}$ and S_1, S_2 defined in (5.15) and (5.16) yields the next algorithm.

Algorithm 5.3. Given system dynamics, $S(A, B, C)$, input constraint set, Ω, output constraint set, Φ, extended reference, r^e, any initial input signal, $u_0 \in \Omega$, initial value, $\tilde{r}_0 \in \Phi$, the input sequence, $\{u_k\}_{k\geqslant 0}$, defined by the updating law

$$\tilde{u}_{k+1} = u_k + G^{s*}(I + G^s G^{s*})^{-1} e_k^s \tag{5.25}$$

followed by the projections

$$u_{k+1} = P_\Omega(\tilde{u}_{k+1}) = \arg\min_{z\in\Phi} \|z - y\|_S^2 \tag{5.26}$$

$$\tilde{r}_{k+1} = P_\Phi(\tilde{y}_{k+1}) = \arg\min_{z\in\Omega} \|z - u\|_R^2 \tag{5.27}$$

iteratively solves the generalized ILC problem (5.14), where G^s is a linear operator defined by

$$G^s u = \begin{bmatrix} G^e_\Lambda u \\ Gu \end{bmatrix} : L_2^\ell[0,\ T] \to \tilde{H} \tag{5.28}$$

whose Hilbert adjoint operator is G^{s*}, the error, e_k^s, is defined as

$$e_k^s = \begin{bmatrix} e_k^e \\ \tilde{e}_k \end{bmatrix},\ e_k^e = r^e - y_k^e,\ \tilde{e}_k = \tilde{r}_k - y_k, \tag{5.29}$$

and $\tilde{H}$ is the Hilbert space denoted by

$$\tilde{H} = \mathbb{R}^{f_1} \times \cdots \times \mathbb{R}^{f_M} \times L_2^{p_1}[t_0,\ t_1] \times \cdots \times L_2^{p_M}[t_{M-1},\ t_M] \times L_2^m[0,\ T] \tag{5.30}$$

and the inner product and associated induced norm of which are naturally derived from (3.4) and (5.3).

Proof. To apply Theorem 5.2 to generalized ILC problem (5.14), the necessary projections are first computed. From the definition of Hilbert space $\hat{H}$ in (5.17), denote $x = (y^e, y, u)$ to be an element belonging to $\hat{H}$. The projection operator P_{S_1} in Theorem 5.2 is hence

$$\begin{aligned} P_{S_1}(x) &= \arg\inf_{\hat{x}\in S_1} \|\hat{x} - x\|_X^2 \\ &= \arg\inf_{(\hat{y}^e,\hat{y},\hat{u})\in\hat{H}} \left\| \left(\hat{y}^e, \hat{y}, \hat{u}\right) - \left(y^e, y, u\right) \right\|_{\{\tilde{Q},\ S,\ R\}}^2,\ \text{s.t. } \hat{y}^e = G^e_\Lambda \hat{u},\ \hat{y} = G\hat{u} \\ &= \arg\inf_{(\hat{y}^e,\hat{y},\hat{u})\in\hat{H}} \|\hat{y}^e - y^e\|_{\tilde{Q}}^2 + \|\hat{y} - y\|_S^2 + \|\hat{u} - u\|_R^2,\ \text{s.t. } \hat{y}^e = G^e_\Lambda \hat{u},\ \hat{y} = G\hat{u}. \\ &= \inf_{\hat{u}} \|G^e_\Lambda \hat{u} - y^e\|_{\tilde{Q}}^2 + \|G\hat{u} - y\|_S^2 + \|\hat{u} - u\|_R^2. \end{aligned} \tag{5.31}$$

Theoptimization problem (5.31) yields solution $\hat{u} = u^*$ with

$$u^* = u + G^{s*}(I + G^s G^{s*})^{-1} \begin{bmatrix} y^e - G^e_\Lambda u \\ y - Gu \end{bmatrix}. \tag{5.32}$$

It follows from the definition (5.15) that

$$P_{S_1}(x) = \left(G^e u^*, Gu^*, u^*\right) \tag{5.33}$$

where u^* is given by (5.32). Performing a similar procedure for projection operator P_{S_2} yields

$$\begin{aligned} P_{S_2}(x) &= \arg \inf_{\hat{x}\in S_2} \|\hat{x}-x\|_X^2 \\ &= \arg \inf_{(\hat{y}^e,\hat{y},\hat{u})\in\hat{H}} \left\| \left(\hat{y}^e,\hat{y},\hat{u}\right) - \left(y^e,y,u\right) \right\|_{\{\tilde{Q},\ S,\ R\}}^2, \text{ s.t. } \hat{y}^e = r^e,\ \hat{u}\in\Omega,\ \hat{y}\in\Phi \\ &= \arg \inf_{(\hat{y}^e,\hat{y},\hat{u})\in\hat{H}} \|\hat{y}^e-y^e\|_{\tilde{Q}}^2 + \|\hat{y}-y\|_S^2 + \|\hat{u}-u\|_R^2, \text{ s.t. } \hat{y}^e = r^e,\ \hat{u}\in\Omega,\ \hat{y}\in\Phi. \end{aligned} \tag{5.34}$$

In optimization problem (5.34), the elements $\hat{y}^e$, $\hat{y}$ and $\hat{u}$ are independent of one another, which means this solution can be obtained separately. Using (5.26) and (5.27), it follows that

$$P_{S_2}(x) = \left(r^e, P_\Phi(y), P_\Omega(u)\right). \tag{5.35}$$

Consider update (5.20) in Theorem 5.2, and let $x_k = (r^e, \tilde{r}_k, u_k)$ and $\tilde{x}_k = (\tilde{y}_k^e, \tilde{y}_k, \tilde{u}_k)$. At the k^{th} trial, the elements $\tilde{x}_{k+1}$ and x_{k+1} are updated using projection operators P_{S_1} and P_{S_2}. For $\tilde{x}_{k+1} = P_{S_1}(x_k)$, it follows from the solution (5.33) that

$$\tilde{u}_{k+1} = u_k + G^{s*}(I + G^s G^{s*})^{-1} \begin{bmatrix} r^e - y_k^e \\ \tilde{r}_k - y_k \end{bmatrix}, \tag{5.36}$$

$$\tilde{y}_{k+1}^e = G_\Lambda^e \tilde{u}_{k+1},\ \tilde{y}_{k+1} = G\tilde{u}_{k+1},\ k \geqslant 0 \tag{5.37}$$

and for $x_{k+1} = P_{S_2}(\tilde{x}_{k+1})$, it follows from (5.35) that

$$\tilde{r}_{k+1} = P_\Phi(\tilde{y}_{k+1}),\ u_{k+1} = P_\Omega(\tilde{u}_{k+1}),\ k \geqslant 0 \tag{5.38}$$

which directly illustrates how the input u_{k+1} and the reference $\tilde{r}_{k+1}$ are updated by P_{S_2}. Therefore, Theorem 5.2 can be applied to problem (5.14) with the solutions (5.36) and (5.38) to yield Algorithm 5.3, which updates the input sequence, $\{u_k\}$, along the trial under the initial condition, $x_0 = (r^e, \tilde{r}_0, u_0) \in S_2$, i.e. $\tilde{r}_0 \in \Phi$, $u_0 \in \Omega$. □

5.3 Convergence Properties

When $S_1 \cap S_2 \neq \varnothing$, perfect tracking of the reference is possible. Algorithm 5.3 iteratively solves the generalized ILC design problem (5.14) with desirable convergence properties as shown in the next theorem.

Theorem 5.4. *If $S_1 \cap S_2 \neq \varnothing$, perfect tracking of the reference is possible. In this case Algorithm 5.3 achieves perfect tracking of the extended reference, i.e.*

$$\lim_{k\to\infty} y_k^e = r^e. \tag{5.39}$$

In addition, the input, u_k, and output, y_k, (if they exist) converge as

$$\lim_{k\to\infty} u_k = u^*, \quad \lim_{k\to\infty} y_k = y^* \tag{5.40}$$

and satisfy the system constraints that $u^* \in \Omega, y^* \in \Phi$. Furthermore, the input u_k converges monotonically with respect to the cost function

$$\tilde{J}_k = \|\tilde{r}_k - y^*\|_S^2 + \|u_k - u^*\|_R^2 \tag{5.41}$$

and the error, e_k^s, converges monotonically with respect to the cost function

$$J_k = \|\mathcal{M} e_k^s\|_{[Q]}^2 + \|\mathcal{N} e_k^s\|_R^2 \tag{5.42}$$

where $\mathcal{M} = (I + G^{s*}G^s)^{-1}, \mathcal{N} = G^{s*}(I + G^{s*}G^s)^{-1}$ and $[Q] = \{\tilde{Q},\ S\}$.

Proof. If $S_1 \cap S_2 \neq \varnothing$, there exists intersection between the two sets and perfect tracking under system constraints is possible and the minimum distance between the two sets is 0. It follows from equation (5.23) that the two sequences $\{(\tilde{y}_k^e, \tilde{y}_k, \tilde{u}_k)\}_{k\geqslant 0}$ and $\{(r^e, \tilde{r}_k, u_k)\}_{k\geqslant 0}$ attain the minimum distance between the two sets, i.e.

$$\lim_{k\to\infty} \|\tilde{y}_k^e - r^e\|_{\tilde{Q}}^2 + \|\tilde{y}_k - \tilde{r}_k\|_S^2 + \|\tilde{u}_k - u_k\|_R^2 = 0. \tag{5.43}$$

According to the above equation, it is clear that $\|\tilde{u}_k - u_k\|_R^2 = 0$ and the sequence $\{\tilde{y}_k^e\}$ converges to r^e. So the limit $y^{e*} = r^e$ of the sequence $\{y_k^e\}$ exists for perfect tracking exists as

$$y_k^e = G_\Lambda^e u_k = G_\Lambda^e \tilde{u}_k = \tilde{y}_k^e. \tag{5.44}$$

In addition, the input, u_k, and reference, $\tilde{r}_k$, are obtained from the projection operators P_Ω and P_Φ respectively, i.e.

$$u_k \in \Omega,\ \tilde{r}_k \in \Phi. \tag{5.45}$$

Therefore, if the limits of the sequences $\{u_k\}$ and $\{y_k\}$ exist, it follows that $u^* \in \Omega, y^* \in \Phi$. Then, substitute $\tilde{x}_k = (\tilde{y}_k^e, \tilde{y}_k, \tilde{u}_k)$ and $x_k = (r^e, \tilde{r}_k, u_k)$ with the update solution (5.25) into the monotonic convergence condition (5.21) to give $J_{k+1} \leqslant J_k$, and substitute $x^* = (r^e, y^*, u^*) \in S_1 \cap S_2$ and $x_k = (r^e, \tilde{r}_k, u_k)$ into the monotonic convergence condition (5.24) to give $\tilde{J}_{k+1} \leqslant \tilde{J}_k$, which completes the proof. □

The above theorem shows that the proposed algorithm solves the generalized ILC design problem, i.e. perfect tracking is achieved and the converged input and output satisfy the

system constraints. Moreover, this convergence has a certain form of monotonicity with respect to the performance defined above, e.g. weighted error norm as in (5.42), which is appealing in practice. Furthermore, as a by product, when there is no constraints, the algorithm can be simplified and has the property that it will converge to a minimum norm solution with zero initial input, i.e. the control input with minimum energy is achieved, as shown in the following corollary.

Corollary 5.5. *If $S_1 \cap S_2 \neq \varnothing$, in the absence of system constraints, Algorithm 5.3 has the monotonic convergence property with respect to the performance index*

$$\hat{J}_k = \langle u_k - u^*,\ \mathcal{H}(u_k - u^*)\rangle_X \tag{5.46}$$

for all $k \geqslant 0$, u_0 and u^*, where $\mathcal{H} = G^{e*}G + I$, and G^{e*} is the Hilbert adjoint operator of G^e. Moreover, the input energy converges to the minimum control effort with initial condition $u_0 = 0$, i.e.

$$\lim_{k\to\infty} \|u_k\|_R^2 = \min_u \{\|u\|_R^2,\ s.t.\ r^e = G^e u\}. \tag{5.47}$$

Proof. In the absence of system constraints, the sets S_1 and S_2 can be simplified as follows.

$$S_1 = \{(y^e, u) \in H \times L_2^\ell[0,\ T] : y^e = G^e u\} \tag{5.48}$$
$$S_2 = \{(y^e, u) \in H \times L_2^\ell[0,\ T] : y^e = r^e\}. \tag{5.49}$$

Substitute $x^* = (r^e, u^*)$ and $x_k = (r^e, u_k)$ into monotonic convergence condition (5.24), which hence gives rise to $\hat{J}_{k+1} \leqslant \hat{J}_k,\ \forall k \geqslant 0$. The proof of input energy follows from Theorem 1 in Owens et al. (2015) by considering the Lagrangian associated with the minimum input energy

$$\mathcal{L}(u, \lambda) = \|u\|_R^2 + 2\langle \lambda,\ r^e - G^e u\rangle_{\tilde{Q}} \tag{5.50}$$

where λ is the Lagrange multiplier. The problem has a unique stationary point $u_\infty = G^{e*}\lambda$ and $r^e = G^e u_\infty$ which hence leads to $r^e = G^e G^{e*}\lambda$. The stationary point solution solves the minimum energy problem. In the absence of system constraints, the input u_k is

$$\begin{aligned} u_k &= G^{e*}\sum_{i=1}^{k} X^i e_0^e = G^{e*}\sum_{i=0}^{k-1} X^i (I - X)\lambda_0 \\ &= G^{e*}(I - X^k)\lambda_0. \end{aligned} \tag{5.51}$$

where $X = (I + G^e G^{e*})^{-1}$ and $e_0^e = r^e = G^e G^{e*}\lambda_0$ as $u_0 = 0$. It follows that u_k converges in norm to an input $\hat{u}_\infty = G^{e*}\lambda_0$ and $r^e = G^e G^{e*}\lambda_0$, which is the unique stationary point

of the Lagrangian. It is clear that $\hat{u}_\infty = u_\infty$ and $\lambda_0 = \lambda$, and hence the input energy converges to the minimum control effort. This completes the proof. □

When $S_1 \cap S_2 = \varnothing$, perfect tracking of the reference is not possible. There does not exist any input satisfying the tracking requirement without violating the system constraints. In this case, this algorithm still attempts to solve the constrained generalized ILC design, as shown in the next theorem.

Theorem 5.6. *If $S_1 \cap S_2 = \varnothing$, perfect tracking of the reference under the system constraints is not possible. In this case, the distance between the two sequences $\{(\tilde{y}_k^e, \tilde{y}_k, \tilde{u}_k)\}$ and $\{(r^e, \tilde{r}_k, u_k)\}$ given by Algorithm 5.3 converges to the minimum distance between S_1 and S_2, i.e.*

$$\inf_u \|r^e - G_\Lambda^e \tilde{u}\|_{\tilde{Q}}^2 + \|\tilde{r} - G\tilde{u}\|_S^2 + \|u - \tilde{u}\|_R^2, \ \tilde{r} \in \Phi, \ u \in \Omega. \tag{5.52}$$

In addition, the input, u_k, at each trial satisfies the constraint that $u_k \in \Omega$. Furthermore, the error, e_k^s, converges monotonically with respect to the cost function, J_k, defined in (5.42).

Proof. If $S_1 \cap S_2 = \varnothing$, perfect tracking is not possible under system constraints. It also follows Theorem 5.2 that the two sequences $\{(\tilde{y}_k^e, \tilde{y}_k, \tilde{u}_k)\}_{k\geqslant 0}$ and $\{(r^e, \tilde{r}_k, u_k)\}_{k\geqslant 0}$ still attain the minimum distance between the two sets as shown in (5.52). Also, the input, u_k, at each trial belongs to the input constraint set, Ω, as it is obtained from the projection operator, P_Ω. The proof of the monotonic convergence with respect to J_k follows from the similar proof in Theorem 5.4. The proof is now complete. □

Remark 5.7. It is worth pointing out that the scenario in the above theorem is not well-posed. In practice an appropriate tracking requirement, r^e, should be specified to avoid this impossible tracking task, i.e. $S_1 \cap S_2 = \varnothing$.

