Iterative Learning Control and Adaptive Control for Systems with Unstable Discrete-Time Inverse

Bowen Wang

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2019

ABSTRACT

Iterative Learning Control and Adaptive Control for Systems with

Unstable Discrete-Time Inverse

Bowen Wang

Iterative Learning Control (ILC) considers systems which perform the given desired trajectory repetitively. The command for the upcoming iteration is updated after every iteration based on the previous recorded error, aiming to converge to zero error in the real-world. Iterative Learning Control can be considered as an inverse problem, solving for the needed input that produces the desired output.

However, digital control systems need to convert differential equations to digital form. For a majority of real world systems this introduces one or more zeros of the system z-transfer function outside the unit circle making the inverse system unstable. The resulting control input that produces zero error at the sample times following the desired trajectory is unstable, growing exponentially in magnitude each time step. The tracking error between time steps is also growing exponentially defeating the intended objective of zero tracking error.

One way to address the instability in the inverse of non-minimum phase systems is to use basis functions. Besides addressing the unstable inverse issue, using basis functions also has several other advantages. First, it significantly reduces the

computation burden in solving for the input command, as the number of basis functions chosen is usually much smaller than the number of time steps in one iteration. Second, it allows the designer to choose the frequency to cut off the learning process, which provides stability robustness to unmodelled high frequency dynamics eliminating the need to otherwise include a low-pass filter. In addition, choosing basis functions intelligently can lead to fast convergence of the learning process. All these benefits come at the expense of no longer asking for zero tracking error, but only aiming to correct the tracking error in the span of the chosen basis functions.

Two kinds of matched basis functions are presented in this dissertation, frequency-response based basis functions and singular vector basis functions, respectively. In addition, basis functions are developed to directly capture the system transients that result from initial conditions and hence are not associated with forcing functions. The newly developed transient basis functions are particularly helpful in reducing the level of tracking error and constraining the magnitude of input control when the desired trajectory does not have a smooth start-up, corresponding to a smooth transition from the system state before the initial time, and the system state immediately after time zero on the desired trajectory.

Another topic that has been investigated is the error accumulation in the unaddressed part of the output space, the part not covered by the span of the output basis functions, under different model conditions. It has been both proved mathematically and validated by numerical experiments that the error in the unaddressed space will remain constant when using an error-free model, and the unaddressed error will demonstrate a process of accumulation and finally converge to a constant level in the presence of model error. The same phenomenon is shown to apply when using unmatched basis functions. There will be unaddressed error accumulation even in the absence of model error, suggesting that matched basis functions should be used whenever possible.

Another way to address the often unstable nature of the inverse of non-minimum phase systems is to use the in-house developed stable inverse theory Longman JiLLL, which can also be incorporated into other control algorithms including One-Step Ahead Control and Indirect Adaptive Control in addition to Iterative Learning Control. Using this stable inverse theory, One-Step Ahead Control has been generalized to apply to systems whose discrete-time inverses are unstable. The generalized one-step ahead control can be viewed as a Model Predictive Control that achieves zero tracking error with a control input bounded by the actuator constraints. In situations where one feels not confident about the system model, adaptive control can be applied to update the model parameters while achieving zero tracking error.

Contents

List of Tables

List of Figures

Acknowledgement

I would like to express my deepest gratitude to my Research Advisor Prof. Richard Longman for being the best guide on my way exploring the fascinating world of control. Our weekly three-hour one-to-one individual meeting has always been enjoyable and rewarding, and my understanding of control, along with schedule planning and time management skills, has been further enhanced during our conference trips covering three continents. I feel particularly fortunate to meet Prof. Nicolas Chbat, who set an example and as a guide in the field of healthcare and biomedical engineering, from a principle research scientist to a company CEO, from a professor in the classroom to a mentor who leads me when I lost my way. Words cannot describe how grateful I am for my family who has been kindly supporting me along my four-year journey of Ph.D. study. Meeting Prof. Fred Stolfi, who guided me to explore the world of mechatronics and Internet of Things, and Prof. V.T. Rajan, the expert of translating physics into mathematical models for communication, during my Ph.D. study is definitely another thing that I am very thankful for. Cheers to my peer classmates Ben, Bumho, Joni, Blair, and Bill for the time that we spent together preparing the Doctoral Qualifying Exam and final exams. A “Thank you” from my heart goes to my friends and neighbors Joy, Yida, Hila, Amanda, Josef, and Joe for being part of my journey. I will always remember the time that I spent with Mudd 164Aers Jianzhong, Te, Ae, Bing, Tony, Alex, and Ayman. I truly appreciate the kind and timely assistance from the ME staff members Bob, Becca, Mel, Sandra, Jean, Lisa, Milko, Aixa, Emily, and Department Chair Prof. Jeffery Kysar, who enabled my Columbia campus life smooth and pleasant. Special thanks to Prof. Raimondo Betti and Prof. Homayoon Beigi for serving as the committee members of my three-hour Dissertation Defense.

To my most beloved family.

Chapter 1 Introduction

1.1 Background

Iterative Learning Control (ILC) aims to achieve zero tracking errors for systems performing the same finite-time tasks repeatedly. It adjusts the input control action to be applied to the real world for the upcoming iteration by learning from the error of the past iteration (Uchiyama, 1978, Arimoto et al., 1984, Bien et al., 1998, Longman, 2000, Bristow et al., 2006, Ahn et al., 2007). It has a sister field named Repetitive Control (RC), which executes a periodic command and learns from error in the previous period, or it similarly learns to eliminate a periodic disturbance of known period (Inoue et al., 1981, Omata et al., 1984, Middleton et al., 1989, Tomizuka et al., 1989, Longman, 2010). ILC has applications in fields such industrial manufacturing, disk storage reading and writing, copy machines, and space telescope, commanding sensors to perform repeated high accuracy scan maneuvers such as tracking pre-defined trajectories. RC is useful in spacecraft for active vibration isolation of fine pointing equipment. In theory, it can completely eliminate the influence of vibrations caused by slight imbalance from rotating CMGs, reaction wheels, cryogenic pumps, etc. Edwards et al. (1999) and Ahn et al. (2015) perform experiments on a Stewart platform vibration isolation mount, and also on a floating spacecraft testbed for laser communication between spacecraft. Both ILC and RC interact with the real world system instead of the estimated model, and improve trajectory tracking performances by learning from errors in the previous run and correspondingly adjust the control actions for the upcoming run. A key difference between ILC and RC, however, is that ILC resets to the same initial conditions before starting a new iteration, making ILC a finite-time trajectory tracking problem, while RC does not restart or return to the same initial conditions after each period and hence can aim at zero error as time goes to infinity. Both ILC and RC can be seen as inverse problems, solving for the required input control to produce the desired output.

1.2 The Unstable Inverse in Systems with Large Pole Excesses

When a continuous time system expressed by a differential equation is fed by a zero-order hold in digital control systems, it can be replaced by a difference equation without any approximation, producing the same output as the differential equation at each time step. However, the process of converting to a difference equation introduces zeros into the transfer function.

For systems whose pole excess, the number of poles minus the number of zeros in continuous time, is three or more, there will be at least one zero introduced into the z-transfer function located outside the unit circle when they are sampled at a reasonable sampling rate (Åström et al., 1980). The zero locations are a function of the time interval between samples T. Åström et al. (1984) gives an analytical derivation of the asymptotic locations of the system zeros. The number of zeros introduced depends on the pole excess. When the sampling period T tends to zero, the locations of the zeros introduced outside the unit circle for different pole excess are shown in Table 1. When the sample time interval T gets very long, so that the system transients have already decayed to near zero before the next time step arrives, then the zeros approach the origin. Usually, however, any sample rate that is able to keep the zeros inside the unit circle has an unreasonably low Nyquist frequency and is not useful because of the aliasing produced.

When solving the inverse problem, these introduced zeros become poles located outside the unit circle as the transfer function is inverted, resulting in an unstable learning process where the magnitude of the control action grows exponentially when targeting to produce a given trajectory. These kinds of exponentially growing commands are not practical and cannot be applied to the real world.

Table 1. Locations of Introduced Zeros Outside the Unit Circle When Sampling Period Tends to Zero

Pole Excess	Locations of Zeros Outside the Unit Circle
2	-1.0000
3	-3.7321
4	-9.8990, -1.0000
5	-23.2039, -2.3225
6	-51.2184, -4.5419, -1.0000
7	-109.3052, -8.1596, -1.8682
8	-228.5110, -13.9566, -3.1377, -1.0000
9	-471.4075, -23.1360, -4.9566, -1.6447
10	-963.8545, -37.5415, -7.5306, -2.5155, -1.0000
11	-1958.6431, -59.9893, -11.1409, -3.6740, -1.5123

Take a pole excess of 3 system as an example. Two zeros will be introduced during the discretization. Because one of the two zeros will be located near -3.732 with a reasonable sampling rate, the control action magnitude will be composed of a constant times $(3.732)^k$, where k is the time step. As shown in Fig. 1, the control action magnitude shown by the blue line is almost parallel to $(3.732)^k$ shown by the red line in logarithm scale, and it reached over 10^{50} after 100 iterations. A more extreme case is a pole excess of 11 system presented in Fig. 2 whose two largest zeros have magnitude of 1958.643 and 59.989 respectively when sampled at 1000 Hz. Similarly, the control action will be mostly influenced by these two zeros, and the control action magnitude will exceed 10^{300} after 100 iterations. Signatures of system unstable inverse can also be observed in the Toeplitz matrix *P*, which is a *p*-by-*p* sized matrix made up of system discrete-time Markov parameters that relates the *p*-step output history with the *p*-step input command.

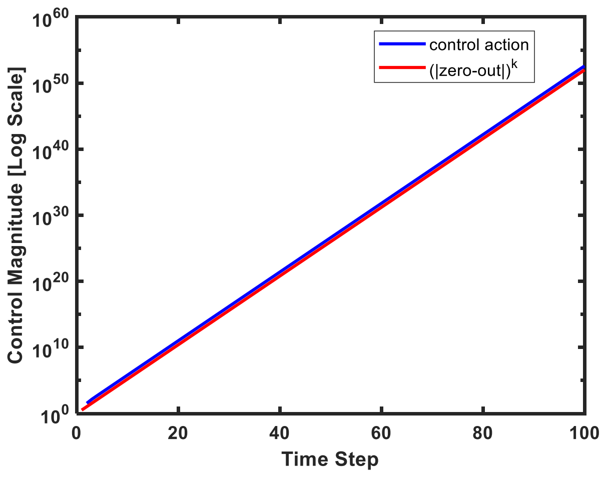


Figure 1. Log of control action magnitude when pole excess is 3.

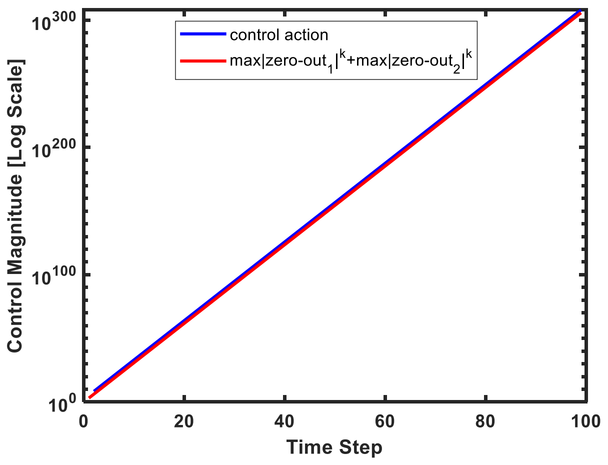


Figure 2. Log of control action magnitude when pole excess is 11.

1.3 Basis Functions

One effective way to address the unstable inverse issue in systems with large pole excesses is to use basis functions. There are several other benefits of using basis functions in ILC. First, it can significantly reduce the control dimension and the associated computation of ILC updates. This is particularly obvious when the desired trajectory is long or the sampling rate is fast, as the number of basis functions to be picked is usually much smaller than the number of time steps in one ILC iteration. In addition, due to the presence of unmodeled high frequency modes, ILC laws are generally expected to be unstable in the world. Often the instability develops very slowly because it is associated with high frequencies when the system magnitude response is small. Instability may not be observed for a large number of iterations, and sometimes the instability is eliminated by finite word length in the analog to digital converters. Yet, it is a safer choice to include a zero-phase low-pass filter to produce stability robustness to such model errors. When using basis functions, the need for such a filter is eliminated, which is another aspect that simplifies the control computation. Moreover, basis function ILC is likely to produce faster learning than typical ILC approaches. These benefits, however, come at a price. One no longer aims for zero tracking error but only aims at eliminating major errors of interest, usually at low frequencies. Besides, the basis functions are supposed to be picked intelligently to converge to small final error while having fast convergence at the same time.

One simple and direct approach to pick basis functions is to simply choose a set of orthonormal functions to span the dimension of the input and output spaces in ILC (Phan et al., 1999). Then one picks a subset of these to be the input and the output basis functions. Phan et al. (1996) and Frueh et al. (2000) use this approach on ILC, the former using a chosen subset of Legendre polynomials spanning the input and output spaces, and the latter using a chosen subset of the Fourier series sine and cosine functions spanning the space. No information of the system is needed to build these kinds of basis functions, and these are considered as unmatched basis functions where output and input basis functions are built independently. Other publications (Kempf et al., 1992, Longman et al., 2000, Chen et al., 2002, Nagashima et al., 2003, Nagashima et al., 2005, Shi et al., 2014, Wen et al., 1998) consider what is termed matched basis function, where each output basis function is the output produced by the associated input basis function. Matched basis functions make use of system knowledge and have a great advantage over the unmatched basis functions. When using matched basis functions, given an error component on output basis functions chosen as the steady state frequency response of the system at one frequency, then the associated input basis functions can cancel this error. It has been adjusted for the amplitude change and the phase change produced by sending the input sinusoidal basis function through the system so that it produces the associated output basis function. Therefore, matched basis functions are preferred over unmatched basis functions when possible.

1.4 In-House Developed Stable Inverse Theory

Besides using basis functions to overcome the instability issue when inverting a system with naturally unstable inverse, there have also been different kinds of stable inverse theories developed to obtain a feasible stable inverse. Devasia's research group developed a stable inverse theory that gives a stable inverse for a given time history, but this theory requires pre- and post-actuation periods in advance and after the zero tracking error trajectory (Devasia et al., 1996 and Zou et al., 1999). Researchers in Control System Research Group at Columbia University have developed a new stable inverse theory named Longman JiLLL, where "JiLLL" stands for the researchers involved in the development of this stable inverse theory (LeVoci et al., 2004, Li et al., 2010, Li et al., 2016, Longman et al., 2017, Ji et al., 2017). Longman JiLLL has four formats, i.e. FS, NS, FI, and NI respectively, where the letter "F" stands for "Factored Form", "N" stands for "Without Factoring", "S" for "Skipping Step", and "I" for "Initial Delete". The fundamental idea for "I" is to not ask for zero error for the number of initial time steps equal to the number of zeros outside the unit circle. The fundamental idea for "S" is to increase the sampling rate and ask for zero error only at the original time steps, i.e. the addressed time steps. The additional time steps introduced at a faster sampling rate are called unaddressed time steps and skipped when aiming for zero tracking error. For one zero outside, double the sample rate and ask for zero error only at the original time steps; triple the rate if there are two zeros outside, etc.

The in-house developed stable inverse theory not only addresses the unstable inverse issue of ILC in high-order system applications, but also offers opportunities for incorporation into various other control methods that have been of limited applications, one of which is one step ahead control. One step ahead control is a digital control approach aiming to produce zero error in following any chosen trajectory (Wang et al., 2018 and Goodwin et al., 1980). It is a promising control algorithm, as it is produced by a feedback control law and gives desired output at the next time step with zero tracking error. However, one step ahead control has been of limited applications mainly due to two failures. First, the control actions computed could be aggressive and go beyond the actuator limits. The other failure lies in the unstable system inverse of high-order systems, resulting in a control of exponential magnitude growth.

Another application of the in-house developed stable inverse theory is in indirect adaptive control. The one step ahead control algorithm is presented in Goodwin et al. (1980) and Goodwin et al. (1984) as a fundamental part of one of the basic approaches to indirect adaptive control. Adaptive control has applications when one may not feel confident about the estimated model, and wants to improve the model by updating the model parameters while tracking the desired trajectory. Similar to the situation~~s~~ with one step ahead control, indirect adaptive control has been of limited applications as it requires systems to have a stable transfer function inverse in discrete time, which is a small number of systems in practice. With the help of the in-house developed stable inverse theory, the indirect adaptive control could be generalized into batch stable inverse indirect adaptive control where the estimated model is updated based on the tracking error from the actual system output via the Projection Algorithm after getting a *p*-step history of input and output.

The third natural use of the new stable inverse is to linear model predictive control (LMPC), and a preliminary investigation of this is given in Zhu et al. (2017).

1.5 Dissertation Layout

This dissertation is made up of six chapters. Chapter 1 introduces the background of the research problem, i.e. instability of the inverse in systems with large pole excess, and proposed two methods, namely basis functions and Stable Inverse Theory Longman JiLLL, to address this issue. Both Chapter 2 and Chapter 3 focus on the applications of basis function ILC on systems with large pole excess. Chapter 2 presents the development of matched basis functions to capture transients and improve tracking performances in frequency response-based basis function ILC. Chapter 3 investigates the error accumulation in the unaddressed output space under different situations when using singular vector basis function ILC. Chapter 4 and Chapter 5 present applications of the in-house developed stable inverse theory in handling the instability in the inverse of high-order systems. Chapter 4 presents the generalization of one step ahead control with the help of the stable inverse theory and quadratic cost penalty, which can also be considered as a Model Predictive Control that, by updating the control batch by batch where the batch of time steps could be of relatively small size and close enough towards one, can achieve zero tracking error at all the original time steps. Chapter 5 presents the generalization of indirect adaptive control to systems whose discrete-time inverse is unstable. Finally, Chapter 6 concludes the dissertation by recapping the contribution.

Chapter 2 Development of Transient Basis Functions to Improve Basis Function Iterative Learning Control

2.1 Introduction

Iterative Learning Control (ILC) initially aims for zero tracking error in a control system performing the same finite-time task repeatedly (Uchiyama, 1978, Arimoto et al., 1984, Bien et al., 1998, Bristow et al., 2006, Ahn et al., 2007). It improves the performance by learning from the tracking error in the previous iteration and adjusting the input control action for the current iteration. It resets to the same initial conditions before starting a new iteration. Repetitive Control (RC) is a similar field, aiming to converge to zero tracking error, but, instead of performing a finite-time task repeatedly starting with the same initial conditions every run, it executes a periodic command and learns from the previous period, or learns to eliminate a period disturbance of a known period (Inoue et al., 1981, Omata et al., 1984, Middleton et al., 1989, Tomizuka et al., 1989, Panomruttanarug et al., 2004, Chen et al., 2006, Bao et al., 2008, Longman et al., 2008, Longman, 2010). ILC applications in spacecraft include commanding sensors to perform repeated high accuracy scan maneuvers. It can also be used in manufacturing industries such as mechanical parts assembly, and any other fields where high precision is required in performing a given finite-time repeated task. RC is useful for active vibration isolation of fine pointing equipment in spacecraft, as theoretically it can completely remove the influence of vibrations caused by slight imbalance from rotating CMGs, reaction wheels, cryogenic pumps, etc. Edwards et al. (1999) and Ahn et al. (2015) perform experiments on a Stewart platform vibration isolation mount, and also on a floating spacecraft testbed for laser communication between spacecrafts.

ILC laws update the input command between two iterations of the size of the number of time steps in one run, which can be a large number and results in substantial computation. By using basis functions, high tracking accuracy can still be achieved with much less computation. However, this is obtained at the expense of no longer asking for overall zero tracking error, but only reducing the larger errors normally associated with lower frequencies. Researchers from various groups have studied the use of basis functions, usually sinusoidal functions spanning up to a chosen frequency (Kempf et al., 1992, Wen et al., 1998, Longman et al., 2000, Chen et al., 2002, Nagashima et al., 2003, Nagashima et al., 2005, Shi et al., 2014). These references mostly consider matched basis function, i.e. each output basis function is the output produced by the associated input basis function. An advantage of using basis functions that are matched is that, given an error component on output basis functions chosen as the steady state frequency response of the system at one frequency, then the associated input basis functions can cancel this error. It has been adjusted for the amplitude change and phase change produced by sending the input sinusoidal basis function through the system so that it produces the associated output basis function. Phan et al. (1996) and Frueh et al. (2000) apply basis functions in ILC, where Phan et al. (1996) considering a chosen set of Legendre polynomials while Frueh et al. (2000) using basis functions based on frequency. However, neither use matched basis functions; instead, general unmatched basis functions are considered, i.e. simply picking functions to span the chosen part of the input and output space independently.

Unlike ILC, RC does not restart and it asks for convergence to zero tracking error as time goes to infinity, and using steady state frequency response-based basis functions for RC is appropriate as usually the influence from the transients will decay and become negligible as time goes on. On the other hand, ILC resets to the same initial conditions before starting a new iteration and tracking the same finite-time trajectory, producing transients every iteration at the start of tracking. ILC asks for zero tracking error at every time step including the transient period as well as the rest part of the trajectory where initial condition influence has faded and steady state frequency response dominates. With transients expected at the beginning of the finite-time trajectory, steady state frequency response-based sinusoidal functions are appropriate in the latter part of the trajectory, but can have difficulty capturing the effects of decaying transients.

In this chapter, a new kind of basis functions for capturing system transients is developed. The prime challenge in developing such transient basis functions lies in the fact that transients are solutions of the homogeneous equation without an input, and they are produced by initial conditions with no forcing function input. This is fundamentally different from the usual matched basis functions, where an input basis function that is a forcing function to the system equation is matched to the resulting output particular solution. After successfully developing basis functions to capture the system transients, the remaining part of the trajectory considers steady state frequency response-based basis functions. An analysis is given to show when these newly developed basis functions are of particular help.

2.2 General Iterative Learning Control Formulation

The general Iterative Learning Control formulation as developed by Phan et al. (1988) is presented here. Consider a general time invariant discrete time system. It can be expressed in state space form as shown in Eq. (2.1):

$$x(k+1) = Ax(k) + Bu(k) \quad ; \quad y(k) = Cx(k) + v(k) \tag{2.1}$$

where $u(k), y(k), v(k)$ are the input control action, system output, and any repeating output disturbance respectively at time step k. The disturbance can represent the effect on the output of a repeating disturbance occurring anywhere in the feedback system. To see the relationship between a *p*-step-long input time history and the resultant output history, Eq. (2.1) can be written as

$$\underline{y} = P\underline{u} + \bar{A}x(0) + \underline{v} \tag{2.2}$$

where vector $\underline{y} = [y(1) \quad y(2) \quad \dots \quad y(p)]^T$, $\underline{u} = [u(0) \quad u(1) \quad \dots \quad u(p-1)]^T$, and $\underline{v} = [v(1) \quad v(2) \quad \dots \quad v(p)]^T$. The system Toeplitz matrix and initial condition matrix are respectively

$$P = \begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \dots & CB \end{bmatrix}, \quad \bar{A} = \begin{bmatrix} CA \\ \vdots \\ CA^p \end{bmatrix} \tag{2.3}$$

In this work a one time-step delay from input to the output is considered as shown in Eq. (2.2), and in such a case the diagonal element *CB* in matrix *P* is non-zero. Simple adjustments can be made to treat other delays. A subscript *j* is used to indicate the iteration number in ILC, and the initial conditions $x(0)$ are reset to the same starting value for each iteration. Define a backward difference operator $\delta_j z(k) = z_j(k) - z_{j-1}(k)$, and apply it to Eq. (2.2) to produce

$$\delta_j \underline{y} = P\delta_j \underline{u} \tag{2.4}$$

which is equivalent to $\underline{y}_j = \underline{y}_{j-1} + P\delta_j \underline{u}$. The other terms in Eq. (2.2) disappeared in Eq. (2.4) because the repeating disturbance and initial conditions are the same for every iteration. In ILC, the input command to be applied for the $(j+1)^{th}$ iteration is adjusted by learning from the tracking errors in the j^{th} learning iteration. A general linear ILC law has the form

$$\delta_{j+1}\underline{u} = L\underline{e}_j \tag{2.5}$$

where the tracking error $\underline{e}_j = \underline{y}_D - \underline{y}_j$, and $\underline{y}_D$ is the desired trajectory. L is the learning matrix, and one of its commonly used forms is $L = diag(\phi_1, \phi_2, \cdots, \phi_p)$ with ϕ_i as the learning gain constants. Other forms of ILC laws and their corresponding stability robustness are further presented by Bao et al. (2010). Combining Eq. (2.5) with Eq. (2.4) gives Eq. (2.6) showing how error propagates from iteration to iteration

$$\delta_{j+1}\underline{e} = -P\delta_{j+1}\underline{u} = -PL\underline{e}_j \tag{2.6}$$

which can be further simplified into $\underline{e}_j = (I - PL)\underline{e}_{j-1}$. To guarantee a stable learning process where tracking errors gradually decay to zero as iteration goes on, it is necessary to make all the eigenvalues of matrix $I - PL$ have magnitudes less than unity.