5.4 Implementation of the Algorithm

Implementation of Algorithm 5.3 consists of two steps: the ILC update (5.25) and projection steps (5.26), (5.27). The update (5.25) can be directly implemented as a feedforward solution using u_k and e_k^s to construct $\tilde{u}_{k+1}$. Alternatively, it can be implemented using a causal feedback plus feedforward structure by employing the state feedback to further embed potential robust performance in practice. This exploits the special properties of the linear operator G^e and its adjoint operator G^{e*}. To formulate the causal feedback plus feedforward solution, the following lemma is needed.

Lemma 5.8. *The Hilbert adjoint operator* $G^{s*} : (\omega, \nu, y) \in \tilde{H} \to u \in L_2^\ell[0,\ T]$ *has the following analytic form*

$$u(t) = R^{-1}B^\top p(t), \tag{5.53}$$

where $p(t)$ *is computed in reverse time as follows*

$$\dot{p}(t) = A^\top p(t) + C^\top(P_i^\top \hat{Q}_i \nu_i(t) + Sy(t)),\ t \in [t_{i-1},\ t_i],\ i = 1, \ldots, M.$$

with boundary conditions

$$\begin{aligned} p(t_i^-) &= p(t_i^+) + C^\top F_i^\top Q_i \omega_i, i = 1, \ldots, M, \\ p(T) &= 0 \end{aligned} \tag{5.54}$$

Proof. The relevant adjoint operator G^{s*} is obtained based on inner product form

$$\langle (\omega, \nu, y),\ G^s u \rangle_{[Q]} = \langle G^{s*}(\omega, \nu, y),\ u \rangle_R \tag{5.55}$$

where $[Q] = \{\tilde{Q},\ S\}$. Note that G^s consists of FG, PG and G, whose adjoints are computed separately as follows:

1). Adjoint Operator of FG: The operator FG has the following structure

$$FGu = \begin{bmatrix} G_1 u \\ \vdots \\ G_M u \end{bmatrix} \tag{5.56}$$

where

$$G_i u = F_i \int_0^{t_i} C e^{A(t_i - t)} B u(t),\ i = 1, \ldots, M. \tag{5.57}$$

Then consider the operator $G_i : L_2^\ell[0,\ T] \to \mathbb{R}^{f_i}$ via equation

$$\begin{aligned} \omega_i^\top Q_i G_i u &= \omega_i^\top Q_i F_i \int_0^{t_i} C A^{t_i - t} B u(t) \\ &= \int_0^{t_i} (R^{-1} B (A^\top)^{t_i - t} C^\top F_i^\top Q_i \omega_i)^\top R u(t), \end{aligned} \tag{5.58}$$

and the condition

$$\omega_i^\top Q_i G_i u = \int_0^T ((G_i^* \omega_i)(t))^\top R u(t), \tag{5.59}$$

held by the definition of the adjoint operator, i.e.

$$\langle \omega_i,\ G_i u \rangle_{Q_i} = \langle G_i^* \omega_i,\ u \rangle_R . \tag{5.60}$$

Hence, the above two equations give rise to the adjoint of G_i as

$$(G_i^*\omega_i)(t) = \begin{cases} R^{-1}Be^{A^\top(t_i-t)}C^\top F_i^\top Q_i\omega_i, & t \leqslant t_i, \\ 0, & t > t_i \end{cases}$$

which can be further written as

$$(G_i^*\omega_i)(t) = R^{-1}B^\top p_i(t) \tag{5.61}$$

where $p_i(t) = 0$ on $[t_i,\ T]$, and on $[0,\ t_i)$

$$\dot{p}_i(t) = A^\top p_i(t),\ p_i(t_i) = C^\top Q_i\omega_i. \tag{5.62}$$

2). Adjoint Operator of G: The adjoint of G is computed from the inner product form

$$\langle y,\ Gu\rangle_S = \langle G^*y,\ u\rangle_R \tag{5.63}$$

as the map $u = G^*y$ as follows:

$$\begin{aligned} &(G^*y)(t) = R^{-1}B^\top p_{M+1}(t),\ p_{M+1}(N) = 0, \\ &\dot{p}_{M+1}(t) = A^\top p_{M+1}(t) + C^\top Sy(t). \end{aligned} \tag{5.64}$$

3). Adjoint Operator of PG: Note that PG is simply the composite map G and the map P such that

$$y(t) \to P_iy(t),\ t \in [t_{i-1},\ t_i],\ i = 1, \ldots, M. \tag{5.65}$$

It follows that the adjoint operator P_i^* is computed as

$$y(t) = S^{-1}P_i^\top \hat{Q}_i\nu_i(t), t \in [t_{i-1}, t_i], i = 1, \ldots, M. \tag{5.66}$$

from the inner product form

$$\langle \nu_i,\ P_iy\rangle_{\hat{Q}_i} = \langle P_i^*y,\ y\rangle_S \tag{5.67}$$

Hence, the adjoint of PG defined by the relation $u = (PG)^*(\nu_1, \ldots, \nu_M)$ can be written as $(PG)^* = G^*P^*$, and computed as

$$\begin{aligned} &((PG)^*\nu)(t) = R^{-1}B^\top p_{M+2}(t),\ p_{M+2}(N) = 0 \\ &\dot{p}_{M+2}(t) = A^\top p_{M+2}(t) + C^\top P_i^\top \hat{Q}_i\nu_i(t),\ t \in [t_{i-1},\ t_i],\ i = 1, \ldots, M. \end{aligned} \tag{5.68}$$

by substituting (5.66) into (5.64).

4). Adjoint Operator of G^{s*}: The adjoint operator G^{s*} is the map defined as $(\omega, \nu, y) \mapsto u$. Due to linearity, the adjoint operator G^{s*} can be expressed as the sum of the adjoints

of FG, PG and G, i.e.

$$\begin{aligned} G^{s*}(\omega,\nu,y) &= \sum_{i=1}^{M}(G_i^*\omega_i)(t) + (G^*y)(t) + ((PG)^*\nu)(t) \\ &= R^{-1}B^\top p(t) \end{aligned} \tag{5.69}$$

as shown in (5.53), where $p(t) = \sum_{i=1}^{M+2} p_i(t)$. Based on the above representations of G_i^*, G^* and $(PG)^*$, the costate $p(t)$ in (5.53) is obtained together with its boundary conditions (5.54). These together generate the definition of the adjoint operator G^{s*} in Lemma 5.8. □

Using Lemma 5.8, the feedback plus feedforward implementation is given in the next proposition.

Proposition 5.9. *The ILC update (5.25) in Algorithm 5.3 can be implemented in a feedforward plus feedback solution*

$$u_{k+1}(t) = u_k(t) + R^{-1}B^\top p_k(t) \tag{5.70}$$

with

$$p_k(t) = -K(t)(x_{k+1}(t) - x_k(t)) + \xi_{k+1}(t) \tag{5.71}$$

where $K(t)$ is the solution of the Riccati equation

$$0 = \dot{K}(t) + (A^\top - K(t)BR^{-1}B^\top)K(t) + K(t)A + C^\top\hat{Q}_i(t)C + C^\top SC \tag{5.72}$$

with boundary conditions

$$\begin{aligned} K(t_i-) &= K(t_i+) + C^\top Q_i C,\ 1 \leqslant i \leqslant M \\ K(T) &= 0, \end{aligned} \tag{5.73}$$

and $\xi_{k+1}(t)$ denotes the feedforward term at the $(k+1)^{th}$ trial generated by the difference equation

$$0 = \dot{\xi}_{k+1}(t) + (A^\top - K(t)BR^{-1}B^\top)\xi_{k+1}(t) - C^\top\hat{Q}(t)e_{k+1}(t) + C^\top S\hat{e}_k(t) \tag{5.74}$$

with boundary conditions

$$\begin{aligned} \xi_{k+1}(t_i-) &= \xi_{k+1}(t_i+) + C^\top Q e_k(t_i),\ 1 \leqslant i \leqslant M \\ \xi_{k+1}(T) &= 0 \end{aligned} \tag{5.75}$$

in which $\hat{Q}(t) = P_i^\top\hat{Q}_iP_i$ for $t \in [t_{i-1},\ t_i]$, $i = 1,\ldots,M$.

Proof. The ILC update (5.25) is equivalent to

$$u_{k+1}(t) = u_k(t) + G^{s*}\tilde{e}^s_{k+1}(t) \tag{5.76}$$

where $\tilde{e}^s_{k+1} = [e^e_{k+1},\ \hat{e}_{k+1}]^\top$ and $\hat{e}_{k+1} = \tilde{r}_k - y_{k+1}$. From Lemma 5.8, $G^{s*}\tilde{e}^s_{k+1}(t)$ can be computed using the analytical form

$$G^{s*}\tilde{e}^s_{k+1}(t) = R^{-1}B^\top p_k(t) \tag{5.77}$$

where the costate, $p_k(t)$, is computed in reverse time as

$$\dot{p}(t) = -A^\top p(t) - C^\top(P_i^\top \hat{Q}_i P_i e_{k+1}(t) + S\hat{e}_{k+1}(t)),\ t \in [t_{i-1},\ t_i],\ i = 1, \ldots, M,$$

with boundary conditions

$$\begin{aligned} p(t_i-) &= p(t_i+) + C^\top F_i^\top Q_i F_i e_{k+1}(t_i),\ i = 1, \ldots, M \\ p_k(T) &= 0 \end{aligned} \tag{5.78}$$

Substituting (5.77) into (5.76) yields the solution (5.70).

Assuming that full state knowledge are known, the costate equation (5.77) yields a causal implementation

$$p_k(t) = -K(t)(x_{k+1}(t) - x_k(t)) + \xi_{k+1}(t). \tag{5.79}$$

Then use the method developed in Amann (1996) such that it follows from (3.1), (5.70) and (5.79) that

$$\begin{aligned} \dot{x}_{k+1}(t) - \dot{x}_k(t) &= A(x_{k+1}(t) - x_k(t)) + B(u_{k+1}(t) - u_k(t)) \\ &= A(x_{k+1}(t) - x_k(t)) + BR^{-1}B^\top p_{k+1}(t) \\ &= (I - BR^{-1}B^\top K(t))A(x_{k+1}(t) - x_k(t)) + BR^{-1}B^\top \xi_{k+1}(t) \end{aligned} \tag{5.80}$$

Then substitute (5.79) and (5.80) into the costate equation (5.77) to yield an equation of the form

$$\begin{aligned} &\mathcal{H}(A,B,C,S,\hat{Q}_i,P_i,R^{-1},K(t),K(t+1))[x_{k+1}(t+1) - x_k(t+1)] \\ &\quad = \mathcal{G}(A,B,C,\hat{Q}_i,P_i,R^{-1},K(t),\xi_{k+1}(t),\xi_{k+1}(t+1),e_k(t+1),\tilde{e}_k(t+1)) \end{aligned} \tag{5.81}$$

where $\mathcal{H}(\cdot)$ and $\mathcal{G}(\cdot)$ are functions of their arguments and independent of the states. If both functions are set to zero, the equation (5.81) holds independently of the current difference in state. Doing this yields the Riccati equation $K(t)$ and the optimal predictor $\xi_{k+1}(t)$ in (5.72) and (5.74).

Comparing to the boundary conditions in equation (5.78), there is one more term added at the end, i.e. $C^\top F_i^\top Q_i F_i e_{k+1}(t_i)$. Note that

$$\begin{aligned} e_{k+1}(t_i) &= r_i - Cx_{k+1}(t_i) \\ &= e_k(t_i) - C(x_{k+1}(t_i) - x_k(t_i)), \end{aligned} \tag{5.82}$$

which yields

$$C^\top F_i^\top Q_i F_i e_{k+1}(t_i) = C^\top F_i^\top Q_i F_i C(x_{k+1}(t_i) - x_k(t_i)) + C^\top F_i Q_i F_i e_k(t_i) \tag{5.83}$$

and gives rise to the boundary conditions in (5.73) and (5.75). □

The ILC update (5.25) is then followed by the projection steps in (5.26) and (5.27) to project the unconstrained input, $\tilde{u}_{k+1}$, and output, $\tilde{y}_{k+1}$, into the input and output constraint sets, Φ and Ω, respectively, which are usually straightforward. For example, the input constraint set, Ω, is usually a pointwise constraint in practice, so the solution of the projection operator, P_Ω, in (5.26) is straightforward. As another example, when the input constraint set, Ω, has the saturation form (5.9), the solution of $u = P_\Omega(\tilde{u})$ is given by

$$u(t) = \begin{cases} M(t), & \tilde{u}(t) \succ M(t), \\ \tilde{u}(t), & -M(t) \preceq \tilde{u}(t) \preceq M(t), \\ -M(t), & \tilde{u}(t) \prec -M(t) \end{cases} \tag{5.84}$$

for $0 \leqslant t \leqslant T$. Also the solution of the projection operator, P_Φ, in (5.27) is guaranteed to be unique. Consider the polyhedral output constraint set (2.59) for example, the solution of $\tilde{r} = P_\Omega(\tilde{y})$ is given by

$$\tilde{r}(t) = \begin{cases} N(t), & \tilde{y}(t) \succ N(t), \\ \tilde{y}(t), & -N(t) \preceq \tilde{y}(t) \preceq N(t), \\ -N(t), & \tilde{y}(t) \prec -N(t) \end{cases} \tag{5.85}$$

for $0 \leqslant t \leqslant T$.

5.5 Piecewise Spatial Path Tracking Problem

As described in the introduction, the spatial tracking problem requires the output trajectory to follow a continuous path defined in space with no temporal constraints. The framework developed in this chapter can be applied to produce the first spatial ILC algorithm capable of converging to an optimal solution to the problem. The common class of piecewise spatial tracking tasks is considered, which is defined as follows:

Definition 5.10. The piecewise linear spatial tracking problem is to design an input such that the output travels between each pair of vertices, r_{i-1} and r_i, in ascending order, i.e. having reached r_{i-1} at t_{i-1}, remain in interval between r_{i-1} and r_i until r_i is reached at t_i, and then repeat the process for the next pair, r_i and r_{i+1}.

Note that the vertices $r_i,\ i = 0, \ldots, M$ are defined in Cartesian space $\mathbb{R}^m$ as $r_i = [r_i^1,\ r_i^2, \ldots, r_i^m]^\top$, and a special example of the piecewise spatial path, i.e. $m = 2$, is shown in Figure 5.2.

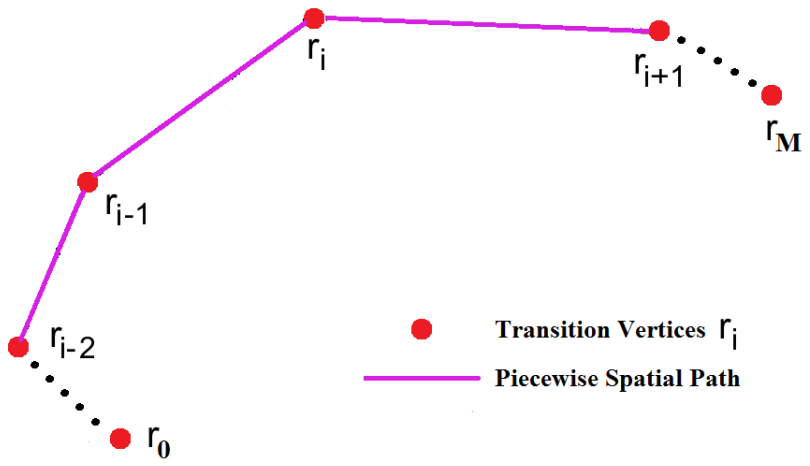


Figure 5.2: Piecewise Spatial Path in $\mathbb{R}^2$.

The next theorem illustrates how Algorithm 5.3 solves the spatial problem of Definition 5.10 as a special case.

Theorem 5.11. *The piecewise linear spatial path tracking problem is solved by Algorithm 5.3 with*

$$r^e = \begin{bmatrix} Fr \\ Pr \end{bmatrix} \tag{5.86}$$

where $r \in L_2^m[0,\ T]$ *is given by*

$$r(t) = r_i,\ t \in (t_{i-1},\ t_i],\ i = 1, \ldots, M, \tag{5.87}$$

the operator F *is defined by* $F_i = I, i = 1, \ldots, M$ *and the operator* P *by* $P_i \in R^{(m-1)\times m}$ *a full rank matrix satisfying*

$$\operatorname{Ker} P_i = \operatorname{Im} a_i, \tag{5.88}$$

where $a_i = r_i - r_{i-1}$, *for* $i = 1, ..., M$, *and the output constraint set is*

$$\Phi = \{y \in L_2^m[0,\ T] : a_i^\top r_{i-1} \leqslant a_i^\top y(t) \leqslant a_i^\top r_i,\ t \in (t_{i-1},\ t_i],\ i = 1, \ldots, M\}. \tag{5.89}$$

Using Algorithm 5.3, if $S_1 \cap S_2 \neq \varnothing$, the extended output, y_k^e, has the following convergence properties

$$\lim_{k\to\infty} y_k^e = r^e. \tag{5.90}$$

That is, perfect tracking of the spatial path is achieved.

Proof. According to Definition 5.10, the piecewise spatial tracking requirement is to track the transition vertices r_i at time instants t_i and the linear path between r_{i-1} and r_i during the sub-interval, $[t_{i-1},\ t_i]$. Therefore, the piecewise spatial path tracking problem is

$$\begin{aligned} &(Gu)(t_i) = r_i,\ \ i = 1, \ldots, M \\ &Gu(t) \in \mathcal{R}_i,\ \ t \in [t_{i-1},\ t_i],\ \ i = 1, \ldots, M. \end{aligned} \tag{5.91}$$

where $\mathcal{R}_i$ is the set of all points along each linear sub-path, i.e.

$$\mathcal{R}_i = \{y \in \mathbb{R}^m : y = r_i - \gamma(r_i - r_{i-1}),\ \gamma \in [0,\ 1]\}. \tag{5.92}$$

Note the all the elements in $\mathcal{R}_i$ satisfy the linear relationship

$$P_i y = P_i r_i,\ \ \forall y \in \mathcal{R}_i \tag{5.93}$$

where $P_i \in R^{(m-1)\times m}$ is a full row rank matrix. It follows that

$$P_i(r_i - \gamma(r_i - r_{i-1})) = P_i r_i \tag{5.94}$$

which yields

$$P_i a_i = 0 \tag{5.95}$$

and further gives rise to the condition (5.88). In addition, the hard output constraint set defined in (5.89) is used to prevent overshoot. Therefore, a piecewise spatial ILC problem is formulated as

$$\begin{aligned} &\lim_{k\to\infty} y_k = y^* \in \Phi,\ \ \lim_{k\to\infty} y_k(t_i) = r_i,\ \ i = 0, \ldots, M, \\ &\lim_{k\to\infty} P_i y_k(t) = P_i r_i, t \in (t_{i-1},\ t_i], i = 1, \ldots, M. \end{aligned} \tag{5.96}$$

Therefore, it is clear that the generalized ILC problem (5.14) is reduced to exactly the problem (5.96) when the parameters are set to the values described in this theorem. Therefore, Algorithm 5.3 solves the problem (5.91) as a special case. As the piecewise spatial path tracking problem is a special case of generalized ILC problems, the proofs of the convergence properties in (5.90) follow exactly from the proofs of Theorem 5.4 and 5.6. □

From the above theorem, the proposed Algorithm 5.3 solves the high performance piecewise linear spatial tracking problem. The next corollary shows that Algorithm 5.3 also fully exploits the spatial tracking tasks design freedom in the temporal domain to allow minimization of control effort for a general class of systems, under some mild conditions.

Lemma 5.12. *If $S_1 \cap S_2 \neq \varnothing$, in the absence of system constraints (e.g. if overshoot can be tolerated or does not occur), Algorithm 5.3 solves the spatial problem in Definition 5.10 with minimum control effort with initial input choice $u_0 = 0$.*

Proof. The proof of the minimum control effort follows from the proof of Corollary 5.5. □

Remark 5.13. The implementation of the aforementioned spatial path tracking design is exactly the same as that discussed in the previous sections, in partiualr the unique solution $\tilde{r} = P_\Phi(\tilde{y})$ is given as

$$\tilde{r}(t) = \begin{cases} \tilde{y}(t) + \Delta_{i,i}(t), & a_i^\top \tilde{y}(t) > a_i^\top r_i, \\ \tilde{y}(t), & a_i^\top r_{i-1} \leqslant a_i^\top \tilde{y}(t) \leqslant a_i^\top r_i \\ \tilde{y}(t) + \Delta_{i,i-1}(t), & a_i^\top \tilde{y}(t) < a_i^\top r_{i-1} \end{cases} \tag{5.97}$$

for $t \in (t_{i-1},\ t_i]$, $i = 1, \ldots, M$ where

$$\Delta_{i,j}(t) = (a_i^\top a_i)^{-1} a_i^\top (r_j - \tilde{y}(t)) a_i.$$

5.6 Experimental Verification

In this section, the proposed algorithm is validated experimentally on a three-axis gantry robot test platform to demonstrate its effectiveness.

5.6.1 Design Task Specification

Consider the multi-axis gantry robot shown in Chapter 3.5.1 with the system model (3.83) as the test platform. The control design objective is to use both the x-axis and z-axis ($m = 2$) to track a piecewise linear spatial path composed of five line segments ($M = 5$) with

$$r_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, r_1 = \begin{bmatrix} 0.00345 \\ 0.00476 \end{bmatrix}, r_2 = \begin{bmatrix} 0.00905 \\ 0.00294 \end{bmatrix},$$
$$r_3 = \begin{bmatrix} 0.00905 \\ -0.00294 \end{bmatrix}, r_4 = \begin{bmatrix} 0.00345 \\ -0.00476 \end{bmatrix}, r_5 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

as shown in subsequent Figure 5.3 (the yellow line), during the given tracking time $T = 2s$. The tracking time instants of the intermediate points are given as

$$t_1 = 0.4,\ t_2 = 0.8,\ t_3 = 1.2,\ t_4 = 1.6,\ t_5 = 2.0. \tag{5.98}$$

and a proportional feedback gain 300 is added on the z-axis again for smooth tracking at turning edge. The input voltage has the saturation constraint in the form (5.9) with $M(t) = [0.6, 2]^\top$. The parameters F_i, P_i and $r(t)$ are chosen according to the values in (5.86). For simplicity, the weighting matrices Q_i, $\hat{Q}_i$, S and R (standing for the weights of intermediate point, sub-interval, output and input respectively) are chosen to be diagonal.

5.6.2 Performance of the Proposed Algorithm

Firstly, the system constraints are artificially removed (i.e. setting $\Omega = L_2^\ell[0,\ T]$ and $\Phi = L_2^m[0,\ T]$) and the proposed algorithm is applied to this task on the gantry robot for 100 trials to see the performance at unconstrained case. The final converged hybrid output and inputs of the two axes are plotted in Figure 5.3 and Figure 5.4 respectively. Although the converged hybrid output performs near perfect tracking along the piecewise linear reference path, it is clear that overshoot problem takes place. In the gantry robot test platform, the overshoot problem may lead to collision between the end-effector and the frame, which causes damage to the machine. In addition, the corresponding inputs of the two axes exceeds the input constraint set Ω defined in (5.89) at certain time intervals, which is not allowed in practice. Therefore, it is necessary to apply the system constraints in practice to meet the actual system requirement of the design task.

To avoid the above problems, Algorithm 5.3 is applied to the same task with the input saturation constraint defined in (5.9) and the hard output constraint defined in (5.89). In the experiment, $\hat{Q}_i = 100,000I$, $Q_i = 500,000I$, $S = 10,000I$ and $R = I$ are chosen, and a total of 100 update trials are performed. The final converged hybrid output of the two axes is plotted as the dashed magenta trajectory in Figure 5.3. Compared to the previous overshoot result in the same figure, it is clear that Algorithm 5.3 not only achieves the generalized tracking requirement, but keeps the hybrid output trajectory within the output constraint set, Φ, defined in (5.89), i.e. this algorithm solves the overshoot problem. Furthermore, the final converged input voltages of the two axes are plotted as the dashed magenta trajectories in Figure 5.4, and it is clear that both stay within the input constraint set, Ω, defined in (5.9) with $M(t) = [0.6, 2]^\top$. Therefore, it is clear from Figure 5.3 and Figure 5.4 that Algorithm 5.3 can not only guarantee high performance spatial tracking but also handle the system constraints well.

The proposed algorithm is further applied with different parameters to compare convergence properties. The coefficients $\hat{Q}_i = 100,000I$, $S = 10,000I$ and $R = I$ are kept as

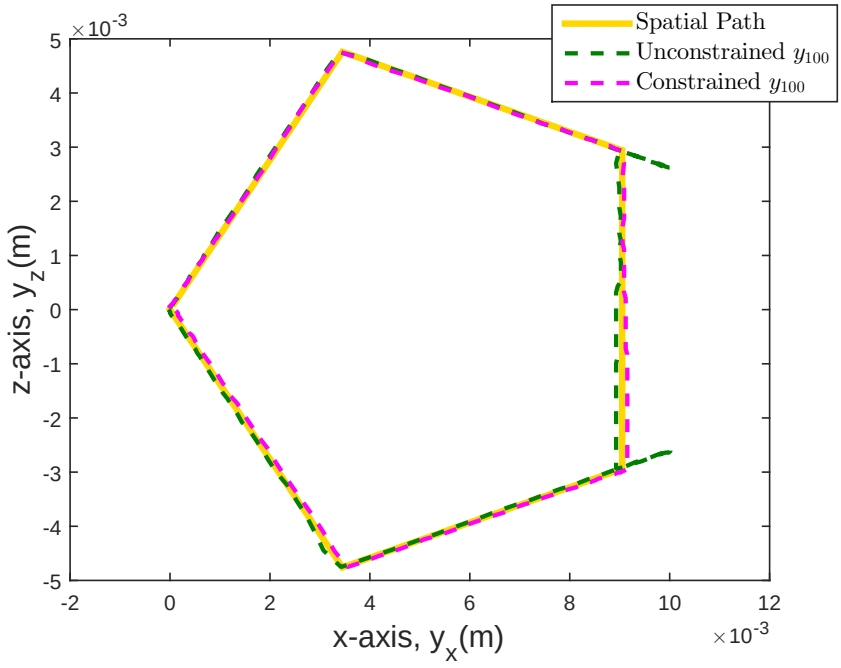


Figure 5.3: Spatial Path and Converged Hybrid Output Trajectories with and without Output Constraints.

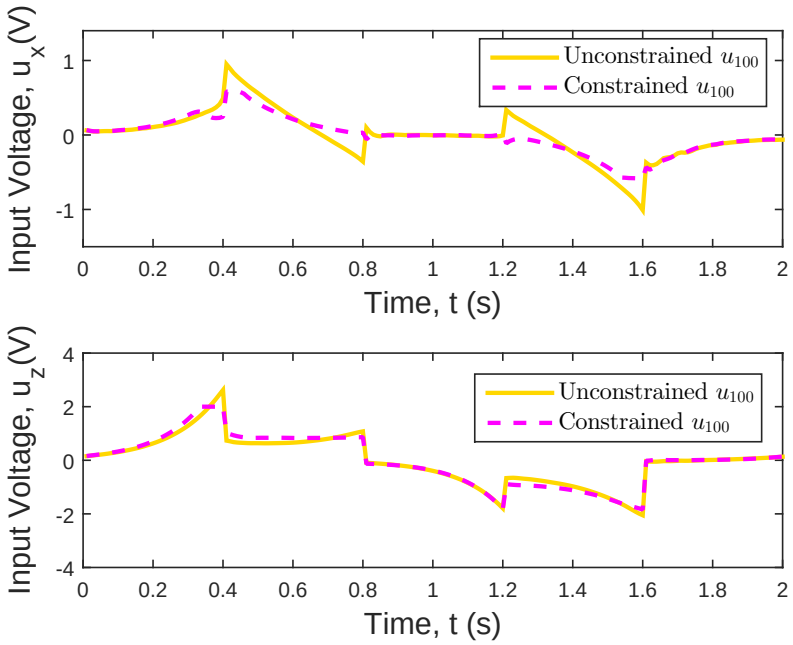


Figure 5.4: Converged Input Trajectories.

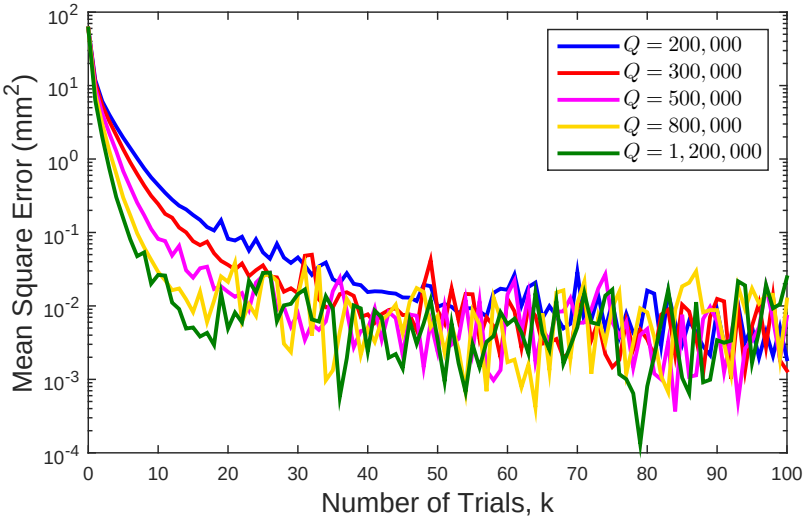


Figure 5.5: Mean Square Tracking Error and Input Energy over 100 Trials with Output Constraints.

Table 5.1: Input Energy Comparison.

	Input Energy (V^2ms)
$Q_i = 200,000$	1,819.9
$Q_i = 300,000$	1,820.3
$Q_i = 500,000$	1,812.8
$Q_i = 800,000$	1,810.9
$Q_i = 1,200,000$	1,807.8
Traditional ILC	2,146.9

constants, and Q_i is selected to take the values $200,000I$, $300,000I$, $500,000I$, $800,000I$, and $1,200,000I$. A total of 100 update trials are performed for each value of Q_i, and the corresponding mean square error, e_k^s, at each trial is plotted in Figure 5.5. From this figure, it is obvious that the convergence rate increases as the increase of the weighting value Q_i. It is noted that all plots converge to below 0.01 mean square error, which verifies accurate tracking in practice despite of model uncertainty and random disturbance. It is noted that there are no particular concerns about the fluctuation in the figure as the mean square errors all converge and satisfy the practical tracking requirement.

In addition, the converged input energy for different values of Q_i are shown in Table 5.1. As in many practical applications, e.g. laser cutting and welding, the end-effector moves at a constant speed along its path, the input energy needed for the constant speed tracking obtained by traditional ILC is also shown in this table for comparison. The table shows that the proposed algorithm fully exploits the design freedom in temporary domain in terms of an approximate 16% energy reduction from that provided by traditional ILC.

Experiments with other combinations of the weighting matrices Q_i, $\hat{Q}_i$, S and R yield similar convergence performance to the results in Figure 5.5. For brevity, these results are omitted.