2.3 Basis Function Iterative Learning Control

2.3.1 Benefits of Using Basis Functions

There are several obvious benefits of using basis functions. First, it is an effective way to address the unstable inverse issue in systems with large pole excesses. ILC is an inverse problem, solving for the input that produces the given desired output. When a continuous time system represented by a differential equation is fed by a zero-order hold in digital control systems, there is an equivalent difference equation model producing identical outputs at the sample times. And for systems whose pole excess, the number of poles minus the number of zeros in continuous time, is three or more, there will be zeros introduced into the *z*-transfer function that are located outside the unit circle for reasonable sample rates. When inverting this transfer function, solving for the control action to produce a given output, these zeros become poles located outside the unit circle, making the inverse problem an unstable process. While introducing a zero-phase filter of the input signal every iteration can produce stability robustness effectively, this can result in substantial computation. Basis functions are a convenient alternative to avoid unstable control input. Besides, using basis functions can significantly reduce the computation burden associated with ILC updates. ILC laws are generally expected to be unstable in the world due to the unmodeled high frequency modes. Although the instability often develops very slowly because it is associated with high frequencies at which the system magnitude response is small, and the instability may not be observed for a large number of iterations or sometimes simply gets eliminated by the finite word length in analog to digital converters, it is still a safe practice to apply a zero phase low pass filter to produce stability robustness against such model errors. Using basis functions eliminates the need to have a filter as at what frequency to be cut off can be chosen, greatly simplifying the control computation. Using basis functions can also reduce the associated computation burden in the sense that the number of basis functions chosen is usually much smaller than the number of time steps in one iteration, and this is a big advantage when the sample rate is fast, the trajectory to be tracked is long, or any other situations when there are many time steps in one learning iteration. The new basis functions aiming to capture the transients developed in this chapter will further reduce the control dimension, as these newly developed transient-based basis functions are designed to directly capture error components on the transients of the system instead of trying to represent them by superposing unrelated basis functions. These transient-based basis functions are particularly useful in obtaining good tracking performances for ILC problems where the desired trajectory is short compared to the settling time of the system, such as seeking accurate performance of particularly high-speed maneuvers. In addition, using basis functions in ILC is expected to reach learning convergence faster than typical ILC approaches when the basis functions are chosen intelligently.

Yet these benefits come at a price, as basis function ILC no longer aims for zero tracking error but only aims to handle large errors usually associated with lower frequencies. What's more, smart choices for basis functions are expected in order to converge to small final error while reaching convergence quickly at the same time.

2.3.2 Basis Functions Addressing the ILC Finite Time Nature

One simple and direct approach to pick basis functions is, as presented in Phan et al. (1996) and Frueh et al. (2000), to simply pick a set of orthonormal functions to span the dimension of the input and output space in ILC, then pick a subset of these orthonormal functions to form the input and the output basis functions. Phan et al. (1996) chooses a subset of Legendre polynomials spanning the p-dimensional input and output spaces, and Frueh et al. (2000) uses a chosen subset of the Fourier series sine and cosine functions spanning the space.

RC does not reset to the same initial conditions before starting a new period, so it is appropriate to ask for zero error as time goes to infinity when tracking a periodic trajectory. Fourier series elements are a good basis function choice for RC, as the sines and cosines for the fundamental and harmonics and DC can represent the desired trajectory perfectly. Using matched basis functions means each input basis function will produce a corresponding output basis function based on steady state frequency response of the model, and these matched input and output basis functions correct for the phase and amplitude changes of sinusoids going through the system. Since usually transients would decay significantly and already become small by the second period of the desired trajectory, using sines and cosines basis functions in RC can be expected to reach convergence fast.

The finite-time nature of ILC problem means that the output usually contains transients. In fact, the system response can be considered as made up of two phases, the transients and steady state frequency response respectively. Each of the transients is associated with a time constant for decay, and the settling time is three or four times of the largest time constant in the solution of the system homogeneous equation, which is defined as the transient time interval. After the settling time, the influence of initial conditions becomes very small and can be neglected, and the rest of the trajectory can be considered as being dominated by steady state frequency response. The concept of matched basis function described about in RC problems can be applied directly to the second part of the trajectory where frequency response dominates, but it will not work well to capture the transients in the output. One way to capture the transients is to apply enough sines and cosines aiming to capture every time step of the *p*-time-step trajectory to a perfect extend, but in this case many sinusoids will be needed to capture non-periodic signals using periodic sinusoidal functions, which will not only result in a huge computation load but also bring no obvious benefit. Therefore, it is desirable to develop basis functions that can directly capture transients in the system response, as the combination of transient basis functions and sinusoid basis functions is expected to produce fast convergence in ILC.

A fundamental issue that must be addressed in designing matched transient basis functions is that the matched output basis function is the response produced by the associated input basis function, but transients are solutions of the system homogeneous equation and hence there are no forcing functions to produce them. The transients are produced by initial conditions only. Think about a third-order scalar difference equation modeling the input to output of a feedback control system. A valid set of initial conditions is a sequence of three outputs, or more generally, a sequence of three outputs together with corresponding three inputs during this interval. These inputs adjusting the three time steps of initial conditions, during or before the start of the tracking problem, are developed to produce three linearly independent transients. The newly developed transient basis functions correspond to adjusting the input at these time steps producing independent initial conditions, and the matched output basis functions are the transients in the output produced by these modified initial conditions. Because the developed input and output basis functions are matched, they are able to eliminate the corresponding part of the error in one ILC update when using an error-free model. With the transients removed from the output by the developed transient basis functions, the matched sinusoidal basis functions are expected to be better able to capture the system output response, as they now represent the actual input-output relationship more closely.

2.4 Matched Basis Functions Based on Steady State Frequency Response

For the part of the trajectory after the settling time where influence from transients have died out, it is appropriate to capture it using basis functions generated from steady state frequency response. In this section, matched sinusoidal basis functions associated with steady state frequency response are generated.

Taking a third-order system whose transfer function in the discrete time domain is expressed by Eq. (2.7) as an example. To obtain the steady state frequency response of the *z*-transfer function

$$G(z) = \frac{b_2 z^2 + b_1 z + b_0}{z^3 + a_2 z^2 + a_1 z + a_0} \tag{2.7}$$

substitute $z = \exp(i\omega_n T) = \exp(i\theta_{pn})$, then the magnitude of the transfer function is the amplitude of the response, and the phase angle of the transfer function is the phase change going through the system. In ILC, there are p time steps in the output of one iteration. Divide 2π by p to produce the angle increment $\Delta\theta$ between the frequencies observable in p time steps of data. For an even number p, the frequencies that can be seen are $n\Delta\theta$ where $n = 0,1,\dots,p/2$; and when p is odd, the frequencies are $n\Delta\theta$ for $n = 0,1, \dots, (p-1)/2$. All possible output frequency components are linear combinations of both sine and cosine at these frequencies, i.e. $\exp(i\theta_{pn}) = \cos(\theta_{pn}) + i\sin(\theta_{pn})$. Inputs of $\sin(n\mathit{\Delta}\theta k)$ and $\cos(n\mathit{\Delta}\theta k)$ produce steady state outputs $r(n\mathit{\Delta}\theta/T)(\sin(n\mathit{\Delta}\theta k + \tau(n\mathit{\Delta}\theta/T))$ and $r(n\mathit{\Delta}\theta/T)(\cos(n\mathit{\Delta}\theta k + \tau(n\mathit{\Delta}\theta/T))$, where T is the sample time interval, k is the time step, $r(n\mathit{\Delta}\theta/T)$ is the magnitude response $|G(\exp(n\mathit{\Delta}\theta/T))|$, and $\tau(n\mathit{\Delta}\theta/T)$ is the phase change given by the angle of $G(\exp(n\mathit{\Delta}\theta/T))$ made with the positive real axis.

The outputs above could be used as output basis functions, projecting the output error onto these functions. Then the associated input sine or cosine becomes the matched basis functions. For simplicity and convenience, $\sin(n\mathit{\Delta}\theta k)$ and $\cos(n\mathit{\Delta}\theta k)$ are made the output basis functions, and the corresponding input basis functions are given as

$$[1/r(n\mathit{\Delta}\theta/T)](\sin(n\mathit{\Delta}\theta k - \tau(n\mathit{\Delta}\theta/T))$$

$$[1/r(n\mathit{\Delta}\theta/T)](\cos(n\mathit{\Delta}\theta k - \tau(n\mathit{\Delta}\theta/T)) \tag{2.8}$$

To form the output basis function space, pick N frequencies of interest, and form the output basis function matrix $H = [H_1\ H_2\ \dots\ H_N]$ whose columns are the pure output sinusoids for all p time steps for the chosen frequency, starting with step $k = 1$ and going to time step p. The corresponding input basis function matrix $B = [B_1\ B_2\ \dots\ B_N]$ has the associated sines and cosines at these frequencies from Eq. (2.8) for time steps $k = 0, \dots, p-1$, where the assumed one time step delay through the system has been accounted for.

2.5 Developing Matched Basis Functions to Capture Transients

For a system whose pole excess is 3 in continuous time, it can be expressed by a differential equation as shown in Eq. (2.9).

$$\frac{d^3y}{dt^3} + \alpha_2 \frac{d^2y}{dt^2} + \alpha_1 \frac{dy}{dt} + \alpha_0 y = u \tag{2.9}$$

Feed this system with a zero-order hold to convert it to an equivalent discrete time model that has the same solution at the sample times, and this conversion process introduces zeros into the *z*-transfer function. For this pole excess of 3 system, the difference equation model will have one introduced zero that is located outside the unit circle for any reasonable sample rates. And when inverting the problem as ILC does, solving for the required command to produce the given output, the zero becomes a pole located outside the unit circle, resulting in an unstable solution. Using basis functions can eliminate this instability problem provided what is applied to the system is not an input on the input basis function associated with the instability, i.e. the input singular vector of the *P* matrix in Eq. (2.3) associated with the singular value that is produced by the zero outside the unit circle. Eq. (2.10) to Eq. (2.13) show the computation procedure of the third-order difference equation with the same solution as the differential equation at the sample times.

$$s^3Y(s) + \alpha_2 s^2 Y(s) + \alpha_1 sY(s) + \alpha_0 Y(s) = U(s) \tag{2.10}$$

$$G(s) = 1/(s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0) \tag{2.11}$$

$$G(z) = (1 - z^{-1})\mathrm{Z}\left\{\frac{G(s)}{s}\right\} = (b_2 z^2 + b_1 z + b_0)/(z^3 + a_2 z^2 + a_1 z + a_0) \tag{2.12}$$

$$y(k+3) + a_2 y(k+2) + a_1 y(k+1) + a_0 y(k) = b_2 u(k+2) + b_1 u(k+1) + b_0 u(k) \tag{2.13}$$

2.5.1 The Freedom in The Initial Conditions for Higher Order Scalar Difference Equations

For a third-order scalar difference equation, it is normal to expect output at three successive time steps to form the initial conditions. More precisely, the three time steps serving as the initial conditions to start Eq. (2.13) as a recursive equation are

$$y(0) = -a_2 y(-1) - a_1 y(-2) - a_0 y(-3) + b_2 u(-1) + b_1 u(-2) + b_0 u(-3)$$

$$y(-1) = -a_2 y(-2) - a_1 y(-3) - a_0 y(-4) + b_2 u(-2) + b_1 u(-3) + b_0 u(-4)$$

$$y(-2) = -a_2 y(-3) - a_1 y(-4) - a_0 y(-5) + b_2 u(-3) + b_1 u(-4) + b_0 u(-5)$$

Consider that the system is totally at rest before the initial conditions specified here, then what need to be known to start the recursion are outputs $y(0), y(-1), y(-2)$ and inputs $u(-1), u(-2), u(-3)$. The output and input at previous time steps are all zero, and substituting these zero values into the above procedure produces the relationship between $y(0), y(-1), y(-2)$ and $u(-1), u(-2), u(-3)$ shown in Eq. (2.14)

$$\begin{bmatrix} 1 & a_2 & a_1 \\ 0 & 1 & a_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y(0) \\ y(-1) \\ y(-2) \end{bmatrix} = \begin{bmatrix} b_2 & b_1 & b_0 \\ 0 & b_2 & b_1 \\ 0 & 0 & b_2 \end{bmatrix} \begin{bmatrix} u(-1) \\ u(-2) \\ u(-3) \end{bmatrix} \tag{2.14}$$

Eq. (2.14) proves that, in fact, only three initial conditions are needed to start the difference equation recursion instead of six initial condition values. Based on the relationship in Eq. (2.14), $u(-1), u(-2), u(-3)$ can produce the desired output $y(0), y(-1), y(-2)$ starting from a rest state.

2.5.2 State Variable Difference Equation from Scalar Difference Equation

Convert the scalar difference Eq. (2.13) to state variable form by starting from the modified Eq. (2.13)

$$\hat{y}(k+1) + a_2\hat{y}(k) + a_1\hat{y}(k-1) + a_0\hat{y}(k-2) = u(k) \tag{2.15}$$

and defining the state vector as

$$x_d(k) = \begin{bmatrix} \hat{y}(k) \\ \hat{y}(k-1) \\ \hat{y}(k-2) \end{bmatrix} \tag{2.16}$$

Multiply Eq. (2.15) by b_2, then shift back one time step and multiply it by b_1, shift back another time step and multiply by b_0, and sum the results to show that

$$y(k) = [b_2 \quad b_1 \quad b_0]x_d(k) = C_d x_d(k) \tag{2.17}$$

The state space difference equation started from Eq. (2.15) becomes

$$\begin{bmatrix} \hat{y}(k+1) \\ \hat{y}(k) \\ \hat{y}(k-1) \end{bmatrix} = \begin{bmatrix} -a_2 & -a_1 & -a_0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \hat{y}(k) \\ \hat{y}(k-1) \\ \hat{y}(k-2) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(k) = A_d x_d(k) + B_d u(k) \tag{2.18}$$

From Eq. (2.18), the relationship of input-output history for any iteration corresponding to Eq. (2.2) with the periodic disturbance neglected is presented in Eq. (2.19), which replaces the $\bar{A}$ of Eq. (2.3) by an $\hat{A}$ employing A_d and C_d.

$$\underline{y} = \begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \\ y(p) \end{bmatrix} = \begin{bmatrix} C_d A_d \\ C_d A_d^2 \\ C_d A_d^3 \\ \vdots \\ C_d A_d^p \end{bmatrix} \begin{bmatrix} \hat{y}(0) \\ \hat{y}(-1) \\ \hat{y}(-2) \end{bmatrix} + \begin{bmatrix} C_d B_d & 0 & \cdots & 0 \\ C_d A_d B_d & C_d B_d & \ddots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_d A_d^{p-1} B_d & C_d A_d^{p-2} B_d & \cdots & C_d B_d \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(p-1) \end{bmatrix} \tag{2.19}$$

2.5.3 Matched Basis Functions to Capture Transient Response Components

Apply Eq. (2.15) to Eq. (2.19) to produce Eq. (2.20) where the inputs of three initial time steps are used as the initial conditions.

$$\underline{y} = \begin{bmatrix} C_d A_d \\ C_d A_d^2 \\ C_d A_d^3 \\ \vdots \\ C_d A_d^p \end{bmatrix} \begin{bmatrix} 1 & -a_2 & a_2^2 - a_1 \\ 0 & 1 & -a_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u(-1) \\ u(-2) \\ u(-3) \end{bmatrix} + P_d \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(p-1) \end{bmatrix} \tag{2.20}$$

For a third-order difference equation, there will be three linearly independent transient solution histories. One way to specify three such solutions is to pick the 3 outputs produced by picking one entry to be unity and the others zero, among the components $u(-1), u(-2), u(-3)$. The associated transients produced are given by

$$\underline{y} = \begin{bmatrix} C_d A_d \\ C_d A_d^2 \\ C_d A_d^3 \\ \vdots \\ C_d A_d^p \end{bmatrix} \begin{bmatrix} 1 & -a_2 & a_2^2 - a_1 \\ 0 & 1 & -a_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u(-1) \\ u(-2) \\ u(-3) \end{bmatrix} = [\underline{S}_1 \quad \underline{S}_2 \quad \underline{S}_3] \begin{bmatrix} u(-1) \\ u(-2) \\ u(-3) \end{bmatrix} \tag{2.21}$$

There are new entries to introduce into the input basis function matrix, appending the three new initial inputs $u(-1), u(-2), u(-3)$ to the existing p time step input history that starts from time step 0 and goes to $p-1$. Pick the first input basis functions related with the transients 1, 0, 0 followed by zeros to $p-1$, second transient input basis function 0, 1, 0 again followed by zero entries to $p-1$, and third transient input basis function 0, 0, 1 similarly. The corresponding output basis functions aiming to capture the transients are then $\underline{S}_1, \underline{S}_2, \underline{S}_3$ in Eq. (2.21), which are the new matched output basis functions.

Note that $\underline{S}_1, \underline{S}_2, \underline{S}_3$ are not orthonormal to each other, and the benefits of having orthonormal basis functions and how to create orthonormal basis functions are illustrated in the following section.

2.6 The Benefits of Orthonormal Matched Basis Functions

2.6.1 Error Reduction When Using Non-Orthogonal Output Basis Functions

In this section, the performance of error reduction using non-orthogonal output basis functions is studied. The tracking error is first projected onto normalized but non-orthogonal output basis functions, and then associated input basis functions are applied aiming to cancel the error. Consider three normalized non-orthogonal basis function vectors $v_{1,}v_{2,}v_3$ spanning the output space. Suppose that $v_{1,}v_2$ are the basis functions spanning the output space being addressed by the learning process, and that v_3 is a vector orthogonal to this space corresponding to errors that are not addressed by the learning process. When the component of the error is projected onto v_1 , the matched basis function will eliminate this component of the error, $v_1(v_1^Te)$, and similarly the error vector $v_2(v_2^Te)$ is also eliminated. The matched basis functions do not influence the component of the error on v_3 , $v_3(v_3^Te)$. For v_2, a new vector $\overline{v}_2$ that is orthogonal to v_1 must be generated to have the error expressed on, and such a vector can be written as $\overline{v}_2 = v_2 - v_1(v_1^Tv_2)$. The original error vector now becomes $e = v_1(v_1^Te) + \overline{v}_2\left(\overline{v}_2^Te\right) + v_3(v_3^Te)$. After applying the matched basis functions that take out the error components described above, the error remaining is

$$e_{rem} = -v_1(v_1^Tv_2)\left(\overline{v}_2^Te\right) + v_3(v_3^Te) \tag{2.22}$$

Obviously, the error has been reduced by applying matched basis functions, but there is still error remaining because the output basis functions are not orthogonal. It leads to the conclusion that, with repeated applications of the matched basis function procedure by ILC updates, one will have a contraction mapping, and there will be convergence of the error to zero in the addressed part of the output space. The convergence is asymptotic instead of immediate.

2.6.2 Error Reduction When Using Matched Orthonormal Output Basis Functions

This section studies the error reduction when matched orthonormal output basis functions are applied. Consider an orthonormal set of output basis functions $\bar{S}_1, \bar{S}_2, \bar{S}_3$. A general output error vector is then projected onto these orthonormal basis functions, and, as in the previous section, matched input basis functions are used to cancel each projected component of the error. Now take the inner product of this with any arbitrary vector in the input space whose components on the orthonormal basis are given by $\gamma_1, \gamma_2, \gamma_3$ respectively,

$$[\gamma_1\bar{S}_1 + \gamma_2\bar{S}_2 + \gamma_3\bar{S}_3]^T \left[e - \bar{S}_1\left(\bar{S}_1^T e\right) - \bar{S}_2\left(\bar{S}_2^T e\right) - \bar{S}_3\left(\bar{S}_3^T e\right)\right] = 0 \qquad (2.23)$$

and the result of Eq. (2.23) is zero, meaning that all error components in the output basis function space have been cancelled.

2.6.3 Creating Orthonormal Output Basis Functions via Gram Schmidt Procedure

Starting from the output basis functions for transients $\underline{S}_1, \underline{S}_2, \underline{S}_3$, an orthonormal set of output basis functions $\bar{S}_1, \bar{S}_2, \bar{S}_3$ is generated using the Gram Schmidt Procedure illustrated in Eqs. (2.24).

$$\bar{S}_1 = \frac{\underline{S}_1}{\|\underline{S}_1\|}$$

$$\bar{S}_2 = \frac{\underline{S}_2 - (\bar{S}_1^T \underline{S}_2)\bar{S}_1}{\left\|\underline{S}_2 - (\bar{S}_1^T \underline{S}_2)\bar{S}_1\right\|}$$

$$\bar{S}_3 = \frac{\underline{S}_3 - (\bar{S}_1^T \underline{S}_3)\bar{S}_1 - (\bar{S}_2^T \underline{S}_3)\bar{S}_2}{\left\|\underline{S}_3 - (\bar{S}_1^T \underline{S}_3)\bar{S}_1 - (\bar{S}_2^T \underline{S}_3)\bar{S}_2\right\|}$$

(2.24)

These are output basis functions to be included into the output basis function matrix.

2.6.4 The Associated Matched Basis Functions

Denote the matched basis functions associated with $\underline{S}_1, \underline{S}_2, \underline{S}_3$ as $\underline{B}_1, \underline{B}_2, \underline{B}_3$, each of which is a vector of length p. The $\underline{B}_1$ is an all zero column vector except that the first entry is unity, $\underline{B}_2$ is all zero except the second entry is unity, and $\underline{B}_3$ has the third entry unity. Define coefficients

$$\beta_1 = \left\|\underline{S}_1\right\|^{-1}$$

$$\beta_2 = \left\|\underline{S}_2 - (\bar{S}_1^T \underline{S}_2)\bar{S}_1\right\|^{-1}$$

$$\beta_3 = \left\|\underline{S}_3 - (\bar{S}_1^T \underline{S}_3)\bar{S}_1 - (\bar{S}_2^T \underline{S}_3)\bar{S}_2\right\|^{-1} \tag{2.25}$$

$$\gamma_{ij} = \bar{S}_i^T \underline{S}_{\mathrm{j}}$$

Denote the matched input basis functions for Eqs. (2.24) by $\bar{B}_1, \bar{B}_2, \bar{B}_3$. Inserting the matched basis functions for the non-orthogonal $\underline{S}_1, \underline{S}_2, \underline{S}_3$ into Eqs. (2.24) and then collecting terms produces the new matched input functions for the orthonormal $\bar{S}_1, \bar{S}_2, \bar{S}_3$. These new matched input basis functions are given by

$$\bar{B}_1 = \beta_1 \underline{B}_1$$

$$\bar{B}_2 = \beta_2\left[\underline{B}_2 - \gamma_{12}\beta_1\underline{B}_1\right] \tag{2.26}$$

$$\bar{B}_3 = \beta_3\left[\underline{B}_3 - \gamma_{23}\beta_2\underline{B}_2 + (\gamma_{23}\gamma_{12}\beta_2\beta_1 - \gamma_{13}\beta_1)\underline{B}_1\right]$$

2.7 Relating Higher Order Scalar Difference Equation Initial Conditions to Trajectory Initial Conditions

2.7.1 Converting the State Space Differential Equation to Diagonal Form

Convert the differential equation Eq. (2.9) to state space form using $y(t), \dot{y}(t), \ddot{y}(t)$ as the state variables

$$\dot{x}_c(t) = A_c x_c(t) + B_c u(t) \qquad y(t) = C_c x_c(t) \tag{2.27}$$

where

$$A_c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\alpha_0 & -\alpha_1 & -\alpha_2 \end{bmatrix} \qquad B_c = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad C_c = [1 \quad 0 \quad 0]$$

Let $\lambda_1, \lambda_2, \lambda_3$ be the roots of the characteristic polynomial of Eq. (2.9) or of matrix A_c, which can be written as $\lambda^3 = -\alpha_0 - \alpha_1\lambda - \alpha_2\lambda^2$. Assuming the roots are distinct, the eigenvectors of matrix A_c are the columns of the Vandermonde matrix, shown by using the characteristic polynomial and examining column by column

$$A_c \begin{bmatrix} 1 & 1 & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 \end{bmatrix} \Lambda_c = M_c \Lambda_c \qquad \Lambda_c = diag(\lambda_1 \quad \lambda_2 \quad \lambda_3) \tag{2.28}$$

From $x_c(t) = \exp(A_c t)\, x_c(0)$ and $\exp(A_c t) = \exp(M_c \Lambda_c M_c^{-1} t) = M_c \exp(\Lambda_c t) M_c^{-1}$, which is obtained by using the Taylor series definition of the exponential of a matrix, the solution of the homogeneous differential equation for any time *t* can then be written as

$$[M_c^{-1} x_c(t)] = \exp\,(\Lambda_c t)\; [M_c^{-1} x_c(0)]$$

and at the sample times with step number *k* and sample time interval *T*

$$[M_c^{-1}x_c(kT)] = \exp(\Lambda_c kT)\,[M_c^{-1}x_c(0)] = diag[(e^{\lambda_1 T})^k \quad (e^{\lambda_2 T})^k \quad (e^{\lambda_3 T})^k][M_c^{-1}x_c(0)] \tag{2.29}$$

2.7.2 Converting the State Space Difference Equation to Diagonal Form

Now consider the state space difference equations Eq. (2.18), whose characteristic polynomial is $z^3 = -a_0 - a_1 z - a_2 z^2$, and perform steps analogous to the steps in the previous section. Matrix A_d is diagonalized, $M_d^{-1}A_d M_d = \Lambda_d$, by the modified Vandermonde matrix with columns that are the eigenvectors as demonstrated by

$$A_d \begin{bmatrix} z_1^2 & z_2^2 & z_3^2 \\ z_1 & z_2 & z_3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} z_1^2 & z_2^2 & z_3^2 \\ z_1 & z_2 & z_3 \\ 1 & 1 & 1 \end{bmatrix} \Lambda_d = M_d \Lambda_d \qquad \Lambda_d = diag(z_1 \quad z_2 \quad z_3) \tag{2.30}$$

From $x_d(kT) = A_d^k\, x_d(0)$ and $A_d^k = (M_d \Lambda_d M_d^{-1})^k = M_d(\Lambda_d)^k M_d^{-1}$, the equation analogous to Eq. (2.29) can be obtained

$$[M_d^{-1}x_d(kT)] = (\Lambda_d)^k [M_d^{-1}x_d(0)] = diag(z_1^k \quad z_2^k \quad z_3^k)[M_d^{-1}x_d(0)] \tag{2.31}$$

2.7.3 Equivalent Initial Control Inputs Matching Trajectory Initial Conditions

The differential equation Eq. (2.9) and the difference equation Eq. (2.13) by construction have the same outputs at the sample times $y(kT)$. Here we are only concerned about the solutions to the homogeneous equations, but it is true of the particular solutions provided the input to Eq. (2.9) comes through a zero-order hold. Therefore, the transients for each system must match, i.e. z_1^k , z_2^k, z_3^k and $(e^{\lambda_1 T})^k, (e^{\lambda_2 T})^k, (e^{\lambda_3 T})^k$ must match, after one adjusts the orders of the discrete and continuous time eigenvectors to match. Hence, if the initial values are the same

$$[M_c^{-1}x_c(0)] = [M_d^{-1}x_d(0)] \tag{2.32}$$

then the solutions for all future time steps *k* are the same

$$[M_c^{-1}x_c(kT)] = [M_d^{-1}x_d(kT)] \tag{2.33}$$

Therefore, the relationship between the two sets of state variables can be written

$$x_c(kT) = (M_c M_d^{-1})x_d(kT) \tag{2.34}$$

with $x_d(kT) = \begin{bmatrix} \hat{y}(k) \\ \hat{y}(k-1) \\ \hat{y}(k-2) \end{bmatrix}$ and $x_c(kT) = \begin{bmatrix} y(kT) \\ \dot{y}(kT) \\ \ddot{y}(kT) \end{bmatrix}$.

Now create the relationship of the initial conditions, when using the control inputs as in Eq. (2.20),

$$x_d(0) = \begin{bmatrix} \hat{y}(0) \\ \hat{y}(-1) \\ \hat{y}(-2) \end{bmatrix} = \begin{bmatrix} 1 & -a_2 & a_2^2 - a_1 \\ 0 & 1 & -a_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u(-1) \\ u(-2) \\ u(-3) \end{bmatrix} = T_{yu}\hat{u}(0)$$

and

$$x_c(0) = (M_c M_d^{-1} T_{yu})\hat{u}(0) \tag{2.35}$$

$$\hat{u}(0) = (T_{yu}^{-1} M_d M_c^{-1})x_c(0) \tag{2.36}$$

2.8 Examining Desired Trajectories for Feasibility

2.8.1 Asking for Zero Tracking Error of a Trajectory in Continuous Time

Examine how feasible it is to ask the third-order system in Eq. (2.9) to track the example desired trajectories in Eqs. (2.37) when the system remains at rest for all the time steps before the desired trajectory starts.

$$y_{D1} = \frac{\pi}{2}\sin(\pi t/4)$$

$$y_{D2} = \frac{\pi}{4}(1 - \cos\left(\frac{\pi t}{2}\right))$$

$$y_{D3} = \pi[5\left(\frac{t}{100T}\right)^3 - 7.5\left(\frac{t}{100T}\right)^4 - 3\left(\frac{t}{100T}\right)^5]$$

$$y_{D4} = \frac{\pi}{8}[1 - \cos\left(\frac{\pi t}{2}\right)]^2$$

(2.37)

Note that for y_{D1}, the value of output y is continuous across time $t = 0$, but the first derivative $\dot{y}(0)$ has a jump discontinuity. Trajectory y_{D2} has y and $\dot{y}$ continuous across time zero, but a discontinuity at $\ddot{y}(0)$; y_{d3} has $y(0)$, $\dot{y}(0)$ and $\ddot{y}(0)$ continuous; y_{D4} has y, $\dot{y}$, $\ddot{y}$, and $\dddot{y}$ continuous at time zero. To find the input needed to produce these trajectories as outputs, plug the desired output into the left-hand side of Eq. (10), and the result is the needed input function $u(t)$. For y_{D4}, this $u(t)$ is continuous across time zero. For y_{D3}, the needed input function has a step change, corresponding to having $u(t)$ have a jump at the start of the trajectory. This is normally reasonable and feasible in control systems.

However, computing the needed $u(t)$ at zero for y_{D2} asks for the third derivative, which is the derivative of the jump. In continuous time, and using generalized function theory, the needed $u(t)$ will have a Dirac delta function at time zero. And for y_{D1}, generalized function theory is needed to produce the first derivative of a Dirac delta function, or a unit doublet. These control input functions go to plus infinity and back to zero in infinitesimal time, or worse, go to plus infinity then to minus infinity and back to zero in infinitesimal time. No physical system can do this, which means no continuous time ILC law could produce zero tracking error to the first two kinds of trajectories. The conclusion is that it is not reasonable to ask for zero tracking error at the start of a continuous-time trajectory unless the desired trajectory has continuities with all zero conditions up through the second derivative. This requirement can be named as a smooth startup, meaning that for the above 3rd-order system, the second derivative at zero is smooth in the mathematical sense, i.e. continuous with continuous derivative. For differential equations of other orders, and in case where there are derivatives of $u(t)$ on the right of the differential equation, appropriate modifications of the requirement can be made for a smooth startup.

2.8.2 Recognizing Initial Discontinuities from Discrete-Time Desired Trajectories

As shown in the previous subsection, it is easy to know how many derivatives are continuous across time zero when the desired trajectory is specified as a function of continuous time as in Eqs. (38). Yet often the desired trajectory is specified as a set of points, i.e. $y_D(kT)$ for $k = 1,2,3, \dots, p$.

This section presents the mathematical relationship between the initial conditions in continuous time and discrete time. For a third-order difference equation, three time steps of output are needed to form the initial conditions. Consider that the first three time steps of the desired output $y_D(T)$, $y_D(2T)$, $y_D(3T)$ are produced by the initial conditions without control commands, then the corresponding y, $\dot{y}$, $\ddot{y}$ at time zero can be implied by the given desired trajectory. From the solution of the homogeneous state variable differential equation Eq. (2.9)

$$y_D(T) = C_c e^{A_c T} x_c(0)$$

$$y_D(2T) = C_c e^{2A_c T} x_c(0)$$

$$y_D(3T) = C_c e^{3A_c T} x_c(0)$$

Therefore, the implied initial values of y, $\dot{y}$, $\ddot{y}$ are given by

$$x_c(0) = \begin{bmatrix} y(0) \\ \dot{y}(0) \\ \ddot{y}(0) \end{bmatrix} = \begin{bmatrix} C_c e^{A_c T} \\ C_c e^{2A_c T} \\ C_c e^{3A_c T} \end{bmatrix}^{-1} \begin{bmatrix} y_D(T) \\ y_D(2T) \\ y_D(3T) \end{bmatrix} \tag{2.38}$$

2.8.3 Determining Feasibility of the Discrete-Time Desired Trajectory

For the continuous-time tracking problem detailed in the previous subsection, it is not possible to track if the system starts from rest state and a discontinuity of the position, velocity, or acceleration is required by the desired trajectory. But in discrete time one has zero-order hold inputs with time steps, and a Dirac delta function or unit doublet input is no longer needed. The wild function in continuous time is spread out over time, so now it may be possible to track discrete time steps on the desired trajectory from the start without exceeding actuator limits. For a given desired trajectory $y_D(kT)$, one can use the first 3 points to determine $x_c(0)$ from Eq. (2.38), then use Eq. (2.36) to determine the corresponding three control inputs $u(-3T), u(-2T), u(-T)$ needed to get to the starting conditions for the given trajectory. If these numbers are reasonable and within actuator limits, then they can be directly applied to obtain zero tracking error from the start by ILC following the given desired trajectory.

But for situations where these values are too large, measures shall be taken to take more initial steps to reduce the maximum control action magnitude needed to follow the desired trajectory from the first step without exceeding the actuator limits. One simple method is presented in this subsection. Note that while introduced more time steps are allowed, it is also desirable to limit the inputs to three linearly independent inputs, each of which generates a linearly independent transient solution. So instead of using $u(-3T), u(-2T), u(-T)$ to make these linearly independent functions by picking one nonzero value among the three for each transient solution, twice as many time steps inputs starting from $u(-6T)$ are used. Initially this creates six starting values, but only three are needed so that the resulting transient basis functions are linearly independent. In this case, pick $u(-6T) = u(-5T)$, and $u(-4T) = u(-3T)$, and $u(-2T) = u(-T)$. Propagating the solution to the needed $x_c(0)$ produces

$$x_c(0) = \left[A_{cd}^5 B_c + A_{cd}^4 B_c\right] u(-6) + [A_{cd}^3 B_c + A_{cd}^2 B_c] u(-4) + [A_{cd} B_c + B_c] u(-2) \quad (2.39)$$

where $A_{cd} = \exp{(A_c T)}$. Eq. (2.39) presents three equations with three unknown inputs. The solutions should have substantially smaller control actions. Similarly, this process can be continued to make even smaller control actions generating the needed transients while still meet the required initial conditions of the desired trajectory.

2.9 Matched Basis Function ILC Laws Including Transient Basis Functions

There are two kinds of basis functions, those associated with the transients and those associated with steady state frequency response. The sine and cosine basis functions are automatically orthogonal and can be chosen as orthonormal. Then the associated matched input basis functions are scaled in magnitude and adjusted in phase to cancel the steady state effects of going through the control system. The challenge is that the matching is based on steady state response, not the response in the finite time of the ILC problem, making them approximate. If the transients are removed from the response with the new transient basis functions, this approximation is improved.

Consider the matrix of the input basis functions $B = [\bar{B}_T \quad B_{ss}]$ and the corresponding matrix of the output basis functions $H = [\bar{H}_T \quad H_{ss}]$, where $\bar{B}_T = [\bar{B}_1 \quad \bar{B}_2 \quad \bar{B}_3]$ contains the orthogonal input basis functions for $u(-1), u(-2), u(-3)$ and $\bar{H}_T = [\bar{S}_1 \quad \bar{S}_2 \quad \bar{S}_3]$ contains the orthonormal output basis function for capturing transient response in the output. The expression for control action becomes

$$\underline{u}_j = B\underline{\alpha}_j = \bar{B}_T \underline{\alpha}_{Tj} + B_{ss}\underline{\alpha}_{ssj} \quad (2.40)$$

Note that the full set of basis functions being used is still not orthogonal, i.e., the transient basis functions are not orthogonal to steady state basis functions, so the convergence is not immediate. However, instead of orthonormalizing all the basis functions, non-orthogonality is handled by first projecting the error onto the output transient basis functions $\bar{H}_T$, then remove this part of error associated with transients from the overall error vector, and apply the sine and cosine basis functions to handle the remaining errors from steady state response. ILC Law first computes $\underline{\varepsilon}_{Tj} = \bar{H}_T^T \underline{e}_j$, and then $\underline{e}_{Tj} = \bar{H}_T \underline{\varepsilon}_{Tj}$ gives back the transient errors in the output. Decompose the tracking error into transient errors $\underline{e}_{Tj}$ and the remainder $\underline{e}_{\perp j}$

$$\underline{e}_j = \underline{e}_{Tj} + \underline{e}_{\perp j} = \bar{H}_T \underline{\varepsilon}_{Tj} + \underline{e}_{\perp j} \tag{2.41}$$

Ideally, one would compute $\underline{e}_{\perp j}$ to be the orthogonal complement, but this step is not taken here. The ILC Law, the input control updates, and the error propagation become

$$\underline{\varepsilon}_{ssj} = H_{ss}^T \underline{e}_{\perp j} \tag{2.42}$$

$$\underline{u}_{j+1} = \underline{u}_j + \phi[\bar{B}_T \quad B_{ss}]\left[\underline{\varepsilon}_{Tj}{}^T \quad \underline{\varepsilon}_{ssj}{}^T\right]^T = \underline{u}_j + \phi[\bar{B}_T \bar{H}_T^T + B_{ss} H_{ss}^T - B_{ss} H_{ss}^T \bar{H}_T \bar{H}_T^T]\underline{e}_j \tag{2.43}$$

$$\delta_{j+1}\underline{e} = -P\delta_{j+1}\underline{u} = -\phi P[\bar{B}_T \bar{H}_T^T + B_{ss} H_{ss}^T - B_{ss} H_{ss}^T \bar{H}_T \bar{H}_T^T]\underline{e}_j \tag{2.44}$$

$$\underline{e}_{j+1} = \{I - \phi P[\bar{B}_T \bar{H}_T^T + B_{ss} H_{ss}^T - B_{ss} H_{ss}^T \bar{H}_T \bar{H}_T^T]\}\underline{e}_j \tag{2.45}$$

Similar as presented in Section 2.2, in order to have a stable learning process, all the eigenvalues of matrix $I - \phi P[\bar{B}_T \bar{H}_T^T + B_{ss} H_{ss}^T - B_{ss} H_{ss}^T \bar{H}_T \bar{H}_T^T]$ in Eq. (2.45) should be no larger than 1.

2.10 Numerical Results

This section presents numerical experiment results of the matched basis function ILC with and without transient basis functions. The desired trajectory is $y_1^* = \sin(4\pi t)$, and, as discussed in the previous sections, it does not have a smooth startup from all zero initial states. Consider a

3rd-order system with continuous time transfer function $G(s) = (\frac{a}{s+a})(\frac{\omega^2}{s^2+2\xi\omega s+\omega^2})$, where $a = 8.8$, $\xi = 0.5$, and $\omega = 37$, fed by a zero-order hold sampling at 100Hz, pick the number of basis functions N=20 and set the learning gain to $\phi = 0.01$. Figures 3 and 4 give the control history at the 4000th iteration, at which we are confident that the tracking has reached convergence, without using transient basis functions and using transient basis functions respectively. By comparing Figure 3 and 4, it can be seen that, with the help of transient basis functions, the input magnitudes at initial and last several time steps have been significantly reduced from around 10 to 4, which is much closer to the peak magnitude 2 at the steady state; and the control action generally becomes much smoother as well after applying the transient basis functions.

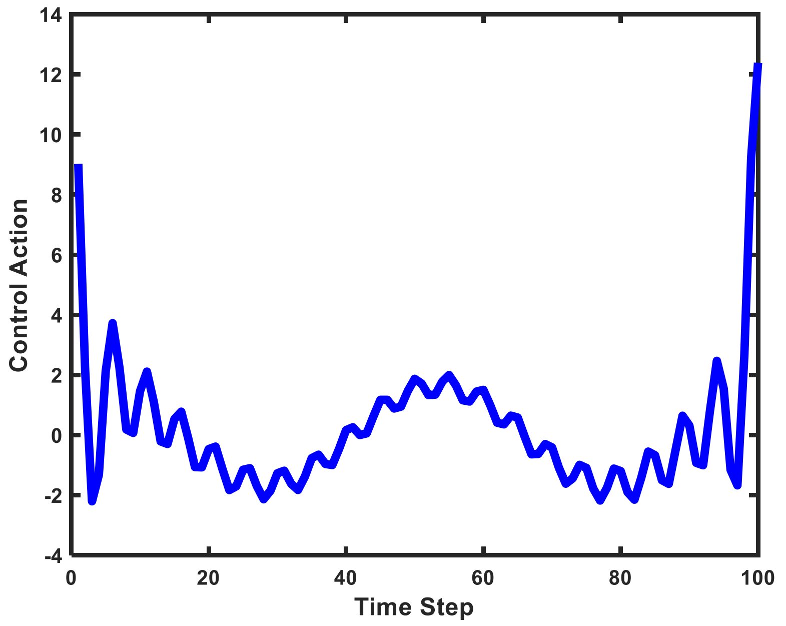


Figure 3. Control action in the 4000th iteration without transient basis functions tracking y_1^*.

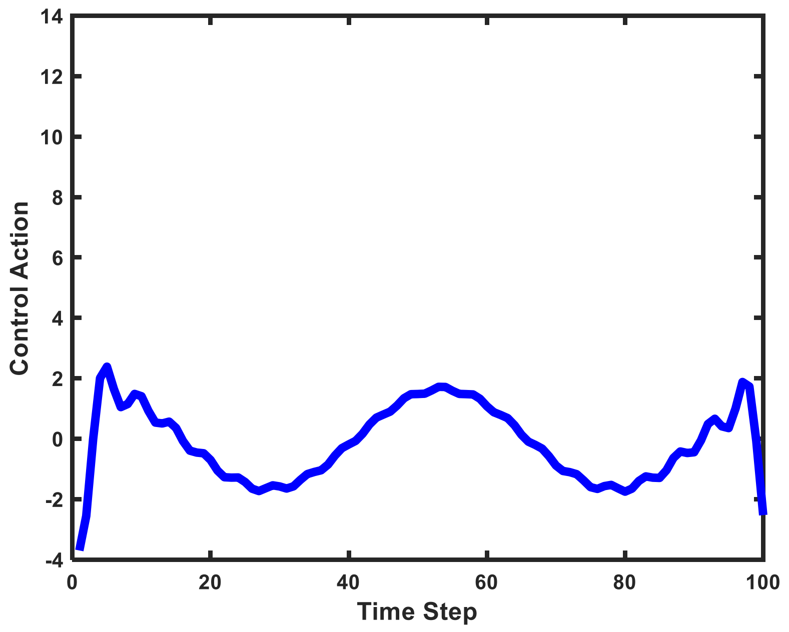


Figure 4. Control action in the 4000th iteration with transient basis functions tracking y_1^*.

The corresponding tracking error histories are given in Figures 5 and 6 respectively. The tracking error has reduced from the range of 0.03 to -0.03 to the range bounded by 0.01 and -0.01, which is a considerable improvement and the improvement is most obvious at the start of trajectory when the initial condition effects are expected to be addressed. As expected, when transient basis functions are applied on a trajectory with “smooth” startup and therefore has no need for these initial condition basis functions, it deteriorates the performance.

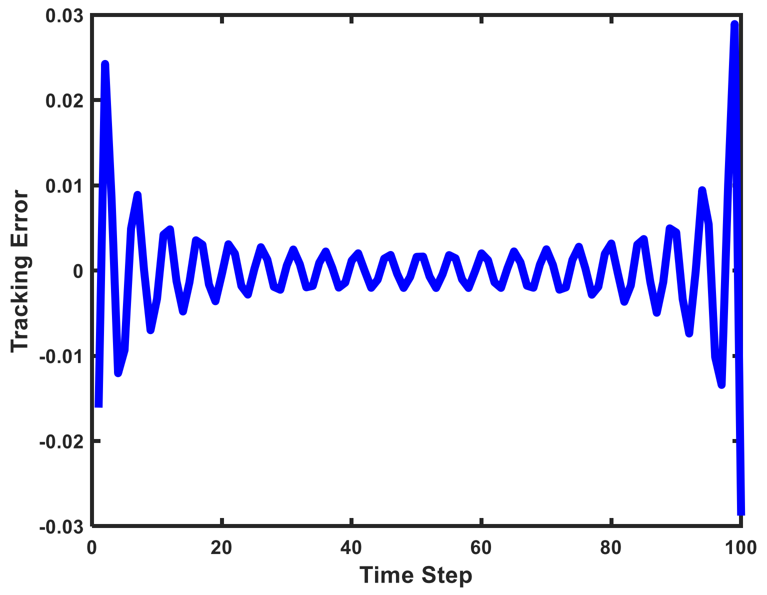


Figure 5. Tracking error in the 4000th iteration without transient basis functions.

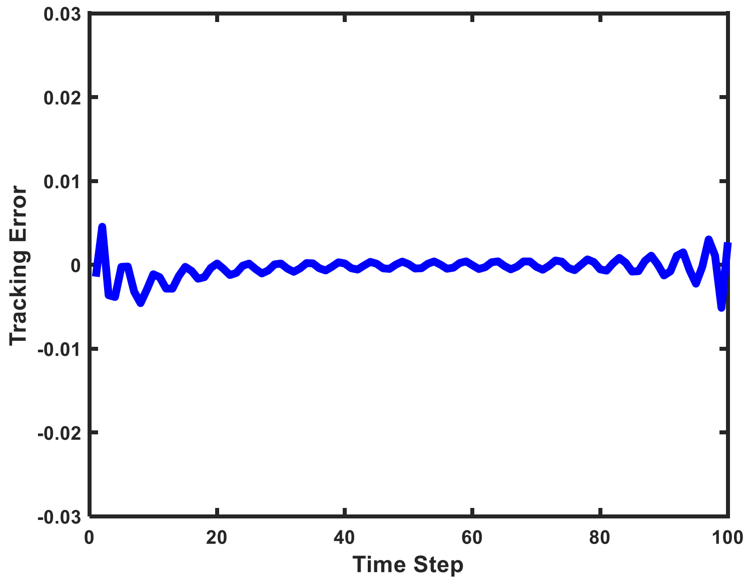


Figure 6. Tracking error in the 4000th iteration with transient basis functions.