5.7 Summary

This chapter develops a novel unified ILC design framework capable of solving tracking requirements defined on both intermediate point and sub-intervals, as well as handling a mixed form of system constraints. To solve this problem, this chapter proposed a new ILC algorithm using successive projection method with well defined convergence properties. A particularly powerful feature of the proposed algorithm is that it can solve a class of spatial tracking problems with high tracking accuracy as well as fully exploit the design freedom in the temporal domain to achieve optimal cost functions, e.g. minimum control effort, which has substantial novelty over the previous research in spatial ILC. The algorithm's convergence properties have been analyzed rigorously, and then verified on a gantry robot platform by tracking a spatial path with stipulated input and output constraints, which demonstrates its practical efficacy. Although the derived algorithm can achieve some optimal cost function within a certain class of systems, the tracking time allocation, Λ, in this framework is considered as given *a priori*. In the next chapter, the generalized ILC framework will be expanded to treat the tracking time allocation, Λ, as a variable to optimize a cost function as well as ensure perfect spatial tracking.

Chapter 6

ILC for Spatial Path Tracking

The previous chapters have formulated a generalized ILC framework, which can be applied to spatial path tracking. However, this framework has not fully harnessed this design freedom as it considers the tracking time allocation, Λ, as given *a priori*. Rather than purely achieving the path tracking, this chapter allows a flexible choice of Λ to fully exploit the design freedom of spatial ILC to optimize an additional cost function while following the defined path. The optimization of such cost functions can bring significant practical benefits, such as reducing control effort, avoiding machine damage and increasing manufacturing efficiency.

This chapter first formulates the spatial tracking problem and characterizes it into the class of piecewise linear path tracking problem. Then a novel spatial ILC framework is proposed that can automatically perform high performance spatial path tracking for a general class of systems as well as optimize a specific cost function, e.g. control effort. After that, an iterative algorithm is derived from the spatial ILC framework based on the Two Stage design framework proposed in Chapter 3 and the generalized ILC algorithm proposed in Chapter 5.

6.1 General Spatial Path Tracking Formulation

This section considers the linear time-invariant system (3.1) required to perform repetitive tracking tasks with the spatial paths, and then defines the general spatial ILC path tracking problem.

Any continuous, non-intersecting path can be defined in the output space by introducing a continuous bijective function, r, mapping each point of the interval $I = [0,\ 1]$ to a point in the output space $\mathbb{R}^m$, $r : I \to L_2^m[0,\ T]$. An example of such a path is shown in Figure 6.1 (black line). In the traditional tracking problem, the requirement that the

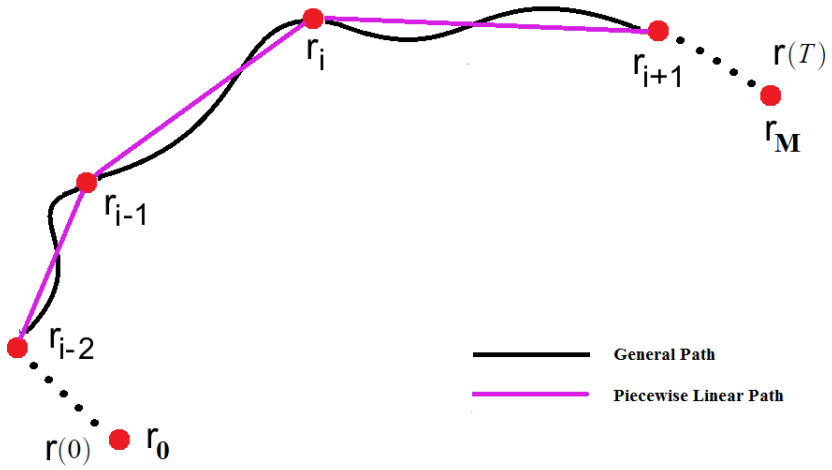


Figure 6.1: Spatial Paths as Set of Points in $\mathbb{R}^m$.

plant output follows the defined path, r, is that

$$(Gu)(t) = r(t/T),\ \forall\ t \in [0,\ T]. \tag{6.1}$$

However, the key feature of the spatial tracking problem is that the temporal component of the movement along this path from initial point $r(0)$ to final point $r(1)$ is arbitrary. It hence follows that an equivalent path is generated if $r(t)$ is replaced by any

$$\tilde{r}(t) = r(g(t)) \tag{6.2}$$

where $g : [0,\ T] \to I$ is any continuous function with $g(0) = 0$ and $g(T) = 1$. Therefore, given a stipulated path map r, the set of all possible maps describing equivalent paths is denoted as

$$\mathcal{R}_r = \{\tilde{r} : I \to \mathbb{R}^m,\ \tilde{r}(t) = r(g(t)),\ 0 \leqslant t \leqslant T\}. \tag{6.3}$$

To see that this set contains all admissible paths, note that the spatial problem is satisfied if and only if there exists a continuous surjective map, s, from I onto the set of points $\{r(t),\ t \in I\}$ making up the path (where the path may repeatedly map to the same point). Since $r^{-1}(\cdot)$ is continuous and bijective, $r^{-1}(s)$ is a continuous map which satisfies $r(r^{-1}(s)) = s$, and it follows that $s \in \mathcal{R}_r$. Then the requirement that the plant output achieve the spatial tracking requirement is

$$(Gu)(t) = \tilde{r}(t),\ \forall\ t \in [0,\ T] \text{ for some } \tilde{r} \in \mathcal{R}_r. \tag{6.4}$$

Given the infinite number of solutions to the problem, suppose it is stipulated that input u must also minimize a cost function, $f(u)$. The general spatial ILC tracking problem

is then defined as iteratively finding an input, u_k, i.e.

$$\lim_{k\to\infty} u_k = u^*, \tag{6.5}$$

such that the output, y_k, accurately pass through the path defined by $\tilde{r} \in \mathcal{R}_r$ and u^* solves problem

$$\min_{\tilde{r}\in\mathcal{R}_r,u} f(u) \text{ subject to } (Gu)(t) = \tilde{r}(t)\ \forall\ t \in [0,\ T]. \tag{6.6}$$

6.2 Characterization of Piecewise Linear Spatial Tracking Problem using ILC

In this section, the class of piecewise linear paths is considered to specify the general form (6.6) so that it may be solved using the tools of ILC. Figure 6.1 (magenta line) shows an example of a piecewise linear path comprising M segments.

6.2.1 Specification to Piecewise Linear Paths

In the simplest case of constant velocity movement along the path, a piecewise linear path may be defined using the bijective function

$$r(t) = r_{i-1} + \left(\frac{t - s_{i-1}}{s_i - s_{i-1}}\right)(r_i - r_{i-1}),\ t \in [s_{i-1}, s_i] \tag{6.7}$$

for $i = 1, \ldots, M$ where $0 = s_0 < s_1 < \ldots < s_M = 1$ and $r_i = [r_i^1,\ r_i^2, \ldots,\ r_i^m]^\top$ are the transition vertices of the piecewise path, defined in Cartesian space. The set of all possible path profiles is then given by $\mathcal{R}_r$ and does not depend on s_i, only on transition vertices r_i. Using the notation (6.2), the spatial tracking of this path is hence equivalently defined by any $\tilde{r} \in \mathcal{R}_r$. If it is further required that the i^{th} segment be completed before the $(i+1)^{th}$ segment is started, then the constraint

$$g(t_{i-1}) \leqslant g(t) \leqslant g(t_i),\ t \in [t_{i-1},\ t_i],\ g(t_i) = r^{-1}(r_i) \tag{6.8}$$

for $i = 1, \ldots, M$, must be further applied on the function g within (6.3). With the preceding piecewise spatial tracking task specifications, the spatial ILC tracking problem is defined as follows.

Definition 6.1. Spatial Piecewise Linear ILC Tracking Problem: Iteratively find an input u_k, i.e.

$$\lim_{k\to\infty} u_k = u^*, \tag{6.9}$$

such that output y_k accurately passes through the path defined by $\tilde{r} \in \mathcal{R}_r$, and u^* solves problem (6.6), where $\mathcal{R}_r$ is generated by vertices set $\{r_i\}$ and ordering constraints (6.8).

6.2.2 Piecewise Linear ILC Problem

Definition 6.1 gives rise to the following result:

Proposition 6.2. *For the piecewise linear spatial tracking problem in Definition 6.1, it follows that*

$$\tilde{r} \in \mathcal{R}_r \Leftrightarrow \tilde{r} \in \left\{ \begin{array}{c} \tilde{r} : I \to L_2^m[0,\ T], \\ \tilde{r}(t_i) = r_i,\ i = 1, \ldots, M, \\ P_i \tilde{r}(t) = P_i r_i,\ t \in [t_{i-1},\ t_i], \\ a_i^\top r_{i-1} \leqslant a_i^\top \tilde{r}(t) \leqslant a_i^\top r_i,\ t \in [t_{i-1},\ t_i], \\ 0 = t_0 < t_1 < \ldots < t_M = T \end{array} \right\} \tag{6.10}$$

where $a_i = r_i - r_{i-1}$ *for* $i = 1, ..., M,$, *and* $P_i \in R^{(m-1)\times m}$ *is a full rank matrix satisfying*

$$\operatorname{Ker} P_i = \operatorname{Im} a_i. \tag{6.11}$$

Proof. If $\tilde{r} \in \mathcal{R}_r$, there exists a continuous function $g : I \to I$ such that $\tilde{r}(t) = r(g(t))$ with $r()$ given by (6.7). By assumption, there exists a collection of time points

$$0 = t_0 < t_1 < \ldots < t_M = T \tag{6.12}$$

such that the constraint on $g(\cdot)$ given by (6.8) holds. Substitute $g(t_i) = t_i$ into (6.7) and obtain

$$\tilde{r}(t_i) = r(g(t_i)) = r(s_i) = r_i,\ i = 1, \ldots, M. \tag{6.13}$$

Since (6.8) holds, it follows that all the points $\tilde{r}(t),\ t \in [t_{i-1},\ t_i]$ belong to the line segment set

$$\mathcal{R}_i = \{y \in \mathbb{R}^m : y = r_i - \gamma(r_i - r_{i-1}),\ \gamma \in [0,\ 1]\}. \tag{6.14}$$

Note that all the elements in $\mathcal{R}_i$ have the linear relationship

$$P_i y = P_i r_i,\ \forall y \in \mathcal{R}_i \tag{6.15}$$

where $P_i \in R^{(m-1)\times m}$ is a full row rank matrix. It follows that

$$P_i(r_i - \gamma(r_i - r_{i-1})) = P_i r_i \tag{6.16}$$

which yields

$$P_i a_i = 0 \tag{6.17}$$

and further gives rise to the condition (6.11). In addition, as $\mathcal{R}_i$ is a line segment set, a hard output constraint should also be applied, i.e.

$$a_i^\top r_{i-1} \leqslant a_i^\top y \leqslant a_i^\top r_i, \ \forall y \in \mathcal{R}_i, \tag{6.18}$$

to prevent overshoot. The above conditions together give the following results

$$\begin{aligned} &\hat{P}_i \tilde{r}(t) = \hat{P}_i r_i, \ t \in [t_{i-1}, \ t_i] \\ &a_i^\top r_{i-1} \leqslant a_i^\top \tilde{r}(t) \leqslant a_i^\top r_i, \ t \in [t_{i-1}, \ t_i] \end{aligned} \tag{6.19}$$

Hence, the equations (6.12), (6.13) and (6.19) imply $\tilde{r} \in \mathcal{R}_r \Rightarrow$ right half side of (6.10).

If the conditions in the right half side of (6.10) are satisfied, the increasing order of the time points is satisfied as

$$0 = t_0 < t_1 < \ldots < t_M = T \tag{6.20}$$

and it follows that

$$\tilde{r}(t_i) = r(s_i), \ i = 1, \ldots, M. \tag{6.21}$$

Since the conditions in (6.19) hold, the steps can be reversed from (6.19) backward to (6.14) to obtain

$$\tilde{r}(t) \in \mathcal{R}_i, \ t \in [t_{i-1}, \ t_i], \ i = 1, \ldots, M \tag{6.22}$$

which is equivalent to

$$\tilde{r}(t) = r(s), s_{i-1} \leqslant s \leqslant s_i, t \in [t_{i-1}, \ t_i], i = 1, \ldots, M. \tag{6.23}$$

From (6.21) and (6.23), the function $\tilde{r}(t)$ is further written as

$$\tilde{r}(t) = r(g(t)), \ 0 \leqslant t \leqslant T \tag{6.24}$$

where $g : I \to I$ is any continuous function satisfying the condition (6.8). This implies: right half side of (6.10) $\Rightarrow \tilde{r} \in \mathcal{R}_r$. □

The right side of (6.10) in Proposition 6.2 states necessary and sufficient requirements on any signal $\tilde{r}$ such that it belongs to the set $\mathcal{R}_r$ of all possible paths satisfying the spatial objective. Proposition 6.2 can hence be used to replace the tracking requirement $Gu(t) = \tilde{r}(t), \ \tilde{r} \in \mathcal{R}_r$ in (6.6) with an equivalent problem definition for the piecewise linear spatial tracking problem. Recall the extended system (5.7) defined in Chapter 5 with $F_i = I$ and $P_i = \hat{P}_i$, the equivalent problem formulation for the piecewise linear spatial tracking task is given in the next theorem.

Theorem 6.3. *The piecewise linear spatial ILC tracking problem in Definition 6.1 is equivalently stated as iteratively finding a tracking time allocation, Λ_k, and an input, u_k, with the asymptotic property such that*

$$\lim_{k\to\infty}(u_k, \Lambda_k) = (u^*, \Lambda^*) \tag{6.25}$$

where u^* and Λ^* are solutions of the problem

$$\underset{\Lambda\in\Theta,u}{\text{minimize}}\ f(u),\ \text{subject to } G^e_\Lambda u = r^e_\Lambda,\ Gu \in \Phi_\Lambda, \tag{6.26}$$

where the admissible set, Θ, for tracking time allocation, Λ, is defined as

$$\Theta = \{\Lambda \in \mathbb{R}^{M-1} : 0 < t_1 < \ldots < t_M = T\}, \tag{6.27}$$

the convex set, Φ_Λ, is defined as

$$\Phi_\Lambda = \{y \in L_2^m[0,T] : a_i^\top r_{i-1} \leqslant a_i^\top y(t) \leqslant a_i^\top r_i,\ t \in [t_{i-1}, t_i],\ i = 1,\ldots,M\} \tag{6.28}$$

and the spatial tracking reference, r^e_Λ, is defined as

$$r^e_\Lambda = [r_1, \ldots, r_M, Pr_\Lambda]^\top \tag{6.29}$$

with $r_\Lambda \in L_2^m[0,\ T]$ being a signal such that $r_\Lambda(t) = r_i$, $t \in (t_{i-1},\ t_i]$, for $i = 1,\ldots,M$.