2.11 Summary and Discussion

Design of ILC using basis functions has several advantages. (1) It can automatically stabilize the inverse problem, because the output basis function(s) related with the output singular vector(s) associated with instability is not likely to be picked. This singular vector(s) associated with instability corresponds to the singular value(s) of Toeplitz matrix P produced by the unstable zero(s). Therefore, such basis functions can be deliberately avoided. (2) Normally, ILC applications require the use of a zero-phase low-pass filter to cut off the learning at high frequencies where parasitic poles or residual modes not presented in the model can destabilize the learning process. When picking the output basis functions of interest, this high frequency model error issue can be easily bypassed.

Yet the new design issue introduced in ILC is how to make a good choice of basis functions. Using Fourier series, any finite time trajectory can always be represented with sinusoids. These can capture the behavior of a control system in steady state, and in this case the system frequency response can be used to know what input would eliminate each error component immediately, i.e. the associated matched input basis functions. The basis functions in this chapter does this, but only for the steady state part, and when decaying functions in the transients are part of the error history, many sinusoids would be needed to capture such a function, which are irrelevant.

The contribution of this chapter is that new basis functions to specifically capture the transients are developed. The original matched basis functions are basis functions for the output error, and the associated matched input forcing functions that cancel this output error. The challenging fact is that transients are solutions of the homogeneous equation and hence have no input functions. The matching is accomplished by appending a number of time steps equal to the order of the system before the start of the tracking and adjust initial conditions. They produce the transient solutions that are the associated output basis functions. Assuming the system is at rest before the start of the desired tracking maneuver, these basis functions learn how to reach the initial conditions needed, to be on the desired trajectory from time zero onward. If the desired trajectory starts very smoothly, then these newly developed transient basis functions are not needed. However, it is very easy to ask for a desired trajectory that does not satisfy this condition, and how to recognize when the desired trajectory needs these new basis functions is shown in this chapter. It is also shown how to test whether the trajectory tracking from time zero is feasible for given actuator strength, and in case it is not, it is shown how to use additional initial time steps to make it feasible.

Chapter 3 Limiting Error Accumulation in Unaddressed Space for Basis Function Iterative Learning Control with Singular Vector Basis Functions

3.1 Introduction

In Chapter 2, the transient response in the output as a result of the finite time nature of ILC is addressed by generating initial condition basis functions. They are fundamentally different from the basis functions discussed so far, which are input functions that map to output functions. Initial condition response is response of a homogeneous equation, which has no input and therefore no input basis function. Instead, a number of time steps is appended to the start of the trajectory, the number being equal to the order of the governing difference equation. The inputs in these new time steps are learned aiming to capture the transients. The remaining basis functions are chosen as matching steady state frequency responses up to the desired frequency.

Various other approaches of picking basis functions appear in the literature. The input basis functions restrict the input to be a linear combination of these functions. The span of the output basis functions is the chosen tracking error space - the aim is to eliminate the error components in this space, denoted as the addressed error space. The remaining part of the output space is termed the unaddressed output space.

This chapter identifies a potentially important issue in basis function ILC: while the ILC is reducing the error in the addressed part of the output space, it can be accumulating error in the unaddressed part of the space as the learning process goes on. Formulas are derived to indicate the behavior of error accumulation in the unaddressed output space, i.e., how this error accumulates with ILC iterations, and give the final value of the accumulated error.

It is suggested that the best choice for input and output basis functions have the property that the input basis function space maps onto the output basis function space. If this is done perfectly, then there will be no error accumulation in the unaddressed part of the output space. It is even better if it is possible to use matched basis functions, having each input basis function be orthogonal and map to a corresponding output basis function, the output basis functions also being orthonormal.

This chapter creates an ideal solution to the matching problem: first, form the singular value decomposition of the input-output system matrix of Markov parameters; the input basis functions are then picked from the right singular vectors (orthonormal but then scaled), and the output orthonormal basis functions as the corresponding output singular vectors. Ideally, a desired output that is a linear combination of the chosen output basis functions is picked. Note that the singular value decomposition of this matrix is closely related to the system's steady state frequency response (Chen et al., 2003), so these basis function choices can be guided by the desire to have good tracking up to some frequency. At the same time, this can avoid the issues of high frequency unmodeled dynamics. It is shown here that if the model used to design the basis function ILC is correct and free of model error, then there is no error accumulation in the unaddressed part of the output error space. Formulas are derived to describe the error accumulation that results from using an inaccurate model.

3.2 Singular Vector Basis Function ILC

Basis function ILC chooses to restrict the control objective to eliminating error in some subspace of the full p-dimensional output space, and correspondingly confine the control action to a subspace of its full p-dimensional input space, where p is the number of time steps in one ILC iteration. This can be done to reduce the computations needed, and to avoid a set of robustness issues that ILC normally faces. As discussed above, one approach is to simply pick a set of orthonormal Legendre polynomials, ask that the control input be a linear combination of these, whose task is to learn to eliminate the output error projected onto the same Legendre polynomial output basis functions (Phan et al., 1996). Such an approach is described as using unmatched basis functions, where no attempt is made to have the image of the input basis function space be the same as the output basis function space when mapped through the system. The approach described above using the concept of Fourier expansions (Frueh et al., 2000) can also be a totally unmatched approach that just expands the input and the output on sine and cosines up to the chosen frequency. However, in this case, there is the option of using system frequency response information, adjusting the magnitude and phase of the input basis functions to cancel the phase and magnitude change through the system, to produce the output basis functions – after the transients have become negligible. This makes matched basis function only in steady state. Chapter 2 tries to fix this issue, by creating extra basis functions to capture the transients produced by the initial conditions.

Here a different approach is used to create truly matched input and output basis functions, creating the basis functions from input and output singular vectors of matrix $P = USV^T$. Partition each matrix according to

$$U = [U_A \quad U_U] \;\; ; \quad S = \begin{bmatrix} S_A & 0 \\ 0 & S_U \end{bmatrix} \quad ; \quad V = [V_A \quad V_U] \tag{3.1}$$

where subscript "A" denotes quantities associated with the "addressed" part of the output error, and subscript "U" denotes the orthogonal error components that are not being corrected, the "unaddressed" part. Collect the chosen output basis functions from matrix U into matrix U_A as the orthonormal output singular vectors that span the addressed output space. Call the ith column U_{Ai}, and similarly for V_{Ai}, and denote the ith diagonal element of S_A by σ_{Ai}. Examine $\delta_j \underline{y} = P\delta_j \underline{u}$. If

$\delta_j \underline{u} = V_{Ai}$ then the output is $\delta_j \underline{y} = \sigma_{Ai} U_{Ai}$. Therefore, pick the input to be the linear combination β_i of input basis functions given as $\delta_j \underline{u} = V_A (S_A)^{-1} \beta_i$. Then the output, making use of the matched input and output basis functions, is the same linear combination of output basis functions given by $\delta_j \underline{y} = U_{Ai} \beta_i$. Each input basis function maps to a corresponding output basis function, the output basis functions are orthonormal, and the input basis functions are orthogonal and scaled to produce normalized output functions by canceling the gain going through the system.

Convert the error space to error in the addressed and the unaddressed parts of the space

$$\underline{\varepsilon}_j = \begin{bmatrix} \underline{\varepsilon}_{A,j} \\ \underline{\varepsilon}_{U,j} \end{bmatrix} = U^T \underline{e}_j = \begin{bmatrix} U_A^T \underline{e}_j \\ U_U^T \underline{e}_j \end{bmatrix} \tag{3.2}$$

And convert the input space as

$$\underline{u}_j = V(S)^{-1} \underline{\mu}_j = V(S)^{-1} \begin{bmatrix} \underline{\mu}_{Aj} \\ \underline{\mu}_{Uj} \end{bmatrix} = V_A (S_A)^{-1} \underline{\mu}_{Aj} \tag{3.3}$$

Because the input is restricted to the input basis function space, $\underline{\mu}_{Uj} = 0$. The ILC law becomes

$$\underline{\mu}_{A,j+1} = \underline{\mu}_{Aj} + L_A \underline{\varepsilon}_{A,j} \tag{3.4}$$

With L_A the identity matrix, this ILC law corrects all of the error in the addressed part of the output space in one iteration. In the presence of model error, it can be beneficial to use slower learning. This can be done using a single gain $L_A = \phi I$ or done independently for each basis function using

$$L_A = diag(\phi_1, \phi_2, \dots, \phi_N) \tag{3.5}$$

where N is the number of basis functions chosen. This ILC law can be rewritten in terms of the original input and output variables by pre-multiplying by $V_A(S_A)^{-1}$ on each side

$$\underline{u}_{j+1} = \underline{u}_j + (V_A S_A^{-1}) L_A U_A^T \underline{e}_j \tag{3.6}$$

To analyze the convergence of this algorithm, start with $\delta_{j+1}\underline{e} = -P\delta_{j+1}\underline{u}$, and premultiply by U_A^T and by U_U^T to find $\delta_j \underline{\varepsilon}_A$ and $\delta_j \underline{\varepsilon}_U$. Replace $\delta_{j+1}\underline{u} = (V_A S_A^{-1})\delta_{j+1}\underline{\mu}_A$ and use the ILC control law $\delta_{j+1}\underline{\mu}_A = L_A \underline{\varepsilon}_{A,j}$ to obtain

$$\underline{\varepsilon}_{A,j+1} = E\underline{\varepsilon}_{A,j} \tag{3.7}$$

$$\underline{\varepsilon}_{U,j+1} = \underline{\varepsilon}_{U,j} - F\underline{\varepsilon}_{A,j}$$

where

$$E = I - U_A^T P V_A S_A^{-1} L_A \tag{3.8}$$

$$F = U_U^T P V_A S_A^{-1} L_A$$

Using the partitioned singular value decomposition of P, these reduce to

$$\underline{\varepsilon}_{A,j+1} = (I - L_A)\underline{\varepsilon}_{A,j} \tag{3.9}$$

$$\underline{\varepsilon}_{U,j+1} = \underline{\varepsilon}_{U,j}$$

If L_A is the identity matrix, then the error in the $\underline{\varepsilon}_A$ part of the space becomes zero in one iteration, while the error in the $\underline{\varepsilon}_U$ part of the space is unaltered. If it is not the identity matrix, the error in the ith component at iteration j has decayed by the factor $(1-\phi_i)^j$, provided $1-\phi_i$ is chosen as less than one in magnitude.

3.3 Basis Function System Inverse Control

The emphasis of this chapter is on ILC, but the basic function modelling in the previous section could be used to simply compute something similar to an inverse control action. By limiting the inverse to the defined matched basis function spaces, one can get a command input that bypasses the common instability of the actual discrete system inverse and the difficulty posed by unmodeled high frequency poles or modes. The relationship of the singular value decomposition of matrix P to the frequency response, allows one to make an inverse up to a desired cutoff frequency as modified by the finite time of the problem (Chen et al., 2003). Set $\underline{y} = \underline{y}^*$ in Eq. (2.2) and solve for $P\underline{u}$. Premultiply by U^T and write the partitions as

$$S_A^{-1}U_A^T\left[\underline{y}^* - \left(\overline{A}x(0) + \underline{v}\right)\right] = V_A^T\underline{u} = \beta_A \tag{3.10}$$

$$S_U^{-1}U_U^T\left[\underline{y}^* - \left(\overline{A}x(0) + \underline{v}\right)\right] = V_U^T\underline{u} = \beta_U$$

Then restrict the control action to have components on the unit vectors of V_A and no components on V_U so that one only aims to satisfy the first of these equations. Since $\underline{u} = V_A\beta_A$, the control action is

$$\underline{u} = V_A S_A^{-1}U_A^T\left[\underline{y}^* - \left(\overline{A}x(0) + \underline{v}\right)\right] \tag{3.11}$$

Notice that in ILC there was no need to know $x(0)$ and $\underline{v}$. Their effect is contained in the first run, and all later runs operate on the changes δ_j. These terms are the same every run, so they drop out when one applies this difference operator to update the control action. Hence, if one does not know know $x(0)$ and $\underline{v}$, one can do one ILC iteration with $L_A = I$, with input $\underline{u}_0$ producing output $\underline{y}_0$ in the first run. Then the basis function inverse control action to eliminate error in the basis function space based on one's model is

$$\underline{u}_1 = \underline{u}_0 - V_A S_A^{-1} U_A^T (\underline{y}_0 - \underline{y}^*) \tag{3.12}$$

3.4 The Influence of Model Error on Singular Vector Basis Function ILC

One of the main purposes for ILC is to converge to good tracking error in the world in spite of error in the model used to design the ILC law. Let our model be $P_M = USV^T$ with the same A and U partitions as before, while the real world model is P_W. Equation (2.6) becomes

$$\delta_{j+1}\underline{e} = -P_w \delta_{j+1}\underline{u} \tag{3.13}$$

This equation has rewritten the P in Eq. (2.6) to explicitly indicate that this is mapping the real world input-output relationship. The remaining equations in Eqs. (3.1) to (3.6) are calculated based on the model available for ILC design P_M. The model error is included in the analysis of convergence following the steps as outlined above Eq. (3.7). This results in the same form of error propagation from run to run as in Eq. (3.7), but with modified coefficient matrices

$$\underline{\varepsilon}_{A,j+1} = E\underline{\varepsilon}_{A,j}$$

$$\underline{\varepsilon}_{U,j+1} = \underline{\varepsilon}_{U,j} - F\underline{\varepsilon}_{A,j}$$

$$\tag{3.14}$$

$$E = I - U_A^T P_w V_A S_A^{-1} L_A$$

$$F = U_U^T P_w V_A S_A^{-1} L_A$$

Because the singular value decomposition of model P_M and of the world P_w do not match, these coefficients do not simplify as they do in Eq. (3.9), and one is left with Eq. (3.7) to describe the error propagation. The error in the addressed part of the output space $\underline{\varepsilon}_{A,j}$ will converge to zero for all errors in the initial run, if and only if the model error is such that all eigenvalue magnitudes of matrix E are less than one. This is expected to happen provided the model error is not too large. The orthonormal basis functions involved in SVD make an ILC law that is numerically rather robust to error, so there is considerable tolerance to error. What is new is that the error in the unaddressed part of the space has a forcing function involving the error in the unaddressed part of the space. Error is accumulating in this part of the space as the iterations progress.

3.5 Model Error Induced Accumulation in the Unaddressed Error Space

Without model error, the output error in the unaddressed part of the space was totally unaffected by the learning process. Using Eqs. (3.14), error accumulation in the unaddressed part of the space can be studied in the presence of model error. Rewrite these equations as

$$\underline{\varepsilon}_{A,j+1} = E^{j+1}\underline{\varepsilon}_{A,0} \tag{3.15}$$

$$\begin{aligned}\underline{\varepsilon}_{U,j+1} &= \underline{\varepsilon}_{U,0} - F\left[\underline{\varepsilon}_{A,0} + \underline{\varepsilon}_{A,1} + \cdots + \underline{\varepsilon}_{A,j}\right] \\ &= \underline{\varepsilon}_{U,0} - F\left[I + E^{1} + \cdots + E^{j}\right]\underline{\varepsilon}_{A,0} \\ &= \underline{\varepsilon}_{U,0} - F\Sigma(j)\underline{\varepsilon}_{A,0}\end{aligned}$$

Note that

$$\Sigma(j) - E\Sigma(j) = I - E^{j+1}$$

$$\Sigma(j) = [I - E]^{-1}[I - E^{j+1}] \tag{3.16}$$

Hence, the accumulated error in the unaddressed part of the space at iteration $j + 1$ is given by

$$\underline{\varepsilon}_{U,j+1} = \underline{\varepsilon}_{U,0} - F[I - E]^{-1}[I - E^{j+1}]\underline{\varepsilon}_{A,0} \tag{3.17}$$

If the maximum magnitude of the eigenvalues of E is less that unity, including the effects of model error, then $\underline{\varepsilon}_{A,j}$ converges to zero, and E^j converges to the zero matrix as j tends to infinity. The final error in the unaddressed part of the space, accumulated as the iterations tend to infinity is given by

$$\underline{\varepsilon}_{U,\infty} = \underline{\varepsilon}_{U,0} - F[I - E]^{-1}\underline{\varepsilon}_{A,0} \tag{3.18}$$

Recall that one of the main advantages of using ILC is its ability to aim for zero error in the world in spite of model error. Its updates of each iteration are based on what the real world does with the command, not based on what the model used to design the ILC does with the command. It can easily happen that ILC converges to an error level below that obtained by inverting one's model when such inversion is possible. This suggests that this accumulation of error could be a significant issue.

A logical question to ask is: is it perhaps best to use learning gains in L_A that are large, with the idea that accomplishing the learning in a few iterations would limit how much accumulation there is. On the other hand, one might think that learning very gently would make the contribution small during each iteration. To study this, examine

$$F[I - E]^{-1} = [U_U^T P_w V_A S_A^{-1} L_A][U_A^T P_w V_A S_A^{-1} L_A]^{-1} = [U_U^T P_w V_A][U_A^T P_w V_A]^{-1} \tag{3.19}$$

This implies that the final value of the error is the same no matter what gains are used in L_A, the gains only influence how fast the error in the unaddressed part of the space converges to the steady state value.

3.6 Choosing Unmatched Basis Functions Results in Error Accumulation Even with a Perfect Model

This chapter shows how to create fully matched basis functions for use in ILC, and a major point is to demonstrate that matched basis functions should be used. The previous sections showed there was no accumulation of error if the basis functions were matched using a perfect model. This section derives the analogous error accumulation study when no attempt is made to pick matched basis functions. For generality, model error is also included in the analysis. Unlike the previous analysis, having no model error causes no simplification of the formulas involved.

Restrict $\underline{u}$ to be a linear combination of orthogonal input basis functions that are the columns of matrix *G*, so that $\underline{u} = G\underline{\alpha}$, where $\underline{\alpha}$ is a column vector. There are *N* columns of basis functions, and *p* entry in each column. The orthogonal complement space is spanned by the p - *N* orthonormal columns of $G_{\perp}$. Pick the basis functions for the output space to be the *N* orthonormal columns of matrix *H*, with the orthogonal complement space represented by $H_{\perp}$. Analogous to the previous definitions, define $\underline{\varepsilon}_A = H^T\underline{e}$ as the addressed error vector, and $\underline{\varepsilon}_U = H_{\perp}^T\underline{e}$ as the unaddressed error vector. Using these equations, $\delta_{j+1}\underline{e} = -P_W\delta_{j+1}\underline{u}$, and $\delta_{j+1}\underline{u} = G\delta_{j+1}\underline{\alpha}$, one can write

$$\delta_{j+1}\underline{\varepsilon}_A = \underline{\varepsilon}_{A,j+1} - \underline{\varepsilon}_{A,j} = -H^T P_w G\delta_{j+1}\underline{\alpha} \tag{3.20}$$

$$\delta_{j+1}\underline{\varepsilon}_U = \underline{\varepsilon}_{U+1} - \underline{\varepsilon}_{U,j} = -H_{\perp}^T P_w G\delta_{j+1}\underline{\alpha}$$

From the first of these equations, if one asks for zero error for $\underline{\varepsilon}_{A,j+1}$, one would pick

$$\underline{\alpha}_{j+1} = \underline{\alpha}_j + (H^T P G)^{\dagger}\underline{\varepsilon}_{A,j} \tag{3.21}$$

The † symbol is used to indicate the inverse, if it exists, and otherwise a Moore-Penrose pseudoinverse. Note that if the image of the input basis function space does not cover all of the output basis function space, one should need to use the pseudoinverse. Use of Legendre polynomials should encounter this problem. Use of Fourier series also has a mismatch because of the finite time nature of ILC problems involving initial transients. A learning gain matrix L_A as before can be introduced into the basis function ILC law to adjust the speed of convergence

$$\underline{\alpha}_{j+1} = \underline{\alpha}_j + (H^T PG)^{\dagger} L_A \underline{\varepsilon}_{A,j} \tag{3.22}$$

One has the choice to insert this matrix either before or after the $(H^T PG)^{\dagger}$. If it is in front of this factor, it directly controls the size of the update increment in $\underline{\alpha}_j$, i.e. the factor by which the associated input basis function is changed in the update. If it is after this factor, it directly influences how much emphasis one wants to put on learning the error component on the associated output basis function.

Substituting this $\delta_{j+1}\underline{\alpha}$ into Eqs. (3.20) produces Eqs. (3.14) with new E and F

$$\underline{\varepsilon}_{A,j+1} = E\underline{\varepsilon}_{A,j}$$

$$\underline{\varepsilon}_{U,j+1} = \underline{\varepsilon}_{U,j} - F\underline{\varepsilon}_{A,j}$$

(3.23)

$$E = [I - (H^T P_w G)(H^T PG)^{\dagger} L_A]$$

$$F = (H_{\perp}^T P_w G)(H^T PG)^{\dagger} L_A$$

The situation is more complicated than previously. One possibility is that E is full rank with all eigenvalues less than unity in magnitude. In this case, the analysis in Eqs. (3.15) through

(3.18) still applies. This situation applies if the image of the input basis function space covers the whole addressed output space. Equation (3.18) predicts the final error level again, but one should expect substantially larger errors. The final error level is now determined by

$$F[I-E]^{-1}=(H_{\perp}^{T}P_{w}G)(H^{T}PG)^{\dagger}L_{A}[(H^{T}P_{w}G)(H^{T}PG)^{\dagger}L_{A}]^{-1} \qquad (3.24)$$

For this choice of location for the gain matrix L_A, the final error level is again not a function of the gains in this matrix.

Otherwise one has to decompose the intended addressed error space into that part that can be reached from the input space, and that part that is now additional unaddressed output space. To make this decomposition, take the singular value decomposition of $(H^{T}P_{w}G)(H^{T}PG)^{\dagger}L_{A}$. The input singular vectors associated with zero singular values do not influence the output in the addressed part of the defined output space. This is the null space of the mapping in which E is simply an identity matrix. The associated output singular vectors of E span the unreachable part of the addressed output space. The error does not update in this part of the space. The concept of Venn Diagrams is helpful here. The set of all inputs in the input basis function space maps into two parts in the output space, one part maps into the addressed part of the output space, the space that the nonzero singular values allow E to influence, and the other part maps into the remainder of the space.

3.7 Introducing Initial Condition Basis Functions

Chapter 2 develops a different kind of basis function that aims to capture the transients. Extra time steps are introduced before the start of the problem, the number equal to the order of the governing scalar difference equation. For simplicity, consider a third order system. Since three outputs in succession are valid initial conditions for the homogeneous difference equation, these three control actions can produce functions to span the space of possible transients. The approach finds the projection of the error onto these transient basis functions, then finds the complement of this error space and applies the singular vector matched basis functions to learn this part of the error space. The potential uses for this are:

(1) Suppose that there is no $\underline{v}$ in Eq. (2.2), so that $\underline{y} = P\underline{u} + \overline{A}x(0)$, then the singular vector basis functions handle the input-output relationship of $P\underline{u}$, and the initial condition basis functions handle the initial conditions to output basis functions for $\overline{A}x(0)$. Therefore, one can use these to create the basis function system inverse of Eq. (3.11) without knowing $x(0)$.

(2) An initial motivation for generating the initial condition basis functions was to combine them with the steady state frequency response basis function, so that transients would not make accumulation in the unaddressed part of the output error space.

(3) This approach can be used to improve performance when the desired trajectory does not have a "smooth" startup as defined in Chapter 2: a desired trajectory that is feasible starting from all zero initial conditions, i.e. one that has enough derivatives that are zero at time zero so that the command input to produce the trajectory does not require more than a step change at time zero. The discrete time ILC can mathematically produce an input producing zero output at every time step for commands that are not smooth in this sense, because it only asks for zero error at the sample times. But the command is a discrete version of the violent motions just described. If one would like to have the command without this behavior near the start of the trajectory, then one can use the initial condition basis functions which include some extra time steps before time zero. During these time steps the inputs can be effectively adjusting the initial conditions associated with time zero, so that they match the derivatives of the desired trajectory at time zero.

To further illustrate the concepts, consider the third order system discussed above, with three initial time steps introduced before time zero. Unit inputs in the control for each step produce output histories in $\underline{y}$ given by $\underline{S}_1$, $\underline{S}_2$, and $\underline{S}_3$. These controls can produce any outputs $y(k)$ for these first three steps, which means that any possible initial condition can be generated. The output history can be written as

$$\underline{y} = [\underline{S}_1 \quad \underline{S}_2 \quad \underline{S}_3]\begin{bmatrix} u(-1) \\ u(-2) \\ u(-3) \end{bmatrix} + P\underline{u}_S \tag{3.25}$$

$$\underline{y} = \underline{H}_T\underline{u}_T + P\underline{u}_S \qquad \underline{H}_T = [\underline{S}_1 \quad \underline{S}_2 \quad \underline{S}_3]$$

$$\underline{y} = \overline{H}_T\underline{\beta} + P\underline{u}_S \qquad \overline{H}_T = [\overline{S}_1 \quad \overline{S}_2 \quad \overline{S}_3] \tag{3.26}$$

$$\underline{u}_T = \overline{B}_T\underline{\beta} \quad \underline{H}_T = \overline{H}_T\overline{B}_T \quad \overline{B}_T = [B_1 \quad B_2 \quad B_3] \tag{3.27}$$

The $\underline{H}_T$ contains unit pulse response histories for time steps 1 through p, produced by unit pulses at $u(-i), i = 1,2,3$. The $\overline{H}_T$ contains three orthonormal vectors spanning the same space as $\underline{H}_T$, and $\underline{\beta}$ gives the components on these vectors. The $\underline{u}_T$ is the 3-step input history that is supposed to match the influence of the $x(0)$ initial conditions. The $\underline{u}_S$ is the usual p time steps of control history which will be designed to eliminate the error history that would be produced by zero initial conditions, or the error produced after the influence of the nonzero initial conditions has been removed from the p step error history.

The first objective is to make $\underline{\beta}$ capture the transients produced by $x(0)$, i.e. the components of $\underline{e}_j$ on $\underline{H}_T$, or $\overline{H}_T^T\underline{e}_j$. The learning law for this is

$$\underline{\beta}_{j+1} = \underline{\beta}_j + L_T\overline{H}_T^T\underline{e}_j \tag{3.28}$$

The next objective is to remove the $x(0)$ influence on the error history $\underline{e}_j$, resulting in $\underline{e}_{\perp,j}$

$$\underline{e}_{\perp,j} = \underline{e}_j - \overline{H}_T \overline{H}_T^T \underline{e}_j$$

$$\underline{e}_{\perp,j} = [I - \overline{H}_T \overline{H}_T^T] \underline{e}_j$$

$$\underline{e}_{\perp,j} = \overline{H}_\perp \underline{e}_j \tag{3.29}$$

Eqs. (3.2) and (3.4) now apply to $\underline{e}_{\perp,j}$ instead of $\underline{e}_j$, resulting in the learning law

$$\underline{\mu}_{A,j+1} = \underline{\mu}_{Aj} + L_A \underline{\varepsilon}_{A,j}$$

$$\underline{\mu}_{A,j+1} = \underline{\mu}_{Aj} + L_A U_A^T \underline{e}_{\perp,j}$$

$$\underline{\mu}_{A,j+1} = \underline{\mu}_{Aj} + L_A U_A^T \overline{H}_\perp \underline{e}_j \tag{3.30}$$

As before, the unaddressed part of the space is not updated

$$\underline{\mu}_{U,j+1} = \underline{\mu}_{Uj} \tag{3.31}$$

Eq. (10) again applies to get back to the actual control actions

$$\underline{u}_j = V_A (S_A)^{-1} \underline{\mu}_{Aj} \tag{3.32}$$

To develop the error propagation formula for the learning law including initial condition basis functions, write the error at $j+1$ iteration

$$\underline{e}_{j+1} = \underline{y}^* - \underline{y}_{j+1} = \underline{y}^* - \overline{H}_T \underline{\beta}_{j+1} - P V_A (S_A)^{-1} \underline{\mu}_{A,j+1} \tag{3.33}$$

and subtract the error at j. Use Eqs. (3.28) and (3.30) to replace the resulting differences in $\underline{\beta}$ and $\underline{\mu}_A$, to obtain

$$\underline{e}_{j+1} = [I - \overline{H}_T L_T \overline{H}_T^T - PV_A(S_A)^{-1} L_A U_A^T \overline{H}_\perp]\underline{e}_j$$

$$\underline{e}_{j+1} = [I - \overline{H}_T L_T \overline{H}_T^T - U_A L_A U_A^T \overline{H}_\perp]\underline{e}_j \tag{3.34}$$

Now one can convert the error to the space in which addressed and unaddressed errors are defined. Use Eq. (3.2), $\underline{\varepsilon}_j = U^T \underline{e}_j$, $\underline{e}_j = U\underline{\varepsilon}_j$, but this time apply it to $\underline{e}_j$, instead of $\underline{e}_{\perp,j}$, to obtain the final result

$$\underline{\varepsilon}_{j+1} = U^T \left[I - \overline{H}_T L_T \overline{H}_T^T - U_A L_A U_A^T \overline{H}_\perp\right] U\underline{\varepsilon}_j \tag{3.35}$$

3.8 Numerical Simulations

3.8.1 Tracking Performance Using Matched Basis Function ILC without Model Error

Consider a 3rd-order system with continuous time transfer function $G(s) = \left(\frac{a}{s+a}\right)\left(\frac{\omega^2}{s^2+2\xi\omega s+\omega^2}\right)$, where $a = 8.8$, $\xi = 0.5$, and $\omega = 37$, fed by a zero-order hold sampling at 100Hz. Pick the desired one-second trajectory $y_1^* = \sin(4\pi t)$, and pick the first N = 10 singular vectors for the addressed basis functions. Use a learning matrix L_A with all the learning gains $\phi_i = 0.2$. This slows down the learning so that the convergence process can be observed. The RMS of the overall error is given in Figure 7 which converges quickly on a linear scale to a value right below 0.1. The RMS of the error in the addressed part of the output space is shown in Figure 8, and it gradually converges to 10^{-16}, a numerical zero. Figure 9 gives the error in the unaddressed part of the output space, and, as predicted by the mathematics, it remains as a constant. Figure 9 is also consistent with Figure 7 in explaining why the RMS of overall error converges to somewhere a bit below 0.1 instead of numerical zero, as it is the error in the unaddressed part that dominates the final level of overall output error.

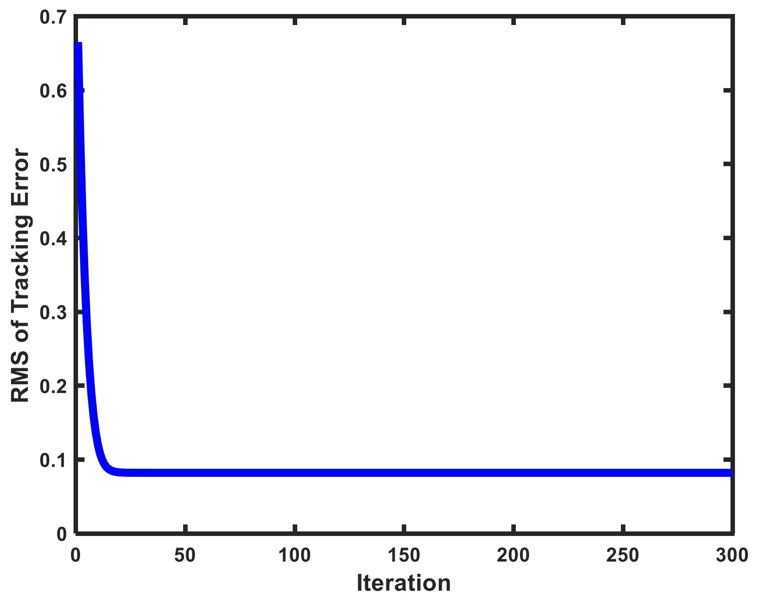


Figure 7. RMS of overall tracking error vs. iteration when there is no model error with *N*=10.

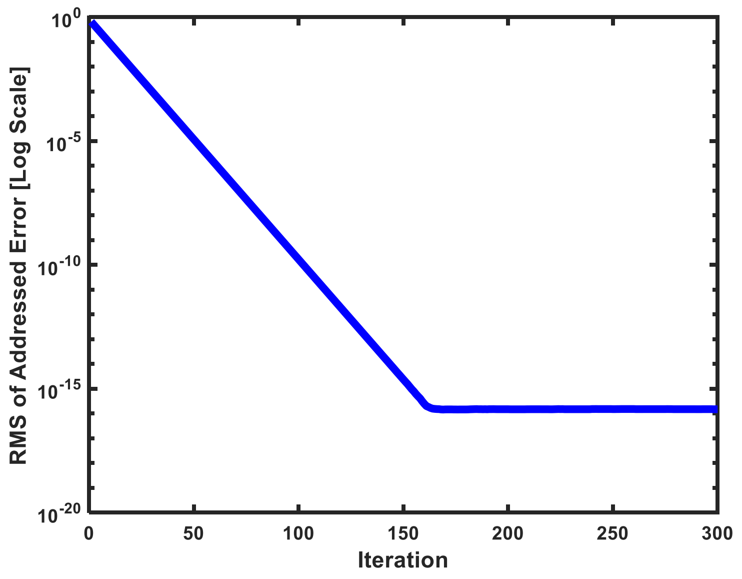


Figure 8. RMS of addressed error vs. iteration in Log Scale associated with Figure 7.

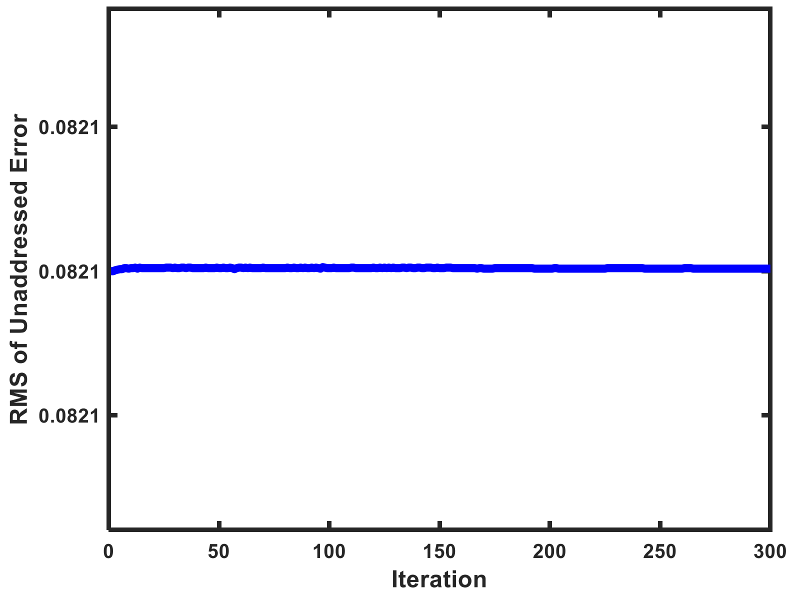


Figure 9. RMS of unaddressed error vs. iteration associated with Figure 7.

The overall error converges to the error of the desired trajectory is represented by the chosen output basis functions, i.e. $\underline{y}_1^* - U_A(U_A^T\underline{y}_1^*)$. Figure 10 studies the amount of error in the representation as a function of *N*, the number of basis functions chosen. The error stops changing for *N* larger than about 60 up to 99. Including the last singular vector, the 100th singular vector, produces the anomalous values indicated, whose singular vectors produce instability of the control action. This is associated with the one particularly small singular value which is computed to be 1.8400×10^{-18}. It is related to the zero outside the unit circle, and is not associated with magnitude frequency response as is the next to the smallest singular value 5.0028×10^{-4}. Figures 7 to 10 are for y_1^* which does not correspond to a "smooth" start up as described above. Figure 11 gives the corresponding plot for $y_2^* = (1 - \cos{(4\pi t)})^2$ which does correspond to a "smooth" startup. This time the error in the addressed part of the space associated with $\underline{y}_2^* - U_A(U_A^T\underline{y}_2^*)$ is a few orders of magnitude smaller.

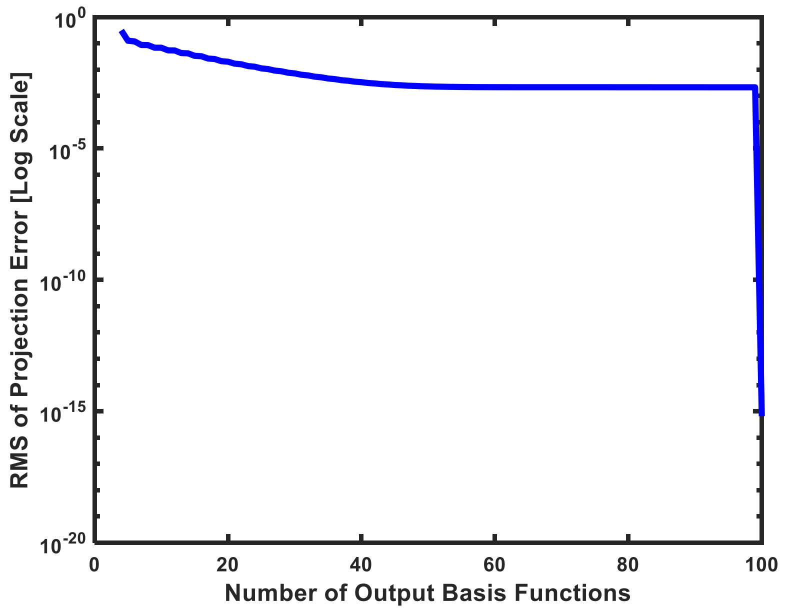


Figure 10. RMS of projection error of y_1^* in Log Scale vs. the number of output basis functions *N* without model error.

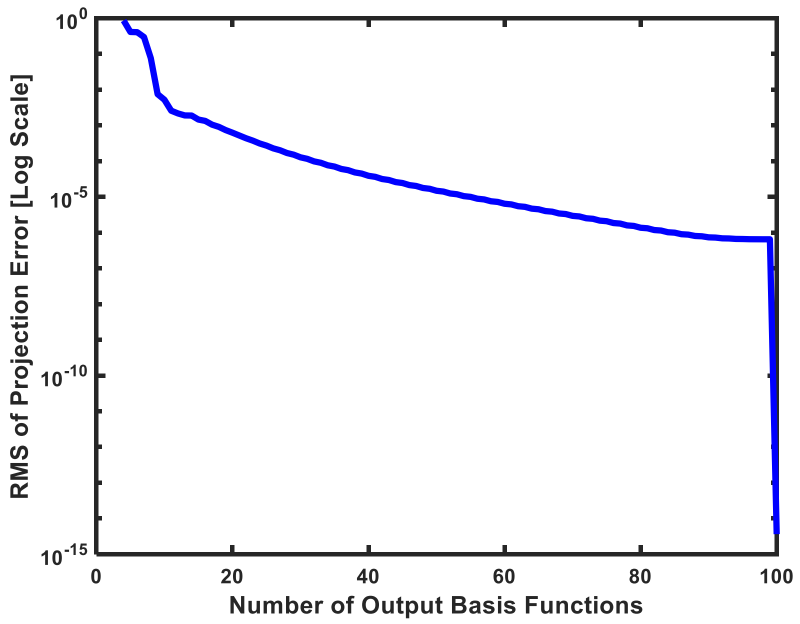


Figure 11. RMS of projection error of y_2^* in Log Scale vs. the number of output basis functions *N* without model error.

3.8.2 Tracking Performances Using Matched Basis Function ILC with Model Error

Now think about the influence of model error. Consider the true world system is the $G(s)$ above, but the ILC design is based on the model whose continuous transfer function is $\hat{G}(s) = \left(\frac{\hat{a}}{s+\hat{a}}\right)\left(\frac{\hat{\omega}^2}{s^2+2\hat{\xi}\hat{\omega}s+\hat{\omega}^2}\right)$ where $\hat{a} = 15$, $\hat{\xi} = 0.8$, and $\hat{\omega} = 50$. Again, the number of basis functions chosen is $N = 10$. Figures 12 to 14 give the results corresponding to Figures 7 to 9. Note that the horizontal scale for Figures 12 and 14 go to iteration 100 to expand the view of the decay and growth, while Figure 13 goes to iteration 300 to see that it reaches a final error level. Compare the RMS of the final unaddressed error level to the RMS of the commanded signal y_1^*. The commanded signal is a sine wave of unit amplitude, so its RMS is 0.707, while the accumulated error in the unaddressed part of the space approaches 0.0789, which is more than 10% of 0.707. The error in representing the desired function with 10 basis functions should correspond to the 0.0624 start value on the plot.

Figures 15 to 17 give corresponding figures for y_2^*. Note that in Figure 17, the error in the unaddressed part of the output space actually decreases instead of increases as the learning iterations reduce the error in the addressed part of the space, indicating this phenomenon is problem dependent.

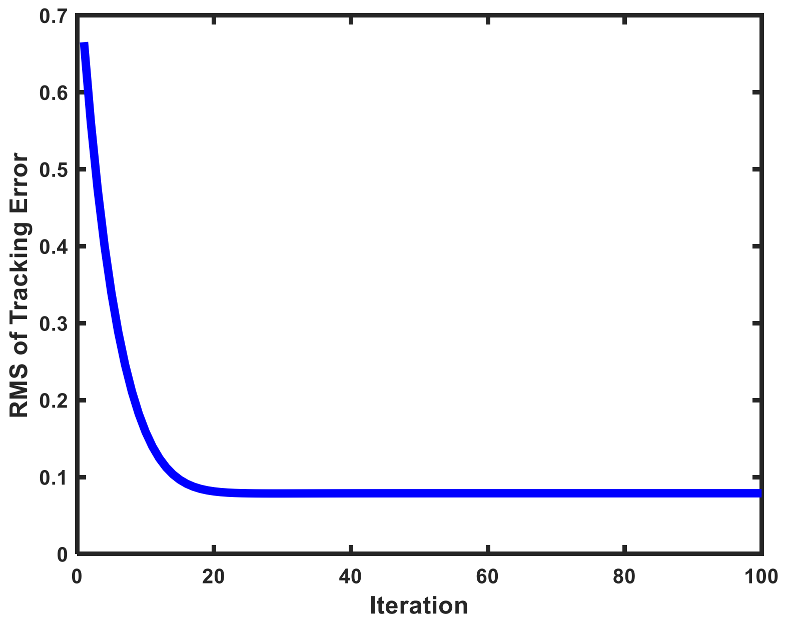


Figure 12. RMS of error when tracking y_1^* vs. iteration in the presence of model error.

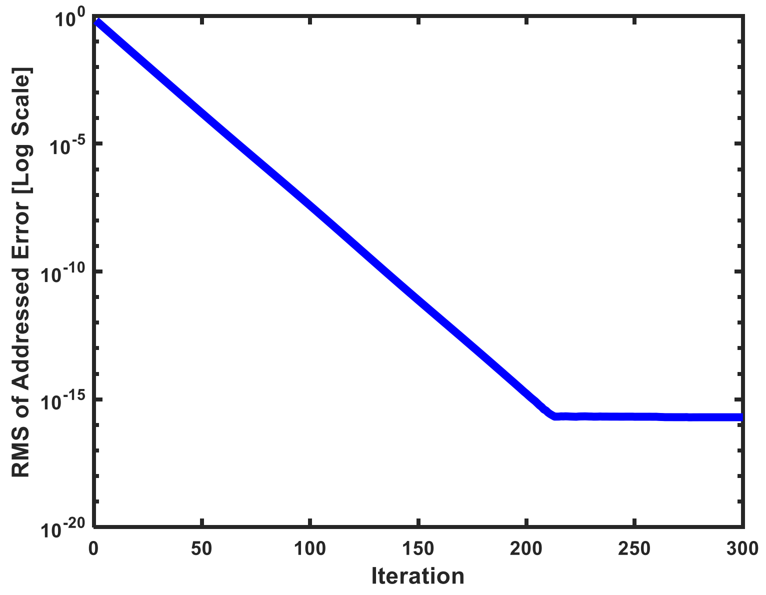


Figure 13. RMS of addressed error vs. iteration in Log Scale associated with Figure 12.

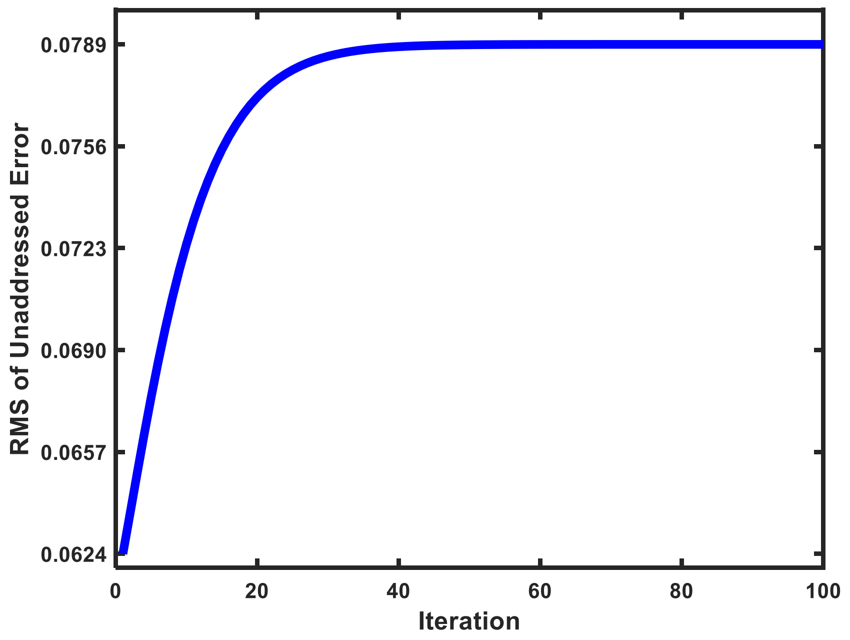


Figure 14. RMS of unaddressed error vs. iteration associated with Figure 12.

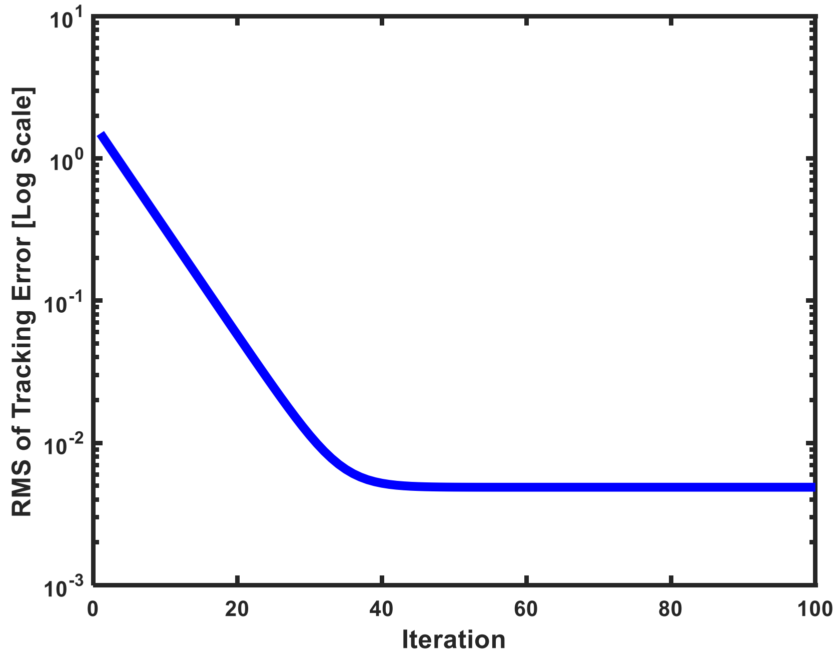


Figure 15. RMS of error in Log Scale when tracking y_2^* vs. iteration in the presence of model error.