Proof. From Proposition 6.2, it is known that the set $\mathcal{R}_r$ in Definition 6.1 is equivalently to (6.10). Using the equivalent definition (6.10) of $\mathcal{R}_r$, the problem in Definition 6.1 can be state as iteratively solving the problem

$$\begin{aligned}
&\underset{u}{\text{minimize}}\ f(u)\\
&\text{subject to}\begin{cases}
0 = t_0 < t_1 < \ldots < t_M = T\\
(Gu)(t_i) = r_i,\ i = 1,\ldots M\\
(P_iGu)(t) = P_ir_i,\ t \in [t_{i-1}, t_i]\\
a_i^\top r_{i-1} \leqslant (a_i^\top Gu)(t) \leqslant a_i^\top r_i, t \in [t_{i-1}, t_i].
\end{cases}
\end{aligned} \tag{6.30}$$

Using the linear mapping (5.2), the optimization problem (6.30) for piecewise linear path is equivalent to

$$\underset{\Lambda\in\Theta,u}{\text{minimize}}\ f(u),\ \text{subject to } (Gu)^e = r^e_\Lambda,\ Gu \in \Phi_\Lambda. \tag{6.31}$$

Consider the extended system dynamics (5.7), the problem (6.31) can be further expressed as the problem (6.26), which can be iteratively solved in practice using a suitable ILC update. Therefore, an equivalent statement of the problem in Definition 6.1 is generated in this theorem. □

6.3 A Two Stage Design Framework

Optimization problem (6.26) requires simultaneous selection of the finite time-point set Λ and plant input signal u, subject to the tracking constraints, $G^e_\Lambda u = r^e_\Lambda$ and $Gu \in \Phi_\Lambda$. Although, the cost function, $f(u)$, is independent on Λ, they are connected by these constraints, which makes the problem non-trivial. In order to solve (6.26), this requires several extensions to existing ILC frameworks reported in the literature, i.e. temporal optimization, hard constraints and projections to couple outputs while maximizing tracking freedom. Therefore, the two stage design framework proposed in Chapter 3 and the generalized ILC framework proposed in Chapter 5 are combined together to solve problem (6.26).

6.3.1 Two Stage Design Framework Description

Optimization problem (6.26) can be written equivalently as

$$\min_{\Lambda\in\Theta}\left\{\min_u f(u),\ \text{subject to } G^e_\Lambda u = r^e,\ Gu\in\Phi_\Lambda\right\} \tag{6.32}$$

by optimizing over u and then over Λ. Define the function $\tilde{f}$ of Λ by

$$\tilde{f}(\Lambda) = \min_u \left\{f(u),\ \text{subject to } G^e_\Lambda u = r^e_\Lambda,\ Gu\in\Phi_\Lambda\right\},$$

and denote a global minimizer for u of the inner optimization problem as $u_\infty(\Lambda) : \Theta \to L^\ell_2[0,\ T]$. Then optimization problem (6.32) is equivalent to

$$\min_{\Lambda\in\Theta} f(u_\infty(\Lambda)). \tag{6.33}$$

Hence the optimization problem (6.26) can be solved using the following two stage design framework:

- *Stage One:* Fix the tracking time allocation, Λ, and solve the inner optimal problem (6.32), i.e.

$$\underset{u}{\text{minimize}}\ f(u),\ \text{subject to } G^e_\Lambda u = r^e_\Lambda,\ Gu\in\Phi_\Lambda. \tag{6.34}$$

- *Stage Two:* Substitute the solution, $u_\infty(\Lambda)$, of the problem (6.34) into the original optimization problem (6.32) and then solve the resulting optimization problem (6.33) to compute the optimal tracking time allocation, i.e.

$$\min_{\Lambda\in\Theta}\{\tilde{f}(\Lambda) := f(u_\infty(\Lambda))\}. \tag{6.35}$$

In this chapter, the control effort, $\|u\|_R^2$, is considered as the target cost function to be optimized. This guarantees the existence of a unique global minimizer for u within (6.34), and solves optimization problem (6.35), i.e. $\min_{\Lambda\in\Theta} \|u_\infty(\Lambda)\|_R^2$.

6.3.2 Implementation of the Framework

6.3.2.1 Solution of Stage One

Stage One problem (6.34) involves intermediate point, sub-interval tracking and hard output constraints, plus a minimum control effort requirement. The problem (6.34) does not have a direct analytical solution, but the generalized ILC algorithm proposed in Chapter 5 can be applied to address this problem, which is illustrated in the next theorem.

Theorem 6.4. *If system $S(A, B, C)$ is controllable with C full row rank, the ILC update law in Algorithm 5.3, i.e.*

$$u_{k+1} = u_k + G_\Lambda^{s*}(I + G_\Lambda^s G_\Lambda^{s*})^{-1} e_k^s \tag{6.36}$$

followed by the projection

$$\tilde{r}_{k+1} = P_{\Phi_\Lambda}(y_{k+1}), \tag{6.37}$$

produces a solution

$$u_\infty = \lim_{k\to\infty} u_k$$

to address the optimization problem (6.34) with initial input, $u_0 = 0$, initial value, $\tilde{r}_0 \in L_2^m[0,\ T]$ and $F_i = I$, $\operatorname{Ker} P_i = \operatorname{Im} a_i$. Note that G_Λ^s is a linear operator defined by

$$G_\Lambda^s u = \begin{bmatrix} G_\Lambda^e u \\ Gu \end{bmatrix} : L_2^\ell[0,\ T] \to \bar{H} \tag{6.38}$$

whose Hilbert adjoint operator is G_Λ^{s*}, the error, e_k^s is defined as

$$e_k^s = \begin{bmatrix} e_k^e \\ \tilde{e}_k \end{bmatrix},\ e_k^e = r_\Lambda^e - y_k^e,\ \tilde{e}_k = \tilde{r}_k - y_k, \tag{6.39}$$

and $\tilde{H}$ is the Hilbert space denoted by

$$\bar{H} = \mathbb{R}^m \times \cdots \times \mathbb{R}^m \times L_2^{m-1}[t_0,\ t_1] \times \cdots \times L_2^{m-1}[t_{M-1},\ t_M] \times L_2^m[0,\ T] \tag{6.40}$$

the inner product and associated induced norm of which are naturally derived from (3.4) and (5.3).

Proof. With the assumptions made in the theorem, the problem (6.34) is a special case of the generalized ILC problem discussed in Chapter 5. Therefore, the generalized

ILC algorithm (Algorithm 5.3) addresses the problem (6.34) with desirable convergence properties by setting the appropriate parameters, i.e. $F_i = I$ and $P_i = \hat{P}_i$. □

As a controllable model can always be constructed for a given system and there is no redundant output, the assumption that system $S(A, B, C)$ is controllable with C full row rank is not restrictive. Instead of using (6.36), the solution of (3.20) can be alternatively implemented using the causal feedback plus feedforward solution, which is similar to that shown in Proposition 5.9, to further embed robust performance in practice.

Note that the hard constraint $Gu \in \Phi_\Lambda$ in problem (6.34) prevents the potential overshoot problem, i.e. the end effector moves beyond the acceptable region. In the next theorem, an alternative 'simpler' ILC update is proposed to solve problem (6.34) in the absence of the hard constraint.

Theorem 6.5. *In the absence of the constraint, $Gu \in \Phi_\Lambda$, the ILC update (6.36) followed by the projection (6.37) collapses to*

$$u_{k+1} = u_k + G_\Lambda^{e*}(I + G_\Lambda^e G_\Lambda^{e*})^{-1} e_k^e \tag{6.41}$$

which provides a global solution to the problem (6.34) with an analytical expression

$$u_\infty(\Lambda) = G_\Lambda^{e*}(G_\Lambda^e G_\Lambda^{e*})^{-1} r_\Lambda^e. \tag{6.42}$$

Proof. In the absence of the hard constraint $Gu \in \Phi_\Lambda$, the ILC update (6.36) followed by the projection (6.37) naturally collapses to the ILC update (6.41). According to Owens et al. (2015), the ILC update (6.41) ultimately converges to provide the optimal solution (6.42) to the problem (3.20) without the hard constraint, i.e.

$$\arg\min_u \|u\|_R^2, \text{ subject to } G_\Lambda^e u = r^e. \tag{6.43}$$

□

Furthermore, the solution (6.42) of the alternative simpler ILC update (6.41) guarantees that the overshoot problem will not occur in some certain class of practical systems as shown in the next lemma.

Lemma 6.6. *If the system can be represented by a chain of integrators ($A = 0$), the solution $u_\infty(\Lambda)$ satisfies $Gu_\infty(\Lambda) \in \Phi_\Lambda$.*

Proof. The proof by contradiction is used to prove that the overshoot does not happen when $A = 0$, so it is assumed that output, $Gu_\infty(\Lambda)$, has overshoot at the i^{th} line segment.

So it is necessary (but not sufficient) that there must exist a time, $t_i^* \in (t_{i-1},\ t_i)$, such that

$$(Gu^*)(t_i^*) = (Gu^*)(t_i) = r_i. \tag{6.44}$$

Based on the assumption of overshoot, an input

$$u^*(t) = \begin{cases} 0, & t \in [t_i^*,\ t_i], \\ (u_\infty(\Lambda))(t), & \text{else} \end{cases} \tag{6.45}$$

on the interval, $[0,\ T]$, can always be constructed, which provides lower input energy than $u_\infty(\Lambda)$, i.e. $\|u^*\|_R^2 < \|u_\infty(\Lambda)\|_R^2$. From the definition (6.45), it is clear that $(Gu^*)(t) = (Gu_\infty(\Lambda))(t)$ for $t \in [0,\ t_i^*]$. As $A = 0$, the following equation is satisfied

$$\begin{aligned} (Gu^*)(t) &= \int_0^{t_i^*} CBu^*(s)\mathrm{d}s + \int_{t_i^*}^{t} CBu^*(s)\mathrm{d}s \\ &= \int_0^{t_i^*} CB(u_\infty(\Lambda))(s)\mathrm{d}s \\ &= (Gu_\infty(\Lambda))(t_i^*) = r_i, \forall t \in [t_i^*,\ t_i] \end{aligned} \tag{6.46}$$

which hence leads to

$$\begin{aligned} (Gu^*)(t) &= \int_0^{t_i} CBu^*(s)\mathrm{d}s + \int_{t_i}^{t} CBu^*(s)\mathrm{d}s \\ &= (Gu_\infty(\Lambda))(t_i) + \int_{t_i}^{t} CB(u_\infty(\Lambda))(s)\mathrm{d}s \\ &= (Gu_\infty(\Lambda))(t),\ \forall t \in [t_i,\ T] \end{aligned} \tag{6.47}$$

Then, it follows that for $j = 1, \ldots, i-1,\ i+1, \ldots, M$

$$\begin{aligned} &(Gu^*)(t_j) = (Gu_\infty(\Lambda))(t_j) = r_j, \\ &(P_j Gu^*)(t) = (Gu_\infty(\Lambda))(t) = P_j r_j,\ t \in [t_{j-1},\ t_j], \end{aligned} \tag{6.48}$$

and for $j = i$

$$\begin{aligned} &(Gu^*)(t_i) = (Gu_\infty(\Lambda))(t_i) = r_i, \\ &(P_i Gu^*)(t) = (Gu_\infty(\Lambda))(t) = P_i r_i,\ t \in [t_{i-1},\ t_i^*], \\ &(P_i Gu^*)(t) = P_i r_i,\ t \in [t_i^*,\ t_i]. \end{aligned} \tag{6.49}$$

Hence the input, u^*, not only provides a lower input energy than the optimal input, $u_\infty(\Lambda)$, but also satisfies the tracking requirement, $G_\Lambda^e u = r^e$, in (6.43). This obviously contradicts the assumption that the optimal input, $u_\infty(\Lambda)$, generate overshoot in this output trajectory. Therefore, the ILC update (6.36) provides a solution satisfying the hard constraint, $Gu \in \Phi_\Lambda$. □

6.3.2.2 Solution of Stage Two

Note that the problem (6.35) is generally non-linear and non-convex with respect to the tracking time allocation, Λ, leading to significant difficulties in solving problem (6.35). Similar to the Stage Two solution in Chapter 3, the next theorem provides an iterative approach to solve the problem (6.35) experimentally.

Theorem 6.7. *The Stage Two optimization problem (3.21) is solved by the gradient projection update equation*

$$\Lambda_{j+1} = P_{\Theta}\big(\Lambda_j - \gamma_j \nabla \tilde{f}(\Lambda_j)\big) \tag{6.50}$$

where $j \in \mathbb{N}$ denotes the gradient projection loop number, the gradient, $\nabla \tilde{f}(\Lambda_j)$, is obtained using experimental measured data, P_{Θ} denotes the gradient projection optimal solution

$$P_{\Theta}(x) = \arg\min_{z}\{\|z - x\| : z \in \Theta\}, \tag{6.51}$$

and $\gamma_j > 0$ is chosen according to the generalized Armijo step size, i.e.

$$\gamma_j = \beta^{m_k}\gamma \tag{6.52}$$

where m_k is the smallest non-negative integer such that

$$\tilde{f}(\Lambda_{j+1}) - \tilde{f}(\Lambda_j) \leqslant \sigma(\nabla \tilde{f}(\Lambda_j))^{\top}(\Lambda_{j+1} - \Lambda_j) \tag{6.53}$$

and σ, β, γ are constant scalars with $0 < \sigma < 1$, $0 < \beta < 1$, $\gamma > 0$. Then the sequence $\{\tilde{f}(\Lambda_j)\}$ converges downward to a limit $\tilde{f}^*$ and every limit point of the sequence, $\{\Lambda_j\}$, is a stationary point for the problem (3.21), i.e.

$$z = P_{\Theta}(z - \nabla \tilde{f}(z)). \tag{6.54}$$

Proof. Recall Lemma 3.7, and note that all assumptions in Lemma 3.7 are satisfied, so that the gradient projection update (6.50) can be applied to the system considered in this chapter. According to Lemma 3.7, the generalized Armijo step size guarantees the convergence of the gradient projection method to a stationary point. □

It is noted that being a stationary point satisfying (6.54) is a necessary condition of a (possibly locally) minimum point. Note that the function, $\tilde{f}(\Lambda)$, is bounded below and Θ is a compact set. Therefore, the (global) minimum of the optimization problem exists and is a stationary point. When the problem only has one such point, it must be the minimum. In this case, the above algorithm converges to the global minimum solution following results in Goldstein (2012), i.e. the best result that can be achieved.

In general, the gradient within (6.50) does not have an analytical expression, and can be computed using a computationally efficient estimation

$$\left.\frac{\partial \tilde{f}}{\partial t_i}\right|_{\Lambda_j} = \frac{\tilde{f}(\Lambda_j^{i+}) - \tilde{f}(\Lambda_j^{i-})}{2\Delta T} \tag{6.55}$$

where $\Lambda_j^{i+} = [t_1^j, t_2^j, ..., t_i^j + \Delta T, ..., t_M^j]^\top$ and $\Lambda_j^{i-} = [t_1^j, t_2^j, ..., t_i^j - \Delta T, ..., t_M^j]^\top$, and $\Delta T \in \mathbb{R}$ is sufficiently small. Either experimental implementation or simulation of Stage One solution (6.36) can be used to obtain $\tilde{f}(\Lambda_j^{i+})$ and $\tilde{f}(\Lambda_j^{i-})$.

In the absence of the hard constraint, problem (3.21) can be expressed analytically as shown in the following lemma.

Lemma 6.8. *In the absence of the hard constraint $Gu \in \Phi_\Lambda$, the Stage Two optimization problem (3.21) can be expressed as*

$$\min_{\Lambda\in\Theta} \|u_\infty(\Lambda)\|_R^2 = \min_{\Lambda\in\Theta} \ \left\langle r_\Lambda^e, \ (G_\Lambda^e G_\Lambda^{e*})^{-1} r_\Lambda^e \right\rangle_{\tilde{Q}}. \tag{6.56}$$

Proof. Substituting the analytical solution (6.42) into the problem (3.21) and using the property of adjoint operator gives

$$\begin{aligned}
\min_{\Lambda\in\Theta} \|u_\infty(\Lambda)\|_R^2 &= \min_{\Lambda\in\Theta} \ \langle (u_\infty(\Lambda), \ u_\infty(\Lambda)\rangle_R \\
&= \min_{\Lambda\in\Theta} \ \left\langle (G_\Lambda^{e*}(G_\Lambda^e G_\Lambda^{e*})^{-1} r_\Lambda^e, G_\Lambda^{e*}(G_\Lambda^e G_\Lambda^{e*})^{-1} r_\Lambda^e \right\rangle_R \\
&= \min_{\Lambda\in\Theta} \ \left\langle G_\Lambda^e G_\Lambda^{e*}(G_\Lambda^e G_\Lambda^{e*})^{-1} r_\Lambda^e, (G_\Lambda^e G_\Lambda^{e*})^{-1} r_\Lambda^e \right\rangle_{\tilde{Q}} \\
&= \min_{\Lambda\in\Theta} \ \left\langle r_\Lambda^e, \ (G_\Lambda^e G_\Lambda^{e*})^{-1} r_\Lambda^e \right\rangle_{\tilde{Q}}
\end{aligned}$$

which completes the proof. □

According to Lemma 6.8, in the absence of the hard constraint, the gradient within (6.50) can be computed analytically by performing discretization of the cost function in (6.56) and calculating its partial derivatives.