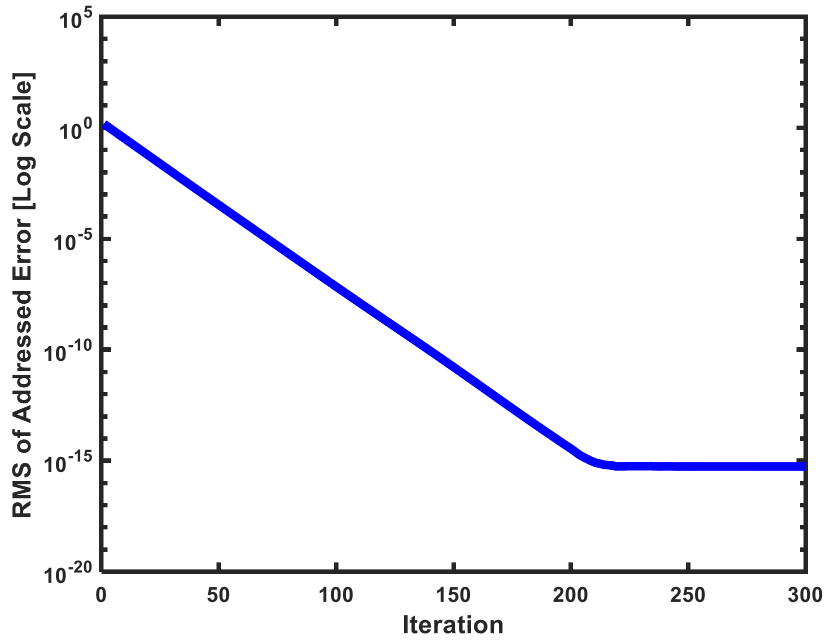


Figure 16. RMS of addressed error vs. iteration in Log Scale associated with Figure 15.

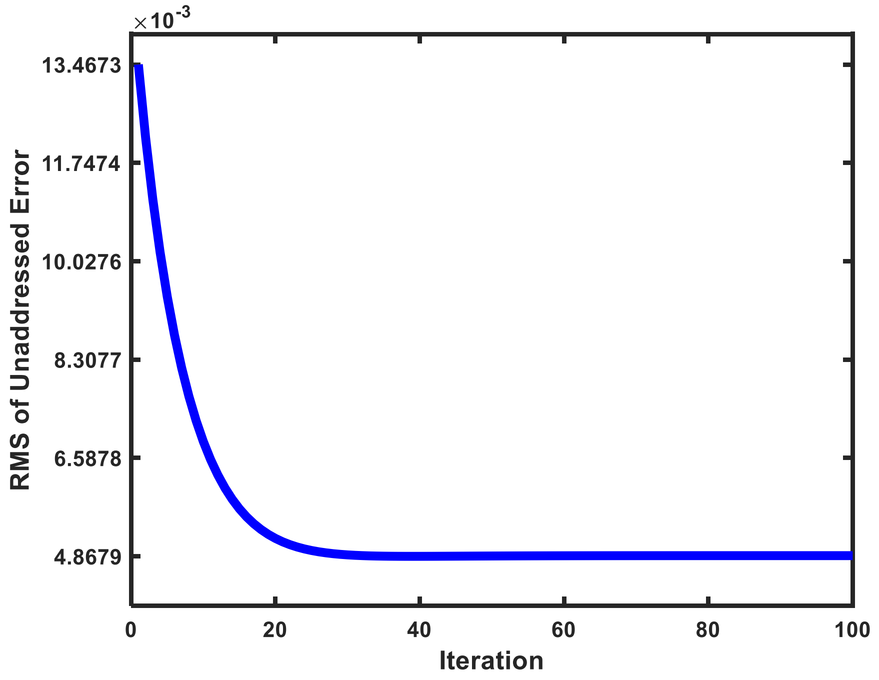


Figure 17. RMS of unaddressed error vs. iteration associated with Figure 15.

Figures 18 and 19 show the effect of the learning gain matrix L_A, making all ϕ_i in L_A the same equal to ϕ and adjusting this value. As mathematically proved above, the final error level reached in the unaddressed error space is independent of the choice of this learning gain, and its only effect is to adjust the rate of convergence, i.e. the number of learning iterations it takes to reach the final convergence level. In ILC, smaller gains are often more robust to model error.

Figures 20 and 21 study the error in representing the desired trajectories, y_1^* and y_2^* respectively, on the basis functions as a function of the number of basis functions *N* in the presence of model error. These curves are compared to the RMS error levels reached after apparent convergence at iteration 300. The difference between these plots represents the error in the unaddressed part of the output space including the error accumulated during the learning process.

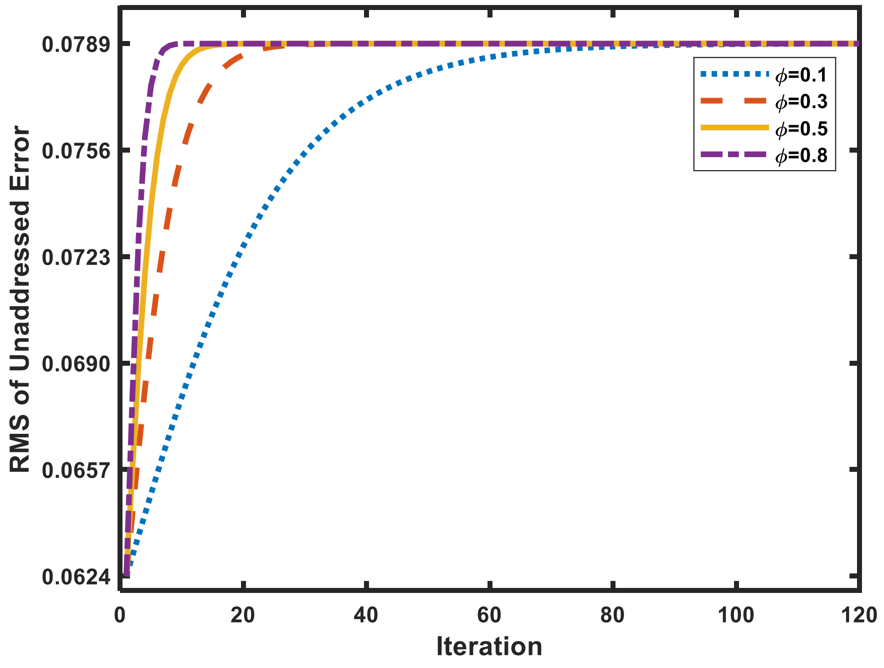


Figure 18. The influence of learning gain on unaddressed error accumulation with *N*=10 when tracking y_1^*.

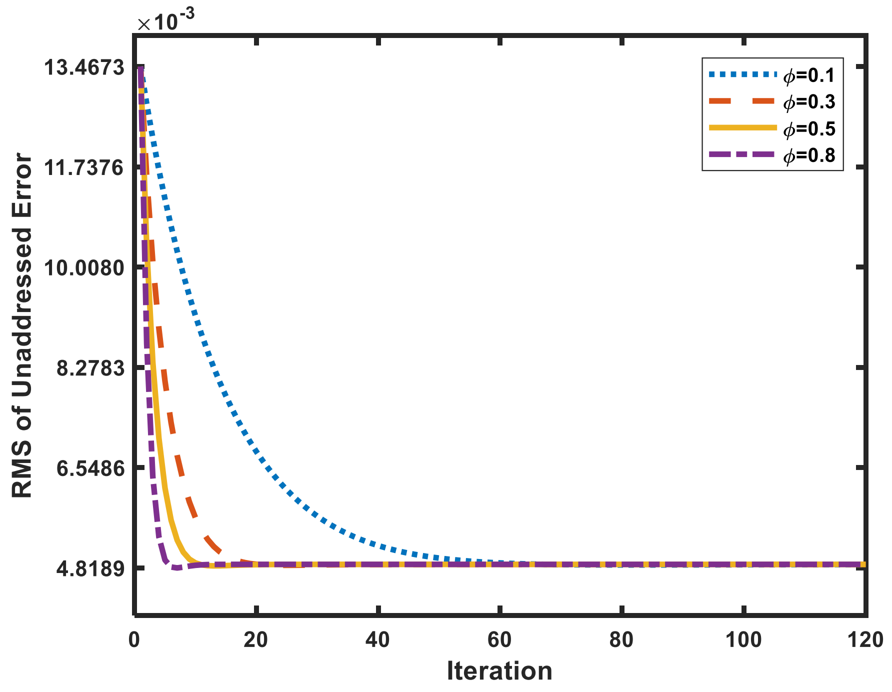


Figure 19. The influence of learning gain on unaddressed error accumulation with N=10 when tracking y_2^*.

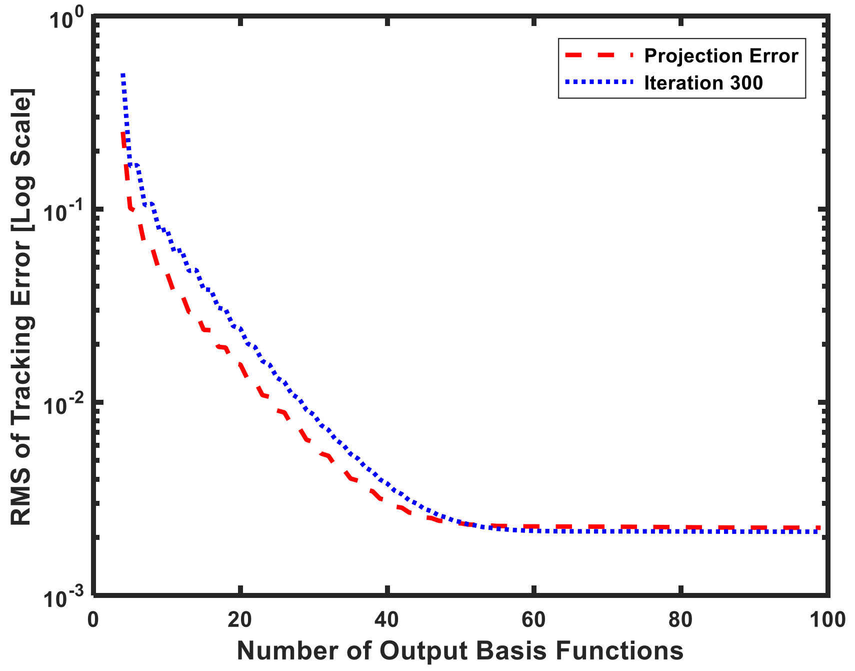


Figure 20. RMS of projection error of desired trajectory and after 300 iterations of learning y_1^* in Log Scale vs. the number of output basis functions including model error.

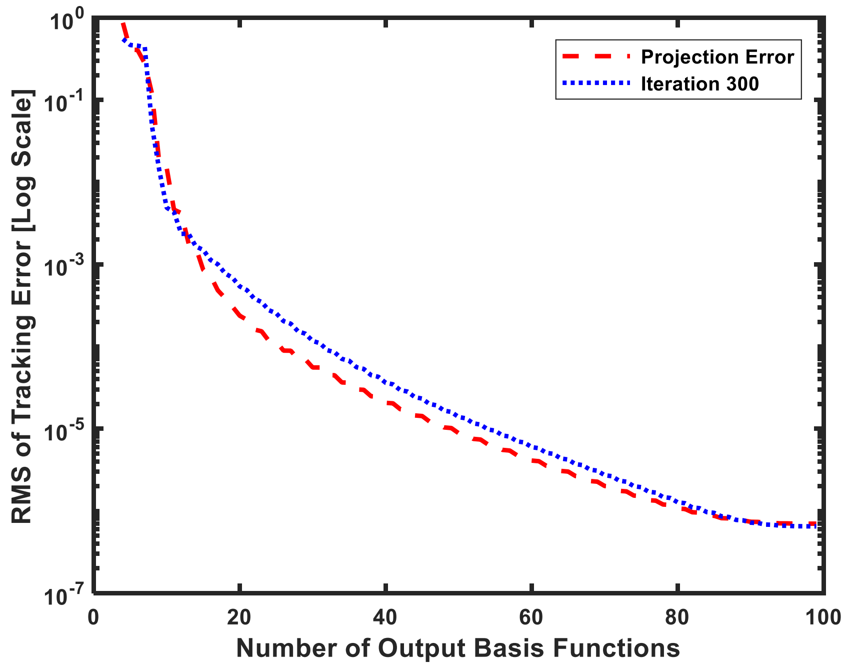


Figure 21. RMS of projection error of desired trajectory and after 300 iterations of learning y_2^* in Log Scale vs. the number of output basis functions including model error.

3.8.3 Tracking Performance Using Matched Basis Function ILC without Model Error Including Transient Basis Functions

This section presents simulations of the ILC law including transient basis functions. Note that y_1^* does not have a “smooth” start up from all zero initial state. Consider the $G(s)$ above without model error, pick $N = 20$, and the learning gain matrix L_A is set to the identity matrix. Figures 22 and 23 give the control history at iteration 500, at which the tracking has reached convergence, without the initial condition basis functions and using the initial condition basis functions respectively. The control input starts from a smaller magnitude at the initial time steps and becomes smoother after applying the transient basis functions, making it less likely to exceed actuator limits. The corresponding trajectory error histories are given in Figures 24 and 25. Note that there is considerable improvement in the tracking error, and the improvement is strongest near the beginning of the trajectory when there should be initial condition effects to address. For example, with the help of transient basis functions, the large tracking error at the beginning has been reduced from about 0.1 to about 0.01. As might have been expected, when transient basis functions are employed on the trajectory y_2^* that has a "smooth" startup and therefore there is no need for these initial condition basis functions, introducing transient basis functions deteriorates the performances, which is consistent with the conclusion in Chapter 2.

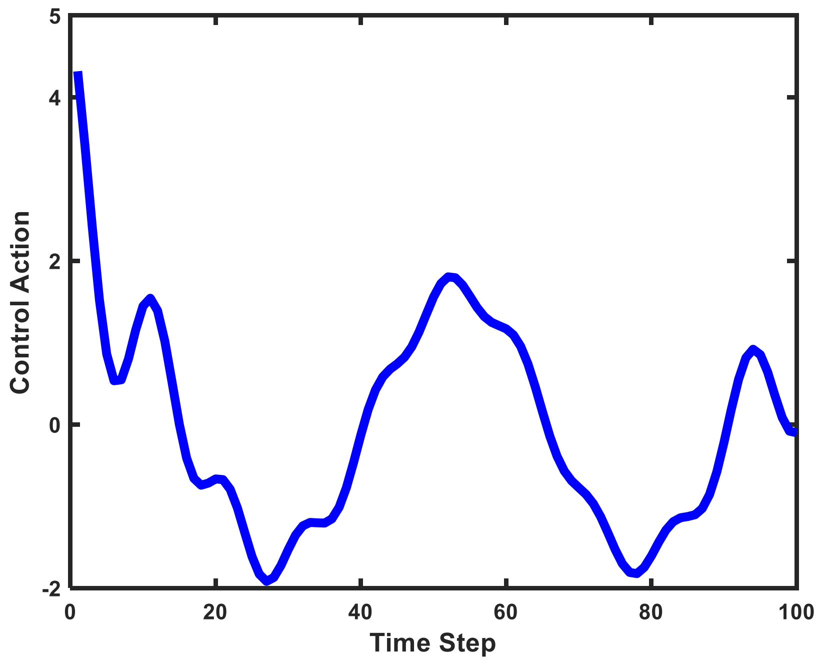


Figure 22. Control action in the 500th iteration without transient basis functions tracking y_1^*.

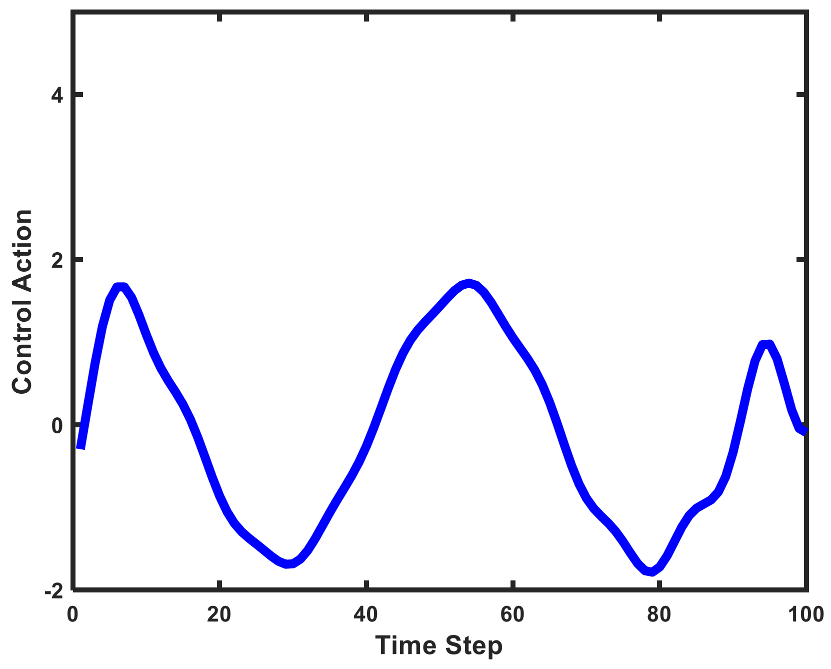


Figure 23. Control action in the 500th iteration with transient basis functions tracking y_1^*.

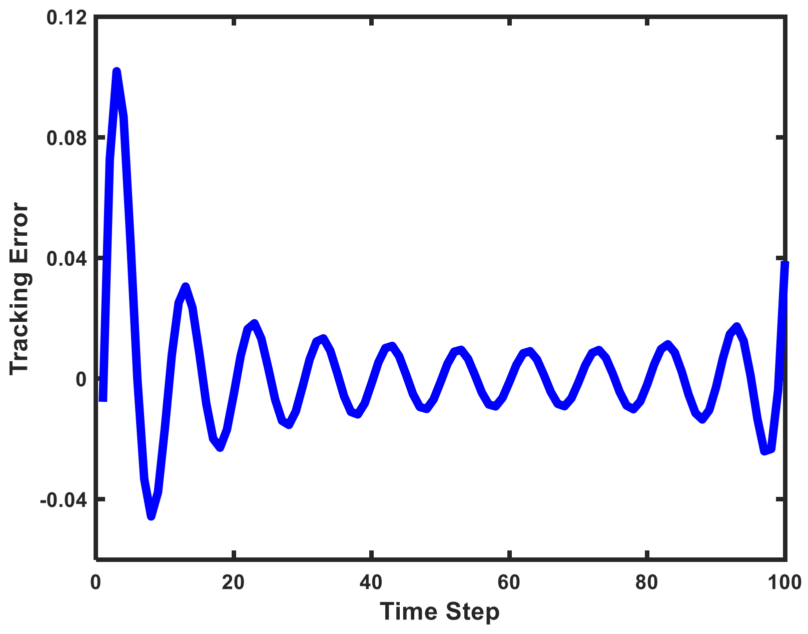


Figure 24. Tracking error in the 500th iteration without transient basis functions.

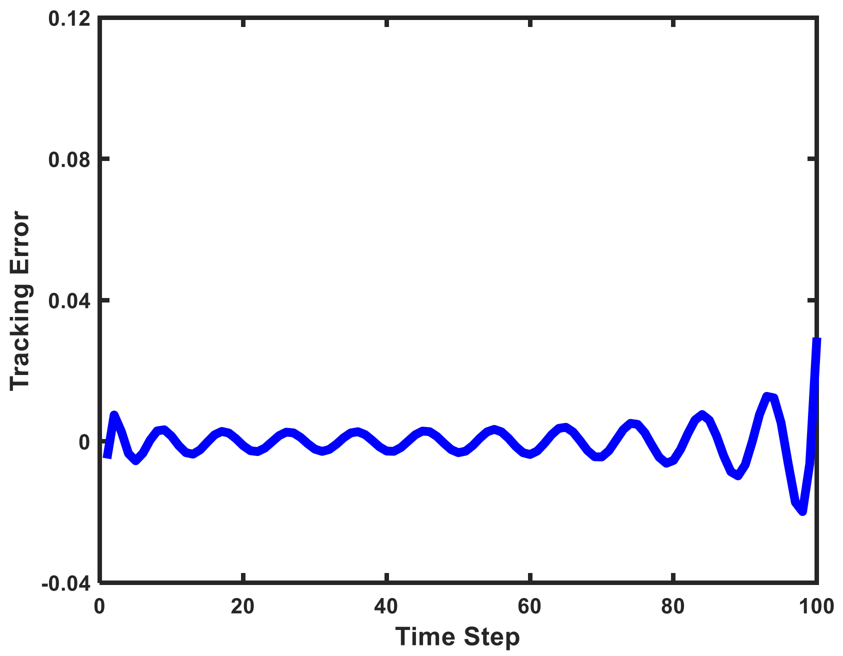


Figure 25. Tracking error in the 500th iteration with transient basis functions.

3.9 Conclusions

Iterative Learning Control normally starts with the concept of converging to zero tracking error. But before it becomes practical, it is likely to require a zero-phase low-pass filter to cut off the learning at high frequencies in order to have robustness to parasitic poles or residual modes. This can also be necessary because the ILC inverse problem for many digital systems is unstable. Basis function ILC from the beginning is willing to sacrifice zero tracking error. Using the frequency response related singular vector basis functions in this work, the choice of basis functions forms another way to make the frequency cutoff to address the unstable inverse issue and the issue of stability robustness to residual dynamics. It offers computational advantages as well.

What can be a serious issue in basis function ILC is identified: the accumulation of error in the part of the output error space that is not being addressed, i.e. not in the span of the chosen output basis functions. In order to try to eliminate the accumulation, it is suggested that matched basis functions should be used, and to use as good a model in generating the matching functions as possible. With a perfect model used to define the matching functions, the error accumulation is eliminated.

For the finite time desired trajectories in ILC, there does not seem to be simple alternatives to the singular vector basis function choice developed here. The main alternative is to pick a set of orthogonal input basis functions, run experiments to determine the outputs produced by each. Then one would need to use an orthonormalization process to produce the output basis functions.

The matched basis functions used here come from the input and output singular vectors of the matrix of Markov parameters representing the input-output relationship of the system. The approach automatically spans the input-output spaces with orthogonal basis function. Also, each input basis function is uniquely paired to one of the automatically orthonormal output basis functions. Note that the singular vector basis functions can be generated from a system model, but alternatively, one can do experimental tests with a long rich input, and then determine the system Markov parameter by the OKID algorithm in Juang et al. (1993). This allows one to design the ILC without having to create a model, avoiding issues involved in system identification such as determining the system order.

Chapter 4 Generalized One Step Ahead Control Made Practical by New Stable Inverses

4.1 Introduction

One step ahead control is a digital control approach that aims to produce zero error in following any chosen trajectory (Goodwin et al., 1980 and Goodwin et al., 1984). Starting from a scalar difference equation, consider a third order difference equation at time step k when one is picking the control input $u(k)$. Past data include outputs $y(k\text{-}1)$, $y(k\text{-}2)$ and inputs $u(k\text{-}1)$, $u(k\text{-}2)$ (assuming a one-step time delay through the system). As soon as the measurement $y(k)$ arrives, the equation contains two variables, $u(k)$ to be chosen, and $y(k+1)$ which one sets to the desired output $y^*(k+1)$ for the next time step. Then solve for the $u(k)$ to produce the desired output.

This control law has three very desirable properties: it promises zero tracking error to within the accuracy of the model, and this is produced by a feedback control law, and one can ask for any output desired at the next time step. Yet this is too good to be true. The obvious difficulty is that it asks for zero error at the end of the current time step to any output one named without regard to actuator limitations. The second difficulty is more hidden. The algorithm is inverting the system, and discrete time models of continuous time differential equations fed by a zero-order hold are usually unstable for any system with pole excess of 3 or more, i.e. a third order or higher system with no zero in continuous time (Åström et al., 1980). So the control law may produce zero error at every step, but it does so at the expense of using a control action that is the solution of an unstable equation. A controller that produces unstable control action is of no use in the world.

A stable inverse theory has been developed in the literature, see for example Devasia et al., (1996) and Zou et al., (1999). The approach gives a stable inverse for a given time history, but it requires a pre-actuation period in advance of the zero tracking error trajectory. Researchers in the Control System Research Group at Columbia University have recently developed a stable inverse theory consisting of four new stable inversion methods. The four stable inverses are termed Longman JiLLL FI, JiLLL NI, JiLLL FS, and JiLLL NS respectively, where JiLLL refers to the researchers involved in the theory development (LeVoci et al., 2004, Li et al., 2010, Li et al., 2016, Longman et al., 2017, Ji et al., 2017), “F” stands for “factorization”, “N” stands for “without factorization”, “I” stands for “initial deletion”, and “S” stands for “skip steps” (Ji et al., 2019). The two “initial deletion”-based versions, JiLLL FI and JILLL NI, have been applied to solve the unstable inverse issue in ILC, where zero tracking error is achieved for all the steps after the several deleted initial time steps. The two “skip steps”-based versions, JiLLL NS and JiLLL FS, are chosen to obtain stable inverses for one-step ahead control generalization because of their uniformity for the introduced time steps. The fundamental part of both JiLLL NS and JiLLL FS are increasing the sampling rate: to create a stable inverse over a given time interval of a discrete time system with one zero outside the unit circle, double the original sample rate but ask for zero error restricted to the original sample times. The two holds between each addressed time step can be considered as a kind of generalized hold. JiLLL NS simply asks for the minimum norm solution for the control action over all time steps, produced by a pseudo inverse. JiLLL FS, on the other hand, after introducing extra time steps as before, uses a factorized input-output relationship with two matrices afterwards, one matrix for the zero(s) outside the unit circle multiplies the other matrix for all poles and zeros inside the unit circle. The latter matrix is stably invertible. Then delete the introduced unaddressed time steps in the output making an underspecified set of equations, and find the minimum Euclidean norm control solution. If there are 2 zeros outside the unit circle, then triple the original sample rate, introducing two extra time steps between two original time steps.

The pre-actuation period of the previous stable inverse solution requires a time period that is related to given system dynamics, and could only promise arbitrarily close to zero error of the rest of the trajectory. The JiLLL FS and JiLLL NS approaches, both make a batch computation, replace this pre-actuation with the change in sample rate, and guarantees zero tracking error at the sample times of the original sample rate. The behaviors at the unaddressed times steps in JiLLL FS and JiLLL NS are also studied and presented, and it leads to the conclusion that JiLLL NS gives more satisfying results at unaddressed time steps than JiLLL FS.

4.2 One Step Ahead Control

Various concepts will be illustrated using a third order model. Given a third order differential equation fed by a zero-order hold, it can be converted to a third order difference equation that has the same solution at the sample times. The time delay from a change in the zero-order hold input to the first time step that one observes a change in the difference equation solution is generically one time step. The solution to the differential equation does not move instantaneously when the zero-order hold input moves to the next hold value, but the solution should be changed after the hold value has been acting for one time step. This means that the most up to date time step on the output side of the difference equation should be one step ahead of the most up to date term on the input side of the equation. Thus, the difference equation model generically has n – 1 terms on the input side, when the output size has n terms, and discretization has introduced enough zeros to make this happen. Consider a general third order difference equation

$$y(k+1)+a_1y(k)+a_2y(k-1)+a_3y(k-2)=b_1u(k)+b_2u(k-1)+b_3u(k-2) \tag{4.1}$$

At time step k when one is picking $u(k)$, the inputs $u(k-1)$ and $u(k-2)$ are known, and $y(k-1)$ and $y(k-2)$ are previous measurements. When the feedback measurement $y(k)$ arrives, one can specify the desired output at the next time step $y^*(k+1)$ and solve for the control input $u(k)$ to produce this output

$$u(k) = [y^*(k+1) + a_1 y(k) + a_2 y(k-1) + a_3 y(k-2) - b_2 u(k-1) - b_3 u(k-2)]/b_1 \tag{4.2}$$

As mentioned above, this control law is a feedback control that promises zero tracking error to within the accuracy of the model, and it allows one to ask for anything one wants right at the next time step. One difficulty is that actuator limits prevent one from making large moves in just one time step. The other issue relates to the nature of the right-hand side of this equation for a majority of physical systems.

Equations such as Eq. (4.1) can be solved recursively, given the initial conditions or the currently known values at time k, $y(k), y(k-1), y(k-2)$ and $u(k), u(k-1), u(k-2)$ are known after having computed the desired control action $u(k)$. Then solves for $y(k+1)$. Then update k to $k + 1$, and repeat. Alternatively, one can look for the analytical solution as one would when solving a differential equation. In this case, substitute the desired output sequence of $y^*(k)$ values into the left-hand side of Eq. (4.1) to create

$$f(k) = y^*(k+1) + a_1 y^*(k) + a_2 y^*(k-1) + a_3 y^*(k-2) \tag{4.3}$$

which then represents a forcing function for the following difference equation with associated characteristic polynomial (for the system zeros) that must be solved to find the needed input history

$$b_1 u(k) + b_2 u(k-1) + b_3 u(k-2) = f(k) \tag{4.4}$$

$$b_1 z^2 + b_2 z + b_3 = 0$$

The solution contains a particular solution $u_P(k)$ associated with the forcing function $f(k)$, plus two linearly independent solutions of the homogeneous solution multiplied by arbitrary constants that are determined by initial conditions

$$u(k) = u_P(k) + C_1 z_1^k + C_2 z_2^k \tag{4.5}$$

The zero locations are a function of the time interval between samples *T*. Åström et al. (1980) gives an analytical derivation of the locations of the system zeros, i.e. the roots z_1, z_2 of the characteristic polynomial in Eq. (4.4). The number of zeros introduced depends on the pole excess, i.e. the number of poles minus the number of zeros in the original Laplace transfer function before converting to discrete time. When T tends to zero, the locations of the zeros introduced outside the unit circle are given as follows: pole excess of 2, zero at -1; pole excess of 3, zero at -3.732; pole excess of 4, zeros at -1, -9.899; pole excess of 5, zeros at -2.322 and -23.20; pole excess of 6, zeros at -1, -4.542, and -51.22; pole excess of 7, zeros at -1.868, -8.160, -109.3. And for every zero indicated outside the unit circle, there is a zero inside the unit circle at the reciprocal location. When the sample time interval *T* gets very long, so that the system transients have already decayed to near zero before the next time step arrive, then the zeros approach the origin. Usually, any sample rate that is able to keep the zeros inside the unit circle have an unreasonably slow Nyquist frequency and are not useful because of the aliasing produced.

For the 3rd order difference equation considered here, with 2 zeros introduced, there is a solution that approaches a constant times $(3.732)^k$. A more extreme case is a 7th order system which will have a solution of the homogeneous equation for fast sample times that approaches a constant times $(109.7)^k$ where k is the time step. If you happen to be sampling at 1000Hz, this is a violent instability in the inverse problem, making a solution of the inverse problem that grows by two orders of magnitude every time step, and by 2000 orders of magnitude at the end of one second. The stable inverse approaches find ways to circumvent such instabilities, and they also handle instability of the inverse problem for non-minimum phase zeros in the original differential equation.

4.3 The Unique Inverse Solution

Consider the difference equation of interest expressed in state space form

$$x(k+1) = Ax(k) + Bu(k); \qquad y(k) = Cx(k) \tag{4.6}$$

and consider a *p* time step input sequence and corresponding output sequence written as column vectors that take into account the one time step delay through the system

$$\underline{u} = [u(0), u(1), \dots, u(p-1)]^T, \qquad \underline{y} = [y(1), y(2), \dots, y(p)]^T \tag{4.7}$$

Then packaging all *p* time steps in matrix form produces

$$\underline{y} = P\underline{u} + \overline{A}x(0), \qquad P = \begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \dots & CB \end{bmatrix} \tag{4.8}$$

and $\overline{A}$ is an observability matrix. If one asks for the input history $\underline{u}$ to produce a desired output history $\underline{y}^*$ given at every time step, the unique solution is

$$\underline{u} = P^{-1}[\underline{y}^* - \overline{A}x(0)] \tag{4.9}$$

Following the presentation in Longman et al. (2017), some properties of the matrix of Markov parameters *P* are given. Start the illustration with a third order system

$$Y(s) = G(s)U(s) = \left(\frac{a}{s+a}\right)\left(\frac{\omega_0^2}{s^2+2\xi\omega_0 s+\omega_0^2}\right)U(s) \tag{4.10}$$

with $a = 8.8$, $\omega_0 = 37$, and $\xi = 0.5$. This is converted to discrete time with sample rate of 100Hz, Nyquist frequency of 50Hz, and a 50 by 50 P matrix is constructed. There is one zero outside the unit circle at -3.3104, and hence the inverse of matrix P contains this instability. One wants to present the signatures of instability as seen in the P matrix. The conclusions and figures come from Longman et al. (2017), although similar figures appear in Li et al. (2010) and Li et al. (2016).

Matrix P can be related to frequency response as presented in Chen et al. (2003). Starting with equation $\underline{y} = P\underline{u}$, multiply from the left by a matrix to take the discrete Fourier transform and insert this matrix and its inverse between P and $\underline{u}$. Considering a long enough trajectory for steady state to dominate, this gives the system frequency response. Compare this complex valued relationship to the singular value decomposition $P = USV^T$, after converting to real values, and ordering the singular values with frequency as dictated by the DFT of the singular vectors. The matrix S aims to match the steady state magnitude frequency response at the frequencies one can see in 50 time steps of data. The matrices U and V are the left and right singular vectors and they contain these frequencies, but with some end effects because P produces a solution starting from zero initial conditions. The phase change is seen in the difference between corresponding input and output singular vectors.

Figure 26 shows all singular values of P, and also plots the steady state magnitude frequency response. These plots are nearly on top of each other. Note that there is a singular value that is very small near the bottom right corner of the plot, around 10^{-19}, while the previous singular value is 5.0172×10^{-4} on the frequency response curve. Figure 27 shows that this anomalous singular value is related to the zero location. The figure plots the smallest singular value as a function of the matrix size p, ending at p = 50. Also plotted is the reciprocal of the magnitude of the zero location taken to power p, and one observes that they have the same slope. Matrix P gets increasingly more ill conditioned as the matrix size increases. Note that it is sufficiently ill conditioned after about 30 by 30, that Matlab is no longer able to compute the true singular value which should continue on the slope.

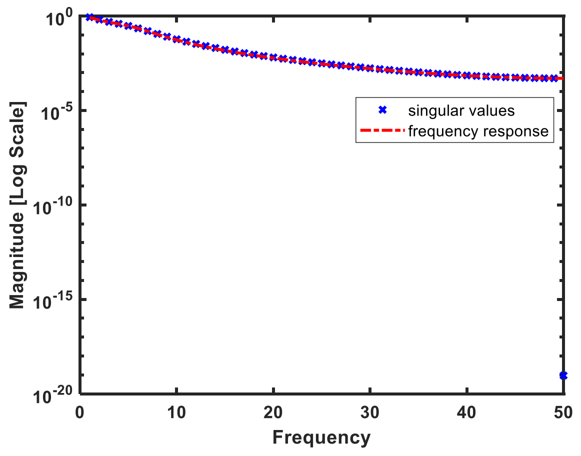


Figure 26. The singular values of P and the steady state magnitude frequency response.

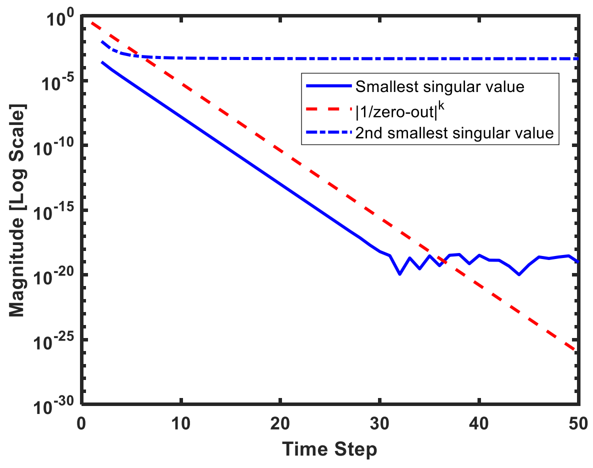


Figure 27. The smallest singular value and the next to smallest as a function of the dimension of P.

Figure 28 plots the magnitude of the last output singular vector component magnitudes (the components alternate in sign) in U on a Log scale, versus component number, and one observes that they decrease with time step k according to $1/(3.3104)^k$. Figure 29 gives the same plot but for the input singular vector components in V. This time the components increase with time step according to $(3.3104)^k$. This implies, if one wants to cancel an output error vector like that in Figure 28, one must apply an input control vector corresponding to Figure 29. Thus, to fix an error in the first few time steps one must apply a control that is growing exponentially with time.

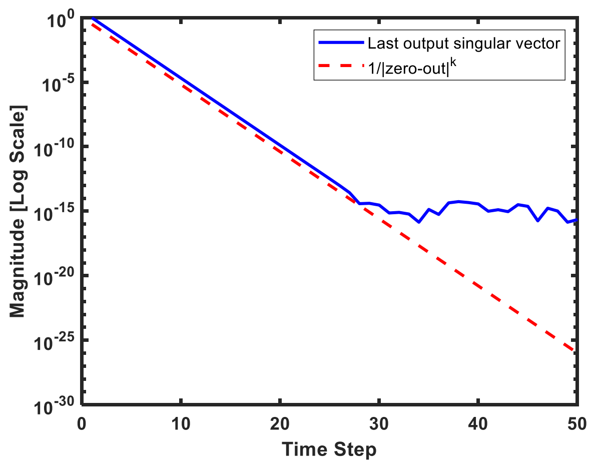


Figure 28. Magnitude of last output singular vector components at time step *k*, and slope of reciprocal of magnitude of zero location to power of time step.

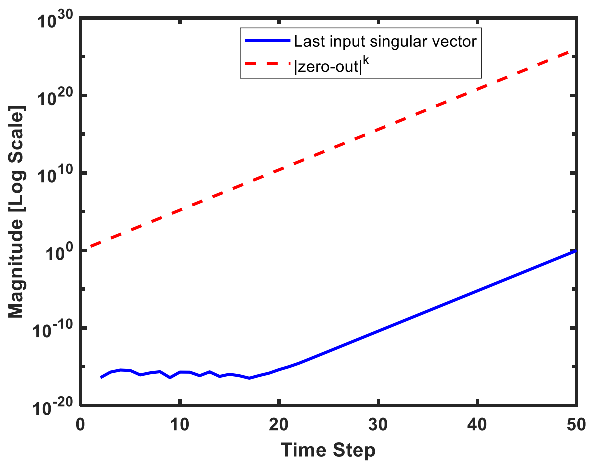


Figure 29. Figure 28 but for the last input singular vector, and the magnitude of the zero location to the power of the time step.

The conclusion from Longman et al. (2017) is that, if one can produce a “system inverse” with the property that it produces zero error at the time steps addressed, and has no singular values and corresponding singular vectors that exhibit the above behavior, then one has generated a zero error stable inverse to the unstable inverse problem.

4.4 Stable Inverse Theory Longman JiLLL

In the following two subsections, the mathematics of in-house developed stable inverse theory Longman JiLLL FS and JiLLL NS are presented respectively. Both Longman JiLLL FS and NS have the fundamental part of first increasing the sampling rate, introducing *m* time steps between the original two time steps of the trajectory when there are *m* zeros located outside the unit circle. Take the illustration in Figure 30 as an example. For a system of pole excess of 3 as presented before, when there is one zero introduced located outside the unit circle, one time step (marked as the red cross) is introduced between the original two time steps (marked as blue circles). Zero tracking error is only aimed at the original time steps, but not at the newly introduced time steps. Hence, the introduced time steps are skipped and unaddressed, while the original time steps are addressed.

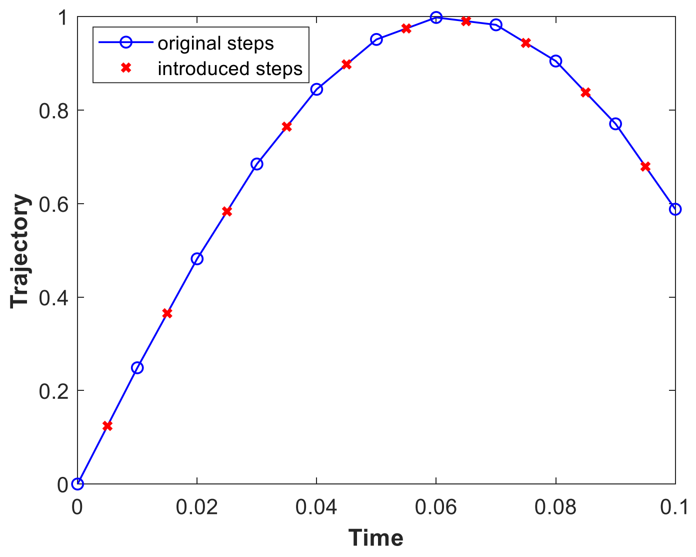


Figure 30. Addressed and unaddressed time steps in a trajectory when 1 zero introduced outside the unit circle.

4.4.1 Longman JiLLL FS

First, develop the stable inverse for a general third-order difference equation system with one zero outside the unit circle, which can be either a sampling zero or an intrinsic zero. The generalization to higher order systems with an original pole excess of 3 in continuous time is immediate. Then the method of producing a stable inverse for general n^{th} order systems with m non-minimum phase zeros is presented. In each case, the first objective after increasing the sampling rate is to factorize matrix P into the form $P = Z_O P_I$ where P_I is the P matrix for the system including all zeros and poles inside the unit circle. By assuming an asymptotically stable system, the poles are always inside. Then Z_O is a matrix for all zeros outside the unit circle. Consider the third order system and its state space model

$$y(k+3) + a_1 y(k+2) + a_2 y(k+1) + a_3 y(k) = b_1 u(k+2) + b_2 u(k+1) + b_3 u(k)$$

(4.11)

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad C = [b_3 \quad b_2 \quad b_1]$$

Use q as a one step forward time step shift operator, and then rewrite Eq. (4.11) as

$$[q^3 + a_1 q^2 + a_2 q + a_3]\{y(k)\} = b_1[(q - z_O)(q - z_I)]\{u(k)\}$$
$$= b_1[q^2 - (z_O + z_I)q + z_O z_I]\{u(k)\}$$

(4.12)

where z_O is the zero outside the unit circle and z_I is the zero inside. The system model without the zero outside the unit circle is

$$[q^3 + a_1 q^2 + a_2 q + a_3]\{y_I(k)\} = b_1(-z_O)(q - z_I)\{u(k)\} \tag{4.13}$$

It can be put into state variable form by defining $\bar{y}_I(k)$ to satisfy the following difference equation and then applying superposition to get the output $y_I(k)$ from output $\bar{y}_I(k)$

$$[q^3 + a_1 q^2 + a_2 q + a_3]\{\bar{y}_I(k)\} = u(k) \tag{4.14}$$

$$x_I(k) = \begin{bmatrix} \bar{y}_I(k) \\ \bar{y}_I(k+1) \\ \bar{y}_I(k+2) \end{bmatrix}, \quad A_I = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix}, \quad B_I = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad C_I = [b_3 \quad b_1(-z_O) \quad 0]$$

$$y_I(k) = b_1(-z_O)\bar{y}_I(k+1) + b_3 \bar{y}_I(k)$$

$$P_I = \begin{bmatrix} C_I A_I B_I & 0 & \cdots & 0 \\ C_I A_I^2 B_I & C_I A_I B_I & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ C_I A_I^p B_I & \cdots & C_I A_I^2 B_I & C_I A_I B_I \end{bmatrix}$$

The Toeplitz matrix P_I for inside poles and zeros makes use of the fact that $C_I B_I$ is zero because there is a two-time step delay from input to output in Eq. (4.11). The actual output $y(k)$ is related to $y_I(k)$ by superposition according to

$$y(k) = y_I(k) - \frac{1}{r_O} y_I(k+1) \tag{4.15}$$

which produces Z_O as

$$\underline{y} = Z_O \underline{y_I},\ Z_O = \begin{bmatrix} -\frac{1}{z_O} & & & 0 \\ 1 & -\frac{1}{z_O} & & \\ & \ddots & \ddots & \\ 0 & & 1 & -\frac{1}{z_O} \end{bmatrix} \tag{4.16}$$

As described before, the Longman Stable Inverse JiLLL FS asks that one start with a sample rate, asking for zero error at each original sample time called the addressed time steps, then for one zero outside the unit circle, double the sample rate, allowing the use of two zero-order holds between each of the original sample times. The newly introduced time steps are termed unaddressed steps. Delete all unaddressed rows from matrix Z_O, and denote the result Z_{Oa}. Then the stable inverse is $P_I^{-1} Z_{Oa}^{\dagger}$, and the control action becomes

$$\underline{u} = P_I^{-1} Z_{Oa}^{\dagger} [\underline{y_a^*} - \bar{A}_a x(0)] \tag{4.17}$$

Matrix P_I of course has a well-behaved inverse, and $Z_{Oa}^{\dagger}$ is the Moore-Penrose pseudo inverse of Z_{Oa}. In the rest part of this chapter, this stable inverse, JiLLL FS, is denoted as $P^{\S} = P_I^{-1} Z_{Oa}^{\dagger}$.

These results generalize to nth order difference equation system with m zeros outside the unit circle. One must increase the sample rate to include m zero-order holds between each addressed time step. The P_I and the Z_O matrices are given as

$$[q^n + a_1 q^{n-1} + \cdots + a_n]\{y(k)\} = b_1 \prod_{i=1}^{m} (q - z_{O_i}) \prod_{j=1}^{n-1-m} (q - z_{I_j})\{u(k)\}$$

$$[q^n + a_1 q^{n-1} + \cdots + a_n]\{y_I(k)\} = b_1 \prod_{i=1}^{m} (-z_{O_i}) \prod_{j=1}^{n-1-m} (q - z_{I_j})\{u(k)\}$$

$$= [c_{m+1} q^{n-1-m} + \cdots + c_{n-1} q + b_n]\{u(k)\}$$

$$A_I = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ \vdots & 0 & \ddots & \vdots \\ 0 & \vdots & \cdots & 1 \\ -a_n & -a_{n-1} & \cdots & -a_1 \end{bmatrix},\ B_I = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},\ C_I = [c_n \quad c_{n-1} \quad \cdots \quad c_{m+1} \quad 0 \quad \cdots \quad 0]$$

$$P_I = \begin{bmatrix} C_I A_I^m B_I & 0 & \cdots & 0 \\ C_I A_I^{m+1} B_I & C_I A_I^m B_I & \cdots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ C_I A_I^{p+m-1} B_I & \cdots & C_I A_I^{m+1} B_I & C_I A_I^m B_I \end{bmatrix} \tag{4.18}$$

$$y(k) = \prod_{i=1}^{m} (-\frac{1}{z_{O_i}} z + 1)\{y_I(k)\}$$

$$Z_O = \prod_{i=1}^{m} Z_{O_i} \tag{4.19}$$

The last equation gives the Z_O as the product of a Z_O of the form of Eq. (4.16) for each zero outside. An alternative and perhaps simpler way to calculate this matrix is to find P_I and then take its inverse to produce $Z_O = PP_I^{-1}$. Then one eliminates all rows of Z_O except those associated with the original sample rate, keeping every mth row to form Z_{Oa}. The stable inverse is again $P^{\S} = P_I^{-1} Z_{Oa}^{\dagger}$.

To complete the set of Stable Inverse Theory, Longman JiLLL FI starts with all rows of P at the original sample rate and does not introduce more sample times by increasing the sample rate. After factorizing to form $P = Z_O P_I$, delete an initial row(s) of Z_O equal to the number of zeros outside the unit circle, which are then the unaddressed time steps, used to define Z_{Oa}, for the addressed time steps. Then JiLLL FI is produced using this matrix in $P^* = P_I^{-1} Z_{Oa}^{\dagger}$. However, this stable inverse is not considered here because of its non-uniform distribution of unaddressed time steps as time evolves.