Remark 6.9. The choice of initial allocation Λ_0 affects the convergence performance of update (6.50). It can be chosen arbitrarily from Θ if no information is available. Alternatively, Λ_0 can be selected using the methods introduced in Chapter 3.3.2.

6.3.3 An Iterative Implementation Algorithm

The experimental implementations of Stage One and Stage Two of the two stage design framework can be integrated to yield a practical iterative algorithm. In Algorithm 6.10,

Λ_0 is a suitable initial tracking time allocation, and ϵ and δ are sufficiently small positive scalars specifying the tracking precision and performance requirements.

Algorithm 6.10. Given initial tracking time allocation, Λ_0, system state space model, $S(A, B, C)$, desired extended tracking reference, r^e, admissible set of tracking time allocation, Θ, weighting matrices, R, Q_i and $\hat{Q}_i$, the following steps provides the solution to the optimal tracking time allocation, Λ_{opt}, and input, u_{opt}

1: **Initialization:** Loop number $j = 0$
2: Repeatedly implement Stage One update (6.36) followed by the projection (6.37) with $\Lambda = \Lambda_0$ using feedback plus feedforward update (3.53) until convergence, i.e. $\|e_k^e\| < \epsilon$; record converged input, $u_\infty^{ex}(\Lambda_0)$, and input energy, $\tilde{f}(\Lambda_0)$.
3: **repeat**
4: Implement Stage Two update (6.50) with the gradient estimation equation (6.55) using $r_i = G_i u_\infty^{ex}(\Lambda_j)$.
5: Set $j \to j + 1$.
6: Repeatedly implement Stage One update (6.36) followed by the projection (6.37) with $\Lambda = \Lambda_j$ using feedback plus feedforward update (3.53) until convergence, i.e. $\|e_k^e\| < \epsilon$; record converged input, $u_\infty^{ex}(\Lambda_j)$, and input energy, $\tilde{f}(\Lambda_j)$.
7: **until** $\left|\tilde{f}(\Lambda_j) - \tilde{f}(\Lambda_{j-1})\right| < \delta \left|\tilde{f}(\Lambda_{j-1})\right|$.
8: **return** $\Lambda_{opt} = \Lambda_j$ and $u_{opt} = u_\infty^{ex}(\Lambda_j)$.

To implement Algorithm 6.10, Step 2 and 6 (i.e. norm-optimal algorithm) must be implemented experimentally and Step 4 must use experimental data, $u_\infty^{ex}(\Lambda_j)$. These requirements are not necessary if an accurate system model is available, but model mismatch and uncertainty exists widely in practice. Therefore, the proposed algorithm embeds significant robustness properties as it exploits experimental plant data.

6.4 Constrained Input Condition Handling

The previous section uses the Two Stage design framework to solve the spatial ILC problem with an additional cost function. This section further extends the proposed algorithm to incorporate system constraints into the design.

6.4.1 Modified Two Stage Design Framework

With input constraints, the optimal problem (6.26) becomes

$$\begin{aligned} \underset{u,\Lambda}{\text{minimize}} \quad & f(u) \\ \text{subject to} \quad & r^e = G^e_\Lambda u, \\ & Gu \in \Phi_\Lambda, \\ & \Lambda \in \Theta,\ u \in \Omega. \end{aligned} \tag{6.57}$$

Then, following a similar procedure to that in Section 6.3, the constrained optimization problem (6.57) can be equivalently written as

$$\min_{\Lambda\in\Theta} \left\{ \min_u f(u),\ \text{s.t. } G^e_\Lambda u = r^e_\Lambda,\ Gu \in \Phi_\Lambda,\ u \in \Omega \right\} \tag{6.58}$$

suggesting a possible two stage design framework:

- *Stage One:*

$$\begin{aligned} \underset{u\in\Omega}{\text{minimize}} \quad & \|u\|^2_R \\ \text{subject to} \quad & r^e_\Lambda = G^e_\Lambda u \quad Gu \in \Phi_\Lambda \end{aligned} \tag{6.59}$$

  whose the solution is denoted as $\hat{u}_\infty(\Lambda)$.

- *Stage Two:*

$$\min_{\Lambda\in\Theta}\{\tilde{f}(\Lambda) := \|\hat{u}_\infty(\Lambda)\|^2_R\}. \tag{6.60}$$

Although the presence of the input constraints adds some difficulties in solving Stage One problem (6.59), this problem can be also solved using the exact form of Algorithm 5.3, i.e.

$$\tilde{u}_{k+1} = u_k + G^{s*}_\Lambda (I + G^s_\Lambda G^{s*}_\Lambda)^{-1} e^s_k \tag{6.61}$$

followed by the projection

$$\tilde{r}_{k+1} = P_{\Phi_\Lambda}(y_{k+1}),\ u_{k+1} = P_\Omega(\tilde{u}_{k+1}). \tag{6.62}$$

The Stage Two optimization problem (6.60) can be solved using the following iterative approach:

$$\Lambda_{j+1} = \arg\min_{\Lambda\in\Theta} \tilde{f}_j(\Lambda) \tag{6.63}$$

where

$$\tilde{f}_j(\Lambda) = \|u_\infty(\Lambda)\|^2_R + \rho \|u_\infty(\Lambda) - \hat{u}_\infty(\Lambda_j)\|^2_R,\ \rho \geqslant 0, \tag{6.64}$$

with $u_\infty(\Lambda)$ and $\hat{u}_\infty(\Lambda)$ denoting the analytical solution (6.42) and the converged input of Stage One problem (6.59) respectively for Λ_j. Note that in problem (3.76), both input

and output constraints are decoupled from the optimization problem as the previous obtained Stage One solution, $\hat{u}_\infty(\Lambda_j)$, can be considered as a known constant value in this problem. Therefore, the problem (3.76) can be solved analytically using the update (6.50) in Theorem 6.7.

6.4.2 Modified Iterative Implementation Algorithm

Combining the solutions of Stage One and Stage Two yields a modified algorithm (Algorithm 6.11) to solve the spatial tracking problem (6.58) with input constraints.

Algorithm 6.11. Given initial tracking time allocation, Λ_0, system state space model, $S(A, B, C)$, desired extended tracking reference, r^e, admissible set of tracking time allocation, Θ, weighting matrices, R, Q_i and $\hat{Q}_i$, the following steps provides the solution to the optimal tracking time allocation, Λ_{opt}, and input, u_{opt}

1: **Initialization:** Loop number $j = 0$
2: Repeatedly implement Stage One update (6.61) followed by the projection (6.62) with $\Lambda = \Lambda_0$ using feedback plus feedforward update (3.53) until convergence, i.e. $\|e_k^e\| < \epsilon$; record converged input, $u_\infty^{ex}(\Lambda_0)$, and input energy, $\tilde{f}(\Lambda_0)$.
3: **repeat**
4: Compute the gradient using (6.42) or (6.55) with $r^e = G^e_{\Lambda_j} \hat{u}_\infty^{ex}(\Lambda_j)$; implement Stage Two update (6.63).
5: Set $j \to j + 1$.
6: Repeatedly implement Stage One update (6.61) followed by the projection (6.62) with $\Lambda = \Lambda_j$ using feedback plus feedforward update (3.53) until convergence, i.e. $\|e_k^e\| < \epsilon$; record converged input, $u_\infty^{ex}(\Lambda_j)$, and input energy, $\tilde{f}(\Lambda_j)$.
7: **until** $\left|\tilde{f}(\Lambda_j) - \tilde{f}(\Lambda_{j-1})\right| < \delta \left|\tilde{f}(\Lambda_{j-1})\right|$.
8: **return** $\Lambda_{opt} = \Lambda_j$ and $u_{opt} = u_\infty^{ex}(\Lambda_j)$.

Algorithm 6.11 has the following convergence properties:

Theorem 6.12. *Suppose perfect tracking is achievable and $\rho \leqslant 1$, the analytical input energy resulting from (6.63) satisfies*

$$\|u_\infty(\Lambda_{j+1})\|_R^2 \leqslant \|\hat{u}_\infty(\Lambda_j)\|_R^2 . \tag{6.65}$$

Proof. The proof of this theorem follows from the proof of Theorem 3.15. □

6.5 Experimental Verification on A Gantry Robot

In this section, the proposed algorithm is validated experimentally on a three-axis gantry robot test platform to demonstrate its effectiveness.

6.5.1 Design Task Specification

The multi-axis gantry robot shown in Chapter 3.5.1 with the system model (3.83) is employed as a test platform assuming two decimal place accuracy on the tracking time allocation in this section. The gantry robot's input is the voltage, and output is the displacement of the three axes. The control design objective is to use all three axes ($m = 3$) to perform a piecewise linear spatial tracking task with five line segments ($M = 5$). The total tracking time is $T = 2s$, and the transition vertices are shown in Figure 6.2.

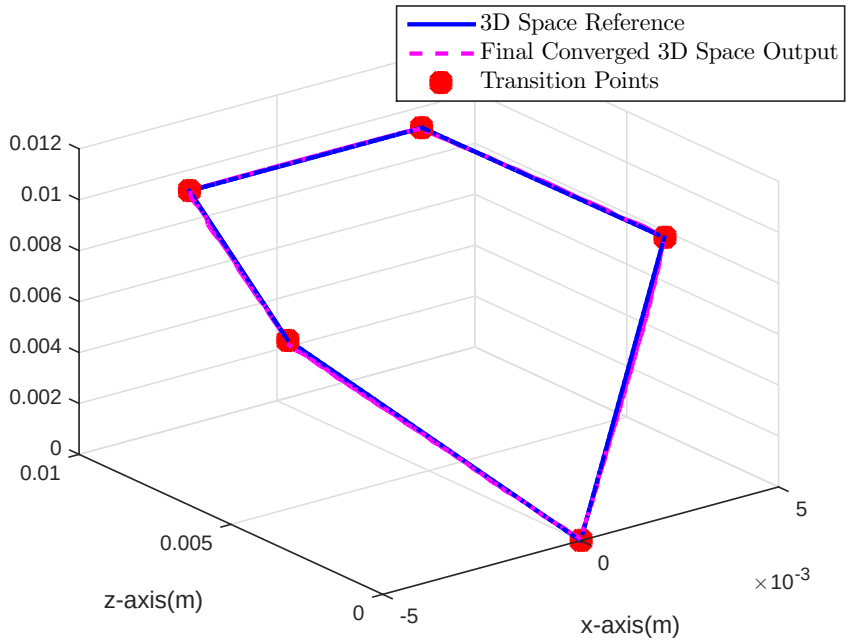


Figure 6.2: Spatial Output for Robustness Test.

For implementational simplicity, the weighting matrices are chosen to be $Q_i = q_i I$, $\hat{Q}_i = \hat{q}_i I$ and $R = \hat{r} I$ where q_i, $\hat{q}_i$ and $\hat{r}$ are positive scalars. Furthermore, appropriate weighting matrices are chosen according to the theoretical predictions in Owens et al. (2013) to balance convergence speed and robust performance, i.e. $q_i/\hat{r} = 30,000$ and $\hat{q}_i/\hat{r} = 3,000,000$, together with $\sigma = 0.1$, $\beta = 0.5$, $\gamma = 0.07$ in Step 4 of Algorithm 6.10.

6.5.2 Performance of the Proposed Algorithm

In practice, it is difficult to identify an accurate system model. Hence it is necessary to test the robust performance of Algorithm 6.10 with significant model uncertainty. Assume that only three inaccurate models (for x-axis, y-axis and z-axis respectively) are available for design as follows:

$$\hat{G}_x(s) = \frac{0.04}{s},\ \hat{G}_y(s) = \frac{0.04}{s} \text{ and } \hat{G}_z(s) = \frac{0.04}{s}. \tag{6.66}$$

Algorithm 6.10 is applied with an (*a priori*) allocation $\Lambda_0 = [0.4, 0.8, 1.2, 1.6]^\top$ over 25 loops with $\gamma = 0.07, 0.08, 0.09$ respectively. This is compared with the simulation result using models in (6.66), which provides the theoretical optimal allocation $\Lambda^* = [0.51, 0.84, 1.16, 1.49]^\top$ and corresponding experimental minimum energy, $\tilde{f}(\Lambda^*) = 598.7$. The results are shown in Figure 6.3 and 6.4 respectively.

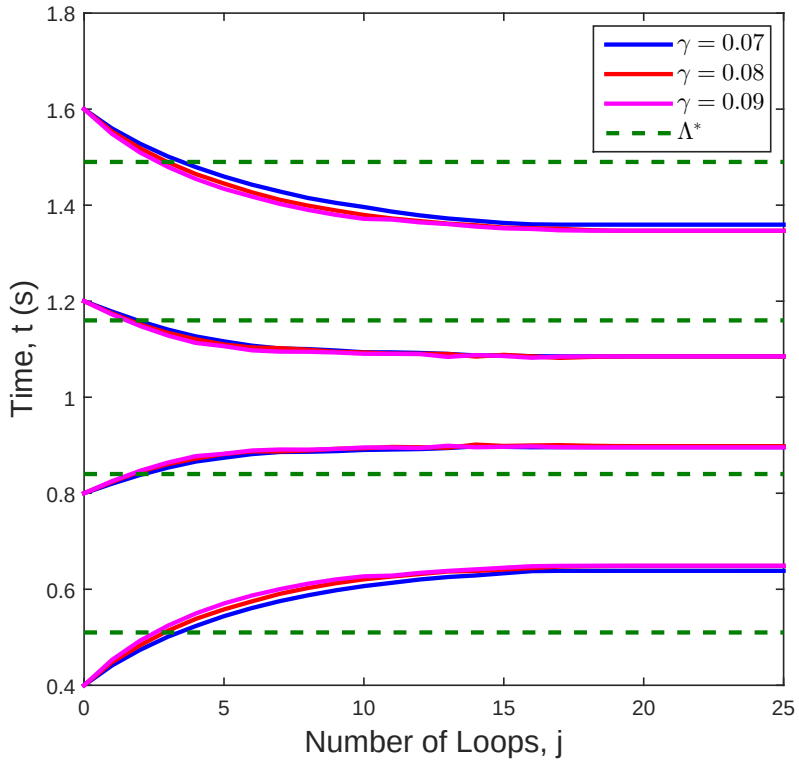


Figure 6.3: Optimal Tracking Time Allocation during Robustness Test.

In Figure 6.3, the blue, red, and magenta curves denote the tracking time allocation Λ_j at each loop for $\gamma = 0.07, 0.08, 0.09$ respectively with the final converged value $\Lambda_{opt} = [0.65, 0.90, 1.09, 1.36]^\top$, and the green dashed lines denote the theoretical optimal allocation, Λ^*. In Figure 6.4, the blue, red, and magenta curves denote the minimum input energy, $\tilde{f}(\Lambda_k)$, at each loop for $\gamma = 0.07, 0.08, 0.09$ respectively with the corresponding minimum energy value 545.1 and the green line is the minimum energy, $\tilde{f}(\Lambda^*)$,

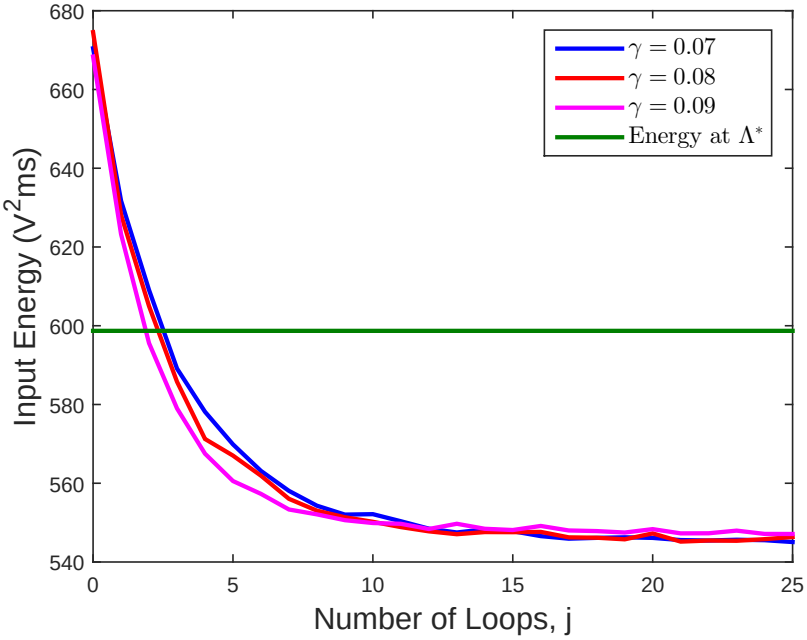
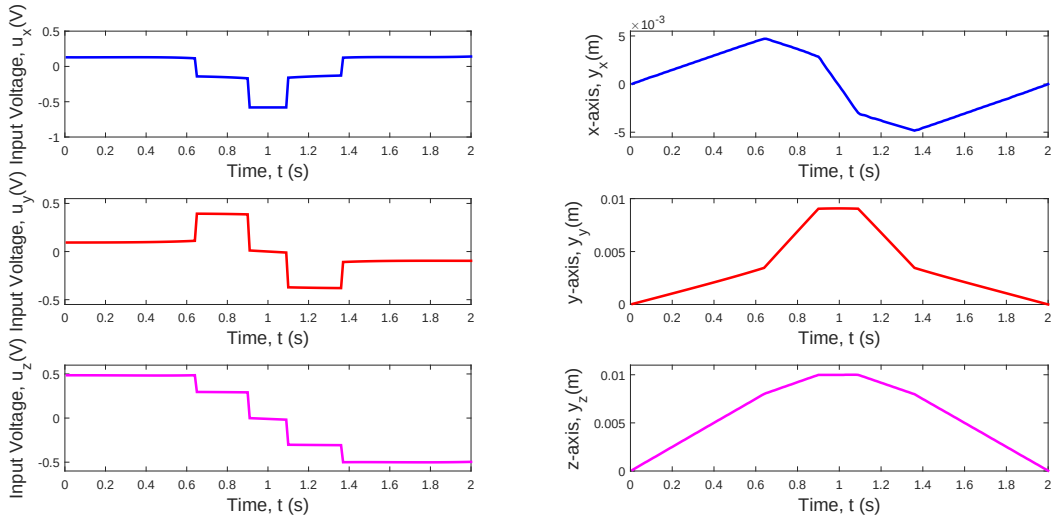


Figure 6.4: Input Energy during Robustness Test.

needed at Λ^*. Note that this is 8.95% less than the value at Λ^* calculated using the (inaccurate) system model, which clearly demonstrates the advantage of the proposed algorithm.



(a) Input Trajectories

(b) Output Trajectories

Figure 6.5: Final Converged Results for a) Input and b) Output.

The experimental converged spatial output trajectory for Λ_{opt} is also shown in Figure 6.2 with the red dots denoting the transition vertices. It can be clear that the spatial output accurately tracks the vertices and satisfies the linear projection constraints between them. For more information, the final converged input and output trajectories

Table 6.1: Summary of Experimental Results with Different Q_i, $\hat{Q}_i$ and R.

	Λ_{opt}	$\tilde{f}(\Lambda_{opt})$	Difference from $\tilde{f}(\Lambda^*)$
$q_i/\hat{r} = 30,000$, $\hat{q}_i/\hat{r} = 3,000,000$	$[0.65, 0.90, 1.09, 1.36]^\top$	545.1	8.95 %
$q_i/\hat{r} = 30,000$, $\hat{q}_i/\hat{r} = 5,000,000$	$[0.65, 0.90, 1.09, 1.36]^\top$	546.8	8.67 %
$q_i/\hat{r} = 30,000$, $\hat{q}_i/\hat{r} = 8,000,000$	$[0.65, 0.90, 1.09, 1.35]^\top$	547.3	8.59 %
$q_i/\hat{r} = 30,000$, $\hat{q}_i/\hat{r} = 10,000,000$	$[0.66, 0.90, 1.09, 1.35]^\top$	546.5	8.72 %
$q_i/\hat{r} = 50,000$, $\hat{q}_i/\hat{r} = 3,000,000$	$[0.65, 0.90, 1.09, 1.35]^\top$	545.9	8.82 %
$q_i/\hat{r} = 80,000$, $\hat{q}_i/\hat{r} = 3,000,000$	$[0.65, 0.90, 1.09, 1.36]^\top$	544.7	9.02 %
$q_i/\hat{r} = 100,000$, $\hat{q}_i/\hat{r} = 3,000,000$	$[0.66, 0.90, 1.09, 1.36]^\top$	546.6	8.70 %

for each axis are shown in Figure 6.5. It is concluded that the proposed algorithm can minimize the input energy and maintain sufficient tracking performance in the presence of significant model uncertainty.

Furthermore, experiments with different values of Q_i, $\hat{Q}_i$ and R have been carried out with respect to $\gamma = 0.07$. The corresponding results are summarized in Table 6.1, and it is clear from the table that the optimal tracking time allocations obtained by the proposed algorithm all converge, and are different from the corresponding theoretical obtained one, Λ^*, as the system model (6.66) is not accurate enough. It is clear that the obtained input energy results at each case are approximately 9 % less than the value at Λ^*, so it can be concluded that the proposed algorithm can minimize the input energy and has a certain degree of robustness against model uncertainty. Experiments using other initial allocations, e.g. low resolution initial allocation, will also be performed as a future work.

6.6 Summary

This chapter considers ILC design for spatial path tracking problem to allow an additional cost function as well as high performance path tracking. This design problem is formulated into an equivalent optimization problem. To solve this non-trivial design problem, the two stage framework proposed in Chapter 3 is considered, which consists

of the generalized ILC update proposed in Chapter 5 and a gradient projection update. Then, the two stage framework yields the first spatial ILC algorithm capable of an additional cost function whilst completing the spatial path tracking objective for a general class of systems. This algorithm has been verified experimentally on a gantry robot, demonstrating significant benefits in terms of reducing the control energy needed to travel along the given path.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Extended ILC frameworks have emerged in recent years to tackle the rigidity in the traditional ILC trajectory tracking formation. This thesis addresses the unmet potential existing within the class of extended ILC frameworks including point-to-point ILC and spatial ILC, which have attracted significant research interest. In previous point-to-point ILC formulations, the tracking time allocation of the critical tracking positions was considered as given *a priori*, meaning that the design freedom of point-to-point ILC was not fully exploited. More generally, prior attempts at spatial ILC were application specific, and each could only be applied to a particular, limited class of systems. Also these attempts did not fully harness the temporary design freedom in choosing an optimal solution to achieve an additional optimal cost function as well as perfect tracking. Furthermore, all previous research on constrained ILC focused on specific classes of ILC problems with a single constraint, and no attempt was made to solve the ILC problem with mixed forms of system constraints for a general class of systems.

The work in Chapter 3 significantly expanded the point-to-point ILC framework to allow the tracking time allocation to optimize some target cost function as well as maintaining precise tracking. The subsequent ILC problem was first formulated into optimization problem (3.15), and a Two Stage design framework was then proposed to make the non-trivial problem tractable. By assuming the control effort as the target cost function, an iterative algorithm (Algorithm 3.12) with desired convergence properties has been proposed based on the solutions of the two stages. Furthermore the input constraint was also incorporated to yield a constrained algorithm (Algorithm 3.14), which adapts to practical working environment of the system. In doing so, this chapter provided the first solution to the point-to-point ILC problem. This is significant in the field of ILC and robotics since it combines the accuracy of ILC with the practically related flexibility of

path planning techniques typically used in robotics. It also shows how the six postulates in ILC can be relaxed without sacrificing mathematical rigor.

Chapter 3 only considered the tracking of a finite number of critical positions, and an ILC framework capable of handling a broader class of spatial ILC problems was therefore needed. Therefore, a generalized ILC framework was formulated in Chapter 5 which was applicable to various tracking problems including spatial path tracking for a general class of systems with a mixed form of system constraints. By choosing appropriate coefficients, this single framework can solve a set of well-known ILC problems, e.g. classical ILC, point-to-point ILC and spatial ILC, as special cases. A generalized ILC algorithm (Algorithm 5.3) with guaranteed convergence properties was derived, which can be efficiently implemented in practice to perform various spatial path tracking tasks, e.g. welding, laser cutting, robotic manufacturing and even rehabilitation. This chapter illustrated how successive projection techniques can be integrated with ILC to expand the problem scope. It also further relaxes the standard ILC postulates to capture even wider practical applications.

Although the generalized ILC framework can optimize a cost function for a wide class of systems, the tracking time allocation in this framework is considered as *a priori*. This potential limitation motivated the work in Chapter 6 to further employ the tracking time allocation as an optimized variable enabling it to handle spatial path tracking tasks with an additional cost function. This can be considered as the parallel extension to the task description of the work in Chapter 3. The general spatial path tracking problem was first described and it was then specialized to a piecewise linear path tracking problem as a special case. The characterized problem was reformulated into an equivalent problem definition, which could be addressed using the Two Stage design framework proposed in Chapter 3. An algorithm (Algorithm 6.10) was derived based on the generalized ILC algorithm to solve the problem with minimum control effort, and a constrained algorithm (Algorithm 6.11) was also derived to handle the practical case with input constraints. This chapter shows that the relaxation of the standard ILC postulates fully exploits the temporary design freedom in spatial tracking tasks, enabling practical benefits to be gained, such as reducing the energy used, reducing the damage to machine components and increasing the efficiency of production (i.e. throughput), while reserving the spatial tracking requirements.

All algorithms proposed in this thesis have solid theoretical derivations, but their experimental verification is also needed to verify the practical performance. A multi-axis gantry robot system was used as a test platform in this thesis, and all the proposed algorithms were verified on this platform. The experimental results shows that all algorithms have significant practical efficacy and can provide high accuracy tracking in practice. The results also show that the proposed algorithms exhibit a degree of robustness against modelling mismatch/error due to the use of the measured real data, which is clearly desirable in practical applications. The experimental verification on

the gantry robot has significant practical relevance, as this platform can fully replicate industrial working environments including pick-and-place task specification, interface between software and hardware, model uncertainty and random disturbance.

7.2 Future Work

This thesis contributes significantly to generalizing the ILC task description in order to achieve more performance and flexibilities. However, the current work presented in this thesis can be further developed as described below.

Spatial ILC with General Path: The research in Chapter 6 considers the piecewise linear path as a special case of the general spatial path. In the future research, the proposed spatial ILC framework will be expanded to handle the general spatial path tracking problem, which significantly broadens the application range of the spatial ILC framework. Note that any general path can be divided into several joint convex curves, so a general spatial path can be regarded as a piecewise convex path. Some initial results have been formulated on how to solve the piecewise convex path tracking problem. For example, considering the piecewise arc path in a 2D plane as a special case, there exists a bijective map called a Mobius transformation, which maps all the points along an arc path to a linear path. Using this map, the piecewise arc path tracking problem can be converted into the piecewise linear path tracking problem without any difficulties, and the proposed spatial ILC framework in Chapter 6 can be then applied to solve the piecewise arc path tracking problem.

Alternative Cost Functions: Note that control effort is considered as the target cost function to be optimized in this thesis. However, in principle, the setup of the proposed Two Stage design framework allows the optimization of various cost functions beside the control effort. In future work, other alternative cost functions, such as the minimum tracking time, will be considered, and corresponding algorithms will be derived.

Furthermore, most practical manufacturing process usually require optimization of multiple cost functions, and some cost functions conflict with one another. In future work, the design framework will also be further generalized to handle the optimization of multiple cost functions. The technique of pareto optimization will be applied to reach a trade-off in the total cost function if there exits more than one conflicting cost function at the same time.

Robustness Analysis and Robust Design: All algorithms proposed in this thesis need a nominal system model to provide solutions, however, it is nearly impossible for the nominal system model to be exactly the same as the plant in practice. As a result, the algorithms cannot perform well if the nominal system model is not significantly accurate. Although the experimental results demonstrate that the algorithms have a

certain degree of robustness against model uncertainties, a rigorous robustness analysis will be carried out in the future to verify the range of model uncertainty such that these algorithms have the guaranteed performance properties and stability. On the other side, robust algorithm design will also be conducted such that the resulting algorithms maintain robust performance properties within given sets of model uncertainty. For example, multiplicative modelling error can be used in the algorithm design following the method introduced in Owens et al. (2014).

References

H.-S. Ahn, Y. Chen, and K. L. Moore. Iterative learning control: brief survey and categorization. *IEEE Transaction on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 37(6):1099–1121, 2007.

H.-S. Ahn, K. L. Moore, and Y. Chen. Kalman filter-augmented iterative learning control on the iteration domain. In *American Control Conference*, pages 250–255, 2006a.

H.-S. Ahn, K. L. Moore, and Y. Chen. Monotonic convergent iterative learning controller design based on interval model conversion. *IEEE Transactions on Automatic Control*, 51(2):366371, 2006b.

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

N. Amann, D. H. Owens, and E. Rogers. Iterative learning control for discrete-time systems with exponential rate of convergence. *IEE Proceedings of the Control Theory and Applications*, 143(2):217–244, 1996a.

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, 1996b.

S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of dynamic system by learning: A new control theory for servomechanism or mechatronics systems. In *Proceedings of 23rd Conference on Decision and Control, Las Vegas*, pages 1064–1069, 1984a.

S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operations of robots by learning. *Journal of Robotic Systems*, 1(2):123–140, 1984b.

S. Arimoto, S. Kawamura, F. Miyazaki, and S. Tamaki. Learning control theory for dynamical systems. In *Proceedings of 24rd Conference on Decision and Control*, pages 1375–1380, 1985.

L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

K. Barton and D. Kingston. Systematic surveillance for uavs: A feedforward iterative learning control approach. In *American Control Conference*, pages 5917–5922, 2013.

D. P. Bertsekas. On the Goldstein - Levitin - Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21(2):174–184, 1976.

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

E. G. Birgin, J. M. Martinez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

J. Bolder and T. Oomen. Inferential iterative learning control: A 2D-system approach. *Automatica*, 71:247–253, 2016.

J. Bolder, T. Oomen, S. Koekebakker, and M. Steinbuch. Using iterative learning control with basis functions to compensate medium deformation in a wide-format inkjet printer. *Mechatronics*, 24(8):944–953, 2014.

D. A. Bristow, M. Tharayil, and A. G. Alleyne. A survey of iterative learning control. *IEEE Control Systems Magzine*, 26(3):96–114, 2006.

Z. Cai, C. T. Freeman, P. L. Lewin, and E. Rogers. Iterative learning control for a non-minimum phase plant based on a reference shift algorithm. *Control Engineering Practice*, 16(6):633–643, 2008.

Y. Chen, B. Chu, and C. T. Freeman. Point-to-point iterative learning control with optimal tracking time allocation. In *54th IEEE Conference on Decision and Control*, pages 6089–6094, Osaka, Japan, 2015.

Y. Chen, B. Chu, and C. T. Freeman. Spatial path tracking using iterative learning control. In *55th IEEE Conference on Decision and Control*, pages 7189–7194, Las Vegas, US, 2016.