4.4.2 Longman JiLLL NS

JiLLL NS is analogous to JiLLL FS. For one zero outside, double the sample rate but ask for zero error only at the original (i.e. addressed) time steps (triple the rate if there are two zeros outside). Originally, P, $\underline{y}^*$, $\bar{A}$ in Eq. (4.8) contain all time steps at the doubled rate. Delete all odd numbered rows to create matrices for only the addressed time steps, and indicate them with subscript "a". Substitute the desired trajectory at addressed steps only, $\underline{y}_a^*$, into the left of Eq. (4.8) and the corresponding P_a and $\bar{A}_a$ containing only the original odd numbered rows into the right, making an underspecified set of equation for the control action $\underline{u}$. Pick the minimum Euclidean norm solution for $\underline{u}$ using the Moore-Penrose pseudo inverse $P_a^{\dagger}$

$$\underline{u} = P_a^{\dagger}[\underline{y}_a^* - \bar{A}_a x(0)] \tag{4.20}$$

This stable inverse is denoted with the symbol $P^{\#} = P_a^{\dagger}$, which indicates start with the full P matrix for all time steps at the fast sample rate, delete all rows except the addressed time steps, for one zero outside the unit circle delete all odd numbered rows. Then take the Moore-Penrose pseudo inverse. This is the analog of JiLLL FS but without the process of factorization.

JiLLL NI, the analog of JiLLL FI, start with P and delete only initial rows, the number equal the number of zeros outside the unit circle, to obtain a different P_a matrix, and then take the Moore-Penrose pseudo inverse. This stable inverse is denoted by $P^* = P_a^{\dagger}$. Again, this version is not applied to the current application because of the lack of uniformity with the unaddressed time steps.

4.5 Recursive *p*-Step Batch Inverse Control

The generalized one-step ahead control law is to update a *p*-step batch of control action recursively every *p* time steps. A model that delivers all *p* time steps of output from the appropriate measured initial conditions is needed, avoiding any need for knowing the usual state variables for state space models. Illustrate the procedures with a third order difference equation model Eq. (4.1). After doubling the sampling rate, start with the usual Linear Model Predictive Control (LMPC) formulation, then look forward 4 time steps and backward 4 time steps, making a batch of size *p*=4 time steps at fast sampling rate. Define forward input vectors and output vectors, and past input and output vectors at a time step *k*

$$\underline{u}_F(k) = \begin{bmatrix} u(k) \\ u(k+1) \\ u(k+2) \\ u(k+3) \end{bmatrix}, \ \underline{y}_F(k) = \begin{bmatrix} y(k+1) \\ y(k+2) \\ y(k+3) \\ y(k+4) \end{bmatrix}, \ \underline{u}_P(k) = \begin{bmatrix} u(k-4) \\ u(k-3) \\ u(k-2) \\ u(k-1) \end{bmatrix}, \ \underline{y}_P(k) = \begin{bmatrix} y(k-3) \\ y(k-2) \\ y(k-1) \\ y(k) \end{bmatrix} \tag{4.21}$$

Write Eq. (4.1) as it appears for *k* + 1, then increase the time argument to *k* + 2, then to *k* +3 and *k* + 4, and package the results in terms of these matrices, to produce

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ a_1 & 1 & 0 & 0 \\ a_2 & a_1 & 1 & 0 \\ a_3 & a_2 & a_1 & 1 \end{bmatrix} \underline{y}_F(k) = \begin{bmatrix} b_1 & 0 & 0 & 0 \\ b_2 & b_1 & 0 & 0 \\ b_3 & b_2 & b_1 & 0 \\ 0 & b_3 & b_2 & b_1 \end{bmatrix} \underline{u}_F(k) + \begin{bmatrix} 0 & 0 & b_3 & b_2 \\ 0 & 0 & 0 & b_3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underline{u}_P(k) - \begin{bmatrix} 0 & a_3 & a_2 & a_1 \\ 0 & 0 & a_3 & a_2 \\ 0 & 0 & 0 & a_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underline{y}_P(k) \tag{4.22}$$

Multiply through by the inverse of the matrix on the left to create the form

$$\underline{y}_F(k) = P_1\underline{u}_P(k) - P_2\underline{y}_P(k) + W\underline{u}_F(k) \tag{4.23}$$

At time step k, the $\underline{y}_P(k)$ and $\underline{u}_P(k)$ are initial conditions. If they are set to zero, and by comparing to Eq. (4.8), one can observe that matrix W is in fact the Markov parameter matrix P. Now pick the size of the future and the past vectors to equal the number p of time steps used in the stable inverse matrix. This makes a matrix of size p by p. Note that increasing the size of past inputs as well as past outputs only introduces extra zeros, which always leaves one with just the same initial conditions needed for Eq. (4.1). But the uniform dimension p for past and future is convenient mathematically. The formula gives the complete future p time step history based on these initial conditions, all of which have been measured or chosen at time step k.

The generalized one ahead control computes p steps into the future and then applies all p time steps to the system. Hence the computations are made every p time steps, so the argument k can be replaced by pj where j is an integer for the start of each p time step update. Then note that the past at one update time is the future for the previous update time, so that $\underline{u}_P(pj) = \underline{u}_F(p(j-1))$ and $\underline{y}_P(pj) = \underline{y}_F(p(j-1))$. Therefore the subscripts for future and past can be dropped in favor of adjusting the argument, resulting in the needed system model

$$\underline{y}\big(p(j+1)\big) = P_1\underline{u}(pj) - P_2\underline{y}(pj) + P\underline{u}\big(p(j+1)\big) \tag{4.24}$$

4.6 Generalized One Step Ahead Control for Zero Error at Addressed Time Steps

Given an initial sample time interval for which one wants zero tracking error, and given a discrete time system that has one zero outside the unit circle, both Longman JiLLL FS and NS ask that the sample rate to be doubled. The original sample times, which are the addressed time steps, are denoted by subscript a, and the new sample time steps, i.e. the unaddressed steps, are denoted by subscript u. Two zeros outside unit circle will need two new sample times between two addressed time steps. Reorder the equations in Eq. (4.24) to separate all addressed time steps from the unaddressed time steps to produce

$$\begin{bmatrix} \underline{y}_a(p(j+1)) \\ \underline{y}_u(p(j+1)) \end{bmatrix} = \begin{bmatrix} P_{1a} \\ P_{1u} \end{bmatrix} \underline{u}(pj) - \begin{bmatrix} P_{2aa} & P_{2au} \\ P_{2ua} & P_{2uu} \end{bmatrix} \begin{bmatrix} \underline{y}_a(pj) \\ \underline{y}_u(pj) \end{bmatrix} + \begin{bmatrix} P_a \\ P_u \end{bmatrix} \underline{u}(p(j+1)) \tag{4.25}$$

Note that this requires reordering the rows of P_1 and P, but both the rows and columns in P_2 must be reordered. Now write separate equations for addressed and unaddressed outputs

$$\underline{y}_a\big(p(j+1)\big) = P_{1a}\underline{u}(pj) - P_{2aa}\underline{y}_a(pj) - P_{2au}\underline{y}_u(pj) + P_a\underline{u}(p(j+1)) \tag{4.26}$$

$$\underline{y}_u\big(p(j+1)\big) = P_{1u}\underline{u}(pj) - P_{2ua}\underline{y}_a(pj) - P_{2uu}\underline{y}_u(pj) + P_u\underline{u}(p(j+1))$$

Using JiLLL FS, P is written as the first of the following equations, written for m possible zeros outside the unit circle

$$P = \left(\textstyle\prod_{i=1}^{m} Z_{O,i}\right)P_I\ , \quad P_a = \left(\textstyle\prod_{i=1}^{m} Z_{O,i}\right)_a P_I, \quad P^{\S} = P_I^{-1}\left(\textstyle\prod_{i=1}^{m} Z_{O,i}\right)_a^{\dagger} \tag{4.27}$$

To obtain P_I, one deletes the unaddressed rows as in the second equation. Then to create the JiLLL FS stable inverse one uses the third expression that takes the inverse of P_I inverting everything inside the unit circle, and then takes the Moore-Penrose pseudo inverse of the term for all outside zeros after deleting.

The first of Eqs. (4.26) is used to produce the p-step batch stable inverse control action. Set the output to be the desired output $\underline{y}_a^*(p(j+1))$. Then solve for the stable inverse control law gives the following for JiLLL NS

$$\underline{u}(p(j+1)) = P^{\#}[\underline{y_a^*}(p(j+1)) - P_{1a}\underline{u}(pj) + P_{2aa}\underline{y_a}(pj) + P_{2au}\underline{y_u}(pj)] \quad (4.28)$$

For JiLLL FS, replace $P^{\#} = P_a^{\dagger}$ by $P^{\S} = P_I^{-1}\left(\prod_{i=1}^{m} Z_{O,i}\right)_a^{\dagger}$ in Eq. (4.27).

This generalized one-step ahead control, updating a batch of size p at fast sampling rate recursively, can also be viewed as a kind of Model Predictive Control (MPC) that produces zero tracking error at certain time steps, unlike the conventional MPC that optimizes a cost functions between control magnitude and output error without aiming to achieve zero error.

4.7 Stability Criterion for Generalized One Step Ahead Control

It is guaranteed that one obtains zero error at addressed time steps every batch update of the control action using the current initial conditions produced by the previous update. But the control actions also produce changes in the output at the unaddressed time steps. It is possible that these unaddressed time steps exhibit instability. Also, even if there is no instability, it is still interesting to investigate the signals at these time steps. This section develops the full batch to batch update equations, as well as the condition for stability.

A complete update of the closed loop dynamics from one p-step batch period to the next is obtained as follows. Substitute $\underline{u}(p(j+1))$ from Eq. (4.28) into both equations in Eq. (4.26), then package all three equations into a single update equation

$$\begin{bmatrix} \underline{y_a}(p(j+1)) \\ \underline{y_u}(p(j+1)) \\ \underline{u}(p(j+1)) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ P_u P_a^{\dagger} P_{2aa} - P_{2ua} & P_u P_a^{\dagger} P_{2au} - P_{2uu} & P_{1u} - P_u P_a^{\dagger} P_{2aa} \\ P_a^{\dagger} P_{2aa} & P_a^{\dagger} P_{2au} & -P_a^{\dagger} P_{1a} \end{bmatrix} \begin{bmatrix} \underline{y_a}(pj) \\ \underline{y_u}(pj) \\ \underline{u}(pj) \end{bmatrix}$$
$$+ \begin{bmatrix} I \\ P_u P_a^{\dagger} \\ P_a^{\dagger} \end{bmatrix} \underline{y_a^*}(p(j+1)) \quad (4.29)$$

The zeros and the identity matrices in the first row result from recognizing that $P_a P_a^{\dagger} = I$ is the identity matrix. Again, for JiLLL FS, replace $P_a^{\dagger}$ by $P_I^{-1}\left(\prod_{i=1}^{m} Z_{O,i}\right)_a^{\dagger}$, which has the same property.

It is guaranteed to get the output required in every *p*-step batch update, and this is accomplished based on the initial conditions produced by the previous *p*-step interval. The zero rows guarantee some zero eigenvalues of the update matrix, but the other eigenvalues must be determined to know if the output at unaddressed time steps propagate in an asymptotically stable manner, and to know if the control action, although stable within each batch period will not grow from batch to batch. These eigenvalues are functions of the choice of the batch size *p* in the stable inverse and independent of the trajectory to be tracked.

4.8 Addressing Actuator Limitations by Tuning a Quadratic Cost Penalty Each Batch Update

Consider that the actuator limits each component i of $\underline{u}(pj)$ according to

$$|u_i(pj)| \leq u_{max} \quad i = 1, 2, \dots, p \tag{4.30}$$

It is assumed that the desired trajectory is feasible so that once on the desired trajectory there is no issue of actuator saturation. But when the initial conditions, or the conditions produced by the previous period are not on the desired trajectory, actuator saturation can easily occur when zero tracking error at the start of an update is asked for. To address this issue, a quadratic cost is generated to use each *p* time step update

$$J(j) = [\underline{y}_a^*(pj) - \underline{y}_a(pj)]^T \left[\underline{y}_a^*(pj) - \underline{y}_a(pj)\right] + r(j)\underline{u}(pj)^T \underline{u}(pj) \tag{4.31}$$

There will always be a value of weight *r* that can penalize the control action sufficiently to keep all components within the actuator limits. Consider the singular value decomposition of

$$P_a^{\dagger} = U \begin{bmatrix} S \\ 0 \end{bmatrix} V^T \tag{4.32}$$

or, alternatively the singular value decomposition of $P_I^{-1}\left(\prod_{i=1}^{m} Z_{O,i}\right)_a^{\dagger}$ having the same form. Substitute this into Eq. (4.28), pre-multiply by U^T, and define control variable $\underline{\mu}$

$$\underline{\mu}(pj) = U^T \underline{u}(pj) = \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} \underline{u}(pj) = \begin{bmatrix} \underline{\mu}_1(pj) \\ \underline{\mu}_2(pj) \end{bmatrix} = \begin{bmatrix} S \\ 0 \end{bmatrix} V^T [\underline{y}_a^*(pj) - h(pj)]$$

$$h(pj) = P_{1a}\underline{u}(pj) - P_{2aa}\underline{y}_a(pj) - P_{2au}\underline{y}_u(pj) \tag{4.33}$$

It is clear that one should set $\underline{\mu}_2(pj) = 0$ since it has no influence on the addressed time steps. Then

$$\underline{u}(pj) = U\underline{\mu}(pj) = U_1\underline{\mu}_1(pj) \tag{4.34}$$

Note that $\underline{y}_a(pj) = P_a U_1 \underline{\mu}_1(pj) + h(pj)$, and that $\underline{u}(pj)^T \underline{u}(pj) = \underline{\mu}_1(pj)^T \underline{\mu}_1(pj)$. Then the cost function can be rewritten as

$$J(j) = [\underline{y}_a^*(pj) - h(pj) - P_a U_1 \underline{\mu}_1(pj)]^T \left[\underline{y}_a^*(pj) - h(pj) - P_a U_1 \underline{\mu}_1(pj)\right] + r(j)\underline{\mu}_1(pj)^T \underline{\mu}_1(pj) \tag{4.35}$$

Differentiating with respect to $\underline{\mu}_1(pj)$, and setting to zero, and converting back to the original control $\underline{u}(pj) = U_1\underline{\mu}_1(pj)$ produces the control update law

$$\underline{u}(pj) = Q(j)[\underline{y}_a^*(pj) - h(pj)]$$

$$Q(j) = U_1[r(j)I + U_1^T P_a^T P_a U_1]^{-1} U_1^T P_a^T \tag{4.36}$$

The update equations from one p-step batch interval to the next are the same as before with $P_a^\dagger$ replaced by Q, except in the case of the $\underline{y}_a\big(p(j+1)\big)$ equation, which can no longer make use of $P_a P_a^\dagger = I$ or $P_a P_a^\S = I$. The result is

$$\begin{bmatrix} \underline{y}_a(p(j+1)) \\ \underline{y}_u(p(j+1)) \\ \underline{u}(p(j+1)) \end{bmatrix} = \begin{bmatrix} [P_a Q - I]P_{2aa} & [P_a Q - I]P_{2au} & [I - P_a Q]P_{1a} \\ P_u Q P_{2aa} - P_{2ua} & P_u Q P_{2au} - P_{2uu} & P_{1u} - P_u Q P_{2aa} \\ Q P_{2aa} & Q P_{2au} & -Q P_{1a} \end{bmatrix} \begin{bmatrix} \underline{y}_a(pj) \\ \underline{y}_u(pj) \\ \underline{u}(pj) \end{bmatrix} + \begin{bmatrix} P_a Q \\ P_u Q \\ Q \end{bmatrix} \underline{y}_a^*\big(p(j+1)\big) \tag{4.37}$$

and the $Q = Q(j+1)$ based on the value of $r(j+1)$ chosen for the update $j+1$.

To use this approach to address actuator saturation constraints, one can set $r(j+1)$ to zero, and examine each component *i* of the control action to see if the inequality is satisfied for all *p* components

$$\left|(U_1 \underline{\mu}_1(k+1))_i\right| \leq u_{max} \tag{4.38}$$

If it is satisfied, then one has zero error at the addressed time steps and there is no need for using the quadratic cost. Otherwise, it is a one-dimensional monotonic search, increasing the $r(j+1)$ until all *p* time steps satisfy the inequality, which is guaranteed to exist. One expects to be able to decrease $r(j+1)$ as periods *p* progress until the quadratic penalty is no longer needed. And by adjusting the value of $r(j+1)$, one can also decide to keep some distance away from the actuator limit if desired. One can create more sophisticated algorithms: increase r in large increments until going to far, then refine locally. Once one has an r that worked for the last *k*, one can do local adjustments to find the next *r*.

An alternative approach is to use quadratic programming. One simply asks to minimize

$$J(j) = [\underline{y}_a^*(pj) - h(pj) - P_a U_1 \underline{\mu}_1(pj)]^T \left[\underline{y}_a^*(pj) - h(pj) - P_a U_1 \underline{\mu}_1(pj)\right] \tag{4.39}$$

subject to the constraints Eq. (4.38).

4.9 Numerical Examples

Numerical experiments are performed on the third order system in Eq. (4.10), using the indicated coefficients and sample rate. There is one zero outside the unit circle at -3.3104. The desired trajectory is a unit amplitude one Hertz sine wave unless otherwise indicated. There are four issues investigated, and comparisons are made between JiLLL NS and FS with respect to their performances.

4.9.1 Deciding Batch Size *p* to Be Used

If there is no actuator saturation, the output error is guaranteed to be zero every addressed time step so that one might initially think there are no transients. However, the unaddressed time steps and therefore the control actions have transients determined by the system matrix in Eq. (4.29). One uses $P^{\S}$ as indicated in the equation when investigating JiLLL FS, and uses $P^{\#}$ in its place when investigating the corresponding JiLLL NS. When investigating convergence to zero error with initial actuator saturation, using Q instead as in Eq. (4.37) where the first row is no longer producing zero tracking error.

As presented before, all the eigenvalues of the system matrix in Eq. (4.29) must be within the unit circle to have asymptotic stability of the tracking process, even though zero error is guaranteed every addressed step. But there is more that goes beyond this requirement. The one step ahead control aims for zero error every step, no matter what you ask to do in the next time step as long as it does not saturate the actuator. The control law generalized here is a *p*-step batch ahead control that requires to wait *p* time steps before the desired trajectory can be updated, so it is desired to have this *p* as small as possible, making it closer towards one step ahead control.

The largest eigenvalue magnitude is studied for these two stable inverses. Table 2 gives the result for JiLLL NS, and Table 3 gives the result for JiLLL FS. Table 2 shows that the *p*-step batch stable inverse control is stable for $p = 6$ or larger and this is using the doubled sample rate, indicating using a batch of 3 time steps at the original sampling rate is sufficient to produce a stable inverse, generalized the original one step ahead control into a 3-step batch stable inverse control. For this 3rd-order system, using a settling time definition of four times the longest time constant, the settling time is around 45 time steps, or 90 time steps at the doubled sample rate. So this update is quick compared to the dynamics of the system. Note that when $p = 12$ the maximum eigenvalue is -0.0680, and when $p = 18$ it is -0.0064 corresponding to rapid decay of the transients of the unaddressed time steps in the first *p* step update.

On the other hand, Table 3 indicates that JiLLL FS requires a $p = 36$ or larger at the doubled rate, which is a significant percentage of the settling time. The conclusion for now is that, for this application, using JiLLL NS gives a good design with small value of batch size *p*, hence JiLLL NS is preferred to JiLLL FS. Note that these results are independent of the desired trajectory.

Table 2 Eigenvalue of largest magnitude vs. *p* using JiLLL NS

Size of *p*	4	6	8	10	12	14	16	18
Eigenvalue	1.3487	-0.6939	0.3251	-0.1491	0.0680	-0.0310	0.0141	-0.0064

Table 3. Eigenvalue of largest magnitude vs. *p* using JiLLL FS

Size of *p*	10	20	26	34	36	40	46	50
Eigenvalue	13.5679	3.5054	2.4595	1.2089	0.9922	0.6895	0.4170	0.2943

4.9.2 Behavior of the Unaddressed Time Steps

Both JiLLL NS and FS give errors at the addressed time steps that are numerical zeros. But in practice one wants to know what is happening at the unaddressed time steps that are determined by control actions unrelated to the desired trajectory at these steps. Figures 31 gives the errors at the addressed time steps (even time steps at the doubled sample rate) using p = 8 with the one Hertz sine as the desired trajectory, which are all numerically zero starting from the beginning of tracking. Figure 32 gives the errors at the unaddressed time steps where some transient behavior can be observed at the beginning, and the maximum "error" is around 10^{-2}. The root mean square (RMS) of all 200 time steps is 0.0043. For JiLLL FS with p = 40, the error at addressed time steps in simulation are around 10^{-12}, but the error at RMS error of unaddressed time steps is 2.4797, which is totally unacceptable, and increasing to p = 80 does not solve this problem. This is more evidence to indicate that JiLLL NS is preferred to JiLLL FS. Figures 33 and 34 give the plots corresponding to Figures 31 and 32 for the desired trajectory set to a 5 Hz sine wave. Behavior is similar. The RMS error at the unaddressed time steps is now 0.0129.

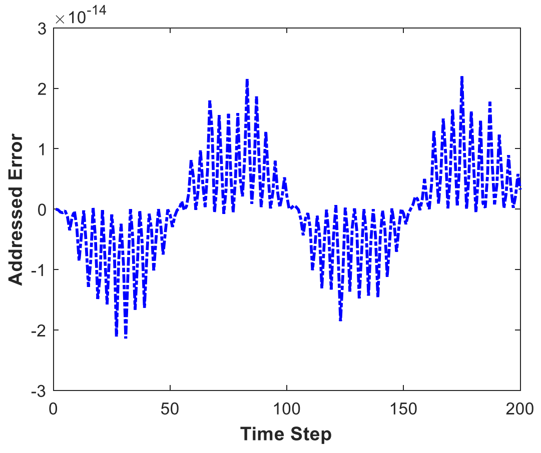


Figure 31. Tracking error at addressed time steps for JiLLL NS using p = 8 and 1Hz desired trajectory.

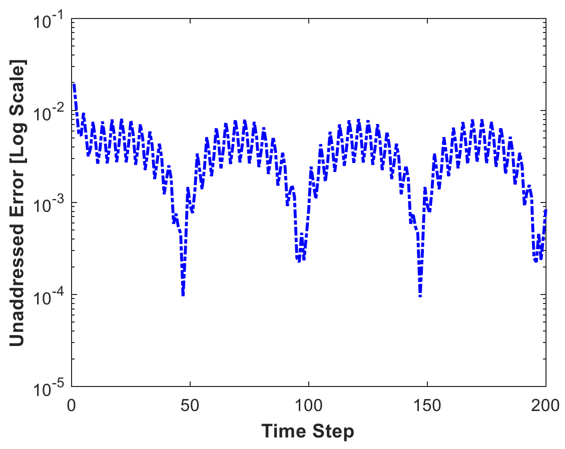
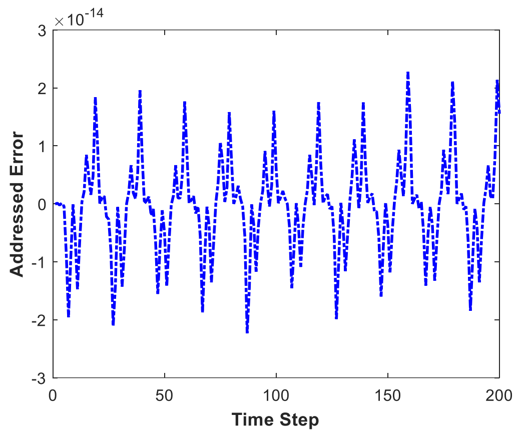


Figure 32. Tracking error at unaddressed time steps for JiLLL NS using p = 8 and 1Hz desired trajectory.



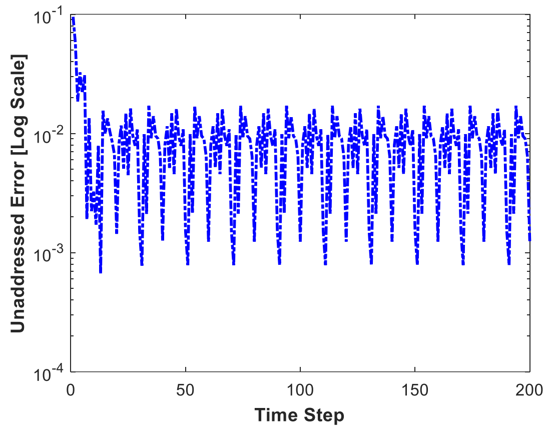