Y. Chen, B. Chu, and C. T. Freeman. Generalized norm optimal iterative learning control: Constraint handling. In *The 20th World Congress of the International Federation of Automatic Control*, Toulouse, France, 2017a.

Y. Chen, B. Chu, and C. T. Freeman. Point-to-point iterative learning control with optimal tracking time allocation. *IEEE Transactions on Control Systems Technology*, 2017b.

Y. Chen, B. Chu, and C. T. Freeman. Point-to-point iterative learning control with optimal tracking time allocation: A coordinate descent approach. In *36th Chinese Control Conference*, Dalian, China, 2017c.

Y. Chen and K. L. Moore. An optimal design of PD-type iterative learning control with monotonic convergence. In *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*, pages 55–60, 2002.

Y. Chen, C. Wen, Z. Gong, and M. Sun. An iterative learning controller with initial state learning. *IEEE Transactions on Automatic Control*, 44(2):371–376, 1999.

R. Chi, Z. Hou, and J.-X. Xu. Adaptive ILC for a class of discrete-time systems with iteration-varying trajectory and random initial condition. *Automatica*, 44(8):2207–2213, 2008.

B. Chu, C. T. Freeman, and D. H. Owens. A novel design framework for point-to-point ILC using successive projection. *IEEE Transactions on Control Systems Technology*, 23(3):1156–1163, 2015.

B. Chu and D. H. Owens. Accelerated norm-optimal iterative learning control algorithms using successive projection. *International Journal of Control*, 82(8):1469–1484, 2009.

B. Chu and D. H. Owens. Iterative learning control for constrained linear systems. *International Journal of Control*, 83(7):1397–1413, 2010.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Steinn. *Introduction to Algorithms*. The MIT Press, Massachusetts Institute of Technology, third edition, 2009.

D. de Roover and O. H. Bosgra. Synthesis of robust multivariable iterative learning controllers with application to a wafer stage motion system. *International Journal of Control*, 73(10):914–929, 2000.

H. Ding and J. Wu. Point-to-point control for a high-acceleration positioning table via cascaded learning schemes. *IEEE Transactions on Industrial Electronics*, 54(5): 2735–2744, 2007.

T. Donkers, J. van de Wijdeven, and O. Bosgra. Robustness against model uncertainties of norm optimal iterative learning control. In *Americal Control Conference*, pages 4561–4566, 2008.

E. Fornasini and G. Marchesini. Doubly-indexed dynamical systems: State-space models and structural properties. *Mathematical Systems Theory*, 12:59–72, 1978.

C. T. Freeman. Constrained point-to-point iterative learning control with experimental verification. *Control Engineering Practice*, 20(5):489–498, 2012.

C. T. Freeman. Robust ILC design with application to stroke rehabilitation. *Automatica*, 81:270–278, 2017.

C. T. Freeman, Z. Cai, E. Rogers, and P. L. Lewin. Iterative learning control for multiple point-to-point tracking application. *IEEE Transactions on Control Systems Technology*, 19(3):590–600, 2011.

C. T. Freeman and T. V. Dinh. Experimentally verified point-to-point iterative learning control for highly coupled systems. *International Journal of Adaptive Control and Signal Processing*, 29:302–324, 2015.

C. T. Freeman, A.-M. Hughes, J. H. Burridge, P. H. Chappell, P. L. Lewin, and E.Rogers. Iterative learning control of FES applied to the upper extremity for rehabilitation. *Control Engineering Practice*, 17(3):368–381, 2009.

C. T. Freeman, P. L. Lewin, and E. Rogers. Experimental evaluation of iterative learning control algorithms for non-minimum phase plants. *International Journal of Control*, 78(11):826–846, 2005.

C. T. Freeman, P. L. Lewin, and E. Rogers. Further results on the experimental evaluation of iterative learning control algorithms for non-minimum phase plants. *International Journal of Control*, 80(4):569–582, 2007.

C. T. Freeman and Y. Tan. Iterative learning control with mixed constraints for point-to-point tracking. *IEEE Transactions on Control Systems Technology*, 21(3):604–616, 2013.

M. French. Robust stability of iterative learning control schemes. international journal of robust and nonlinear control. *International Journal of Robust and Nonlinear Control*, 18(10):1018–1033, 2008.

G. Gauthier and B. Boulet. Robust design of terminal ILC with mixed sensitivity approach for a thermoforming oven. *Journal of Control Science and Engineering*, 2008. Article ID 289391.

X. Ge, J. L. Stein, and T. Ersal. Frequency-domain analysis of robust monotonic convergence of norm-optimal iterative learning control. *IEEE Transactions on Control Systems Technology*, PP(99):1–15, 2017.

A. A. Goldstein. *Constructive Real Analysis*. Dover Publications, 2012.

L.G. Gubin, B.T. Polyak, and E.V. Raik. The method of projections for finding the common point of convex sets. *USSR Computational Mathematics and Mathematical Physics*, 7(6):1–24, 1967.

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

T. J. Harte. *Discrete-time model-based iterative learning control: Stability, monotonicity and robustness*. PhD thesis, University of Sheffield, Sheffield, 2007.

T. J. Harte, J. Hatonen, and D. H. Owens. Discrete-time inverse model-based iterative learning control: Stability, monotonicity and robustness. *International Journal of Control*, 78(8):577–586, 2005.

L. Hladowski, K. Galkowski, Z. Cai, E. Rogers, C. T. Freeman, and P. L. Lewin. Experimentally supported 2D systems based iterative learning control law design for error convergence and performance. *Control Engineering Practice*, 18:339–348, 2010.

L. Hladowski, K. Galkowski, Z. Cai, E. Rogers, C. T. Freeman, and P. L. Lewin. A 2D systems approach to iterative learning control for discrete linear processes with zero markov parameters. *International Journal of Control*, 84(7):1246–1262, 2011.

D. J. Hoelzle and K. L. Barton. On spatial iterative learning control via 2-D convolution: Stability analysis and computational efficiency. *IEEE Transactions on Control Systems Technology*, 24(4):1504–1512, 2016.

D. Huang, J.-X. Xu, V. Venkataramanan, and T. C. T. Huynh. High-performance tracking of piezoelectric positioning stage using current-cycle iterative learning control with gain scheduling. *IEEE Transactions on Industrial Electronics*, 61(2):1085–1098, 2014.

P. Janssens, W. V. Loock, G. Pipeleers, F. Debrouwere, and J. Swevers. Iterative learning control for optimal path following problems. In *52nd IEEE Conference on Decision and Control*, pages 6670 – 6675, Florence, Italy, 2013a.

P. Janssens, G. Pipeleers, M. Diehl, and J. Swevers. Energy optimal time allocation of a series of point-to-point motions. *IEEE Transactions on Control Systems Technology*, 22(6):2432–2435, 2014.

P. Janssens, G. Pipeleers, and J. Swevers. A data-driven constrained norm-optimal iterative learning control framework for LTI systems. *IEEE Transactions on Control Systems Technology*, 21(2):546–551, 2013b.

P. Jiang and R. Unbehauen. An iterative learning control scheme with deadzone. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 3816–3817, 1999.

X. Jin. Iterative learning control for non-repetitive trajectory tracking of robot manipulators with joint position constraints and actuator faults. *International Journal of Adaptive Control and Signal Processing*, 31, 2016.

X. Jin and J.-X. Xu. Iterative learning control for output-constrained systems with both parametric and nonparametric uncertainties. *Automatica*, 49(8):2508–2516, 2013.

X. Li, J.-X. Xu, and D. Huang. An iterative learning control approach for linear systems with randomly varying trial lengths. *IEEE Transactions on Automatic Control*, 59 (7):1954–1960, 2014.

I. Lim and K. L. Barton. Pareto iterative learning control: Optimized control for multiple performance objectives. *Control Engineering Practice*, 26:125–135, 2014.

I. Lim, D. J. Hoelzle, and K. L. Barton. A multi-objective iterative learning control approach for additive manufacturing applications. *Control Engineering Practice*, 64: 74–87, 2017.

T. Lin, D. H. Owen, and J. Hatonen. Newton method based iterative learning control for discrete non-linear systems. *International Journal of Control*, 79(10):1263–1276, 2006.

T. Lippa and S. Boyd. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6):1297–1311, 2014.

S. Mishra, U. Topcu, and M. Tomizuka. Optimization-based constrained iterative learning control. *IEEE Transactions on Control Systems Technology*, 19(6):1613–1621, 2011.

J. H. Moon, T. Y. Doh, and M. J. Chung. A robust approach to iterative learning control design for uncertain systems. *Automatica*, 34(8):1001–1004, 1998.

K. L. Moore. *Iterative Learning Control for Deterministic Systems*. London: Springer-Verlag, 1993.

K. L. Moore, Y. Q. Chen, and V. Bahl. Monotonically convergent iterative learning control for linear discrete-time systems. *Automatica*, 41(9):1529–1537, 2005.

K. L. Moore, M. Ghosh, and Y. Q. Chen. Spatial-based iterative learning control for motion control applications. *Meccanica*, 42:167–175, 2007.

M. Norrlof. *Iterative learning control, analysis, design and experiments.* PhD thesis, Linkoping University, Linkoping, 2000.

T. Oomen, J. de Wijdeven, and O. Bosgra. Suppressing intersample behavior in iterative learning control. *Automatica*, 45(4):981–988, 2009.

J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics, 2000.

D. H. Owens and B. Chu. Combined inverse and gradient iterative learning control: Performance, monotonicity, robustness and non-minimum-phase zeros. *International Journal of Robust and Nonlinear Control*, 24(3):406–431, 2014.

D. H. Owens and K. Feng. Parameter optimization in iterative learning control. *International Journal of Control*, 76(11):1059–1069, 2003.

D. H. Owens, C. T. Freeman, and B. Chu. An inverse-model approach to multivariable norm optimal iterative learning control with auxiliary optimisation. *International Journal of Control*, 87(8):1646–1671, 2014.

D. H. Owens, C. T. Freeman, and B. Chu. Generalized norm optimal iterative learning control with intermediate point and sub-interval tracking. *International Journal of Automation and Computing*, 12(3):243–253, 2015.

D. H. Owens, C. T. Freeman, and T. V. Dinh. Norm-optimal iterative learning control with intermediate point weighting: Theory, algorithms, and experimental evaluation. *IEEE Transactions on Control Systems Technology*, 21(3):999–1007, 2013.

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

D. H. Owens, J. Hatonen, and S. Daley. Robust monotone gradient-based discrete-time iterative learning control. *International Journal of Robust and Nonlinear Control*, 19 (6):634–661, 2008.

D. H. Owens and R. P. Jones. Iterative solution of constrained differential/algebraic systems. *International Journal of Control*, 27(6):957–964, 1978.

J. Park, P. H. Chang, H. S. Park, and E. Lee. Design of learning input shaping technique for residual vibration suppression in an industrial robot. *IEEE/ASME Transactions on Mechatronics*, 11(1):55–65, 2006.

K.-H. Park, Z. Bien, and D.-H. Hwang. Design of an iterative learning controller for a class of linear dynamic systems with time delay. *IEE Proceedings of Control Theory and Applications*, 145(6):507–512, 1998.

W. Paszke, E. Rogers, K. Galkowski, and Z. Cai. Robust finite frequency range iterative learning control design and experimental verification. *Control Engineering Practice*, 21(10):1310–1320, 2013.

W. Paszkea, E. Rogers, and K. Galkowski. Experimentally verified generalized KYP lemma based iterative learning control design. *Control Engineering Practice*, 53:57–67, 2016.

J. D. Ratcliffe. *Iterative learning control implemented on a multi-axis system*. PhD thesis, University of Southampton, Sothampton, 2005.

R. Roesser. A discrete state-space model for linear image processing. *IEEE Transactions on Automatic Control*, 20:1–10, 1975.

S. K. Sahoo, S. K. Panda, and J.-X. Xu. Application of spatial iterative learning control for direct torque control of switched reluctance motor drive. In *IEEE Power Engineering Society General Meeting*, pages 1–7, Tampa, US, 2007.

D. Shen and Y. Wang. Survey on stochastic iterative learning control. *Journal of Process Control*, 24(12):64–77, 2014.

D. Shen, W. Zhang, and J.-X. Xu. Iterative learning control for discrete nonlinear systems with randomly iteration varying lengths. *Systems and Control Letters*, 96: 81–87, 2016.

T. D. Son, H.-S. Ahn, and K. L. Moore. Iterative learning control in optimal tracking problems with specified data points. *Automatica*, 49(5):1465–1472, 2013.

T. D. Son, D. H. Nguyen, and H.-S. Ahn. Iterative learning control for optimal multiple-point tracking. In *50th IEEE Conference on Decision and Control and European Control Conference*, pages 6025–6030, Orlando, US, 2011.

T. D. Son, G. Pipeleers, and J. Swevers. Robust monotonic convergent iterative learning control. *IEEE Transactions on Automatic Control*, 61(4):1063–1068, 2016.

M. Sun and D. Wang. Initial shift issues on discrete-time iterative learning control with system relative degree. *IEEE Transactions on Automatic Control*, 48(1):144–148, 2003.

H. Tao, W. Paszke, E. Rogers, H. Yang, and K. Galkowski. Iterative learning fault-tolerant control for differential time-delay batch processes in finite frequency domains. *Journal of Process Control*, 56:112–128, 2017.

A. Tayebi and M. B. Zaremba. Robust iterative learning control design is straightforward for uncertain LTI systems satisfying the robust performance condition. *IEEE Transactions on Automatic Control*, 48(1):101–106, 2003.

M. Uchiyama. Formation of high speed motion pattern of mechanical arm by trial. *Transactions of the Society of Instrumentation and Control Engineers*, 19(5):706–712, 1978.

J. van de Wijdeven and O. Bosgra. Non-causal finitetime iterative learning control. In *46th IEEE Conference on Decision and Control*, 2007.

J. van de Wijdeven and O. Bosgra. Residual vibration suppression using Hankel iterative learning control. *International Journal of Robust Nonlinear Control*, 18(10):1034–1051, 2008.

J. van de Wijdeven, T. Donkers, and O. Bosgra. Iterative learning control for uncertain systems: Robust monotonic convergence analysis. *Automatica*, 45:2383–2391, 2009.

D. Verscheure, B. Demeulenaere, J. Swevers, J. de Schutter, and M. Diehl. Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10):2318–2327, 2009.

M. Volckaert, M. Diehl, and J. Swevers. Generalization of norm optimal ILC for non-linear systems with constraints. *Mechanical Systems and Signal Processing*, pages 280–296, 2013.

J.-X. Xu, Y. Chen, T. Lee, and S. Yamamoto. Terminal iterative learning control with an application to RTPCVD thickness control. *Automatica*, 35(9):1535–1542, 1999.

J.-X. Xu and D. Huang. Initial state iterative learning for final state control in motion systems. *Automatica*, 44(12):3162–3169, 2008a.

J.-X. Xu and D. Huang. Spatial periodic adaptive control for rotary machine systems. *IEEE Transactions on Automatic Control*, 53(10):2402–2408, 2008b.

J.-X. Xu and X. Jin. State-constrained iterative learning control for a class of mimo systems. *IEEE Transactions on Automatic Control*, 58(5):1322–1327, 2013.

J.-X. Xu, J. Xu, and T. H. Lee. Iterative learning control for systems with input deadzone. *IEEE Transactions on Automatic Control*, 50(9):1455–1459, 2005.

Y.Wang and Z. Hou. Terminal iterative learning control based station stop control of a train. *International Journal of Control*, 84(7):1263–1277, 2011.

S.-H. Zhou, Y. Tan, D. Oetomo, C. T. Freeman, E. Burdet, and I. Mareels. Modeling of endpoint feedback learning implemented through point-to-point learning control. *IEEE Transactions on Control Systems Technology*, 25(5):1576–1585, 2017.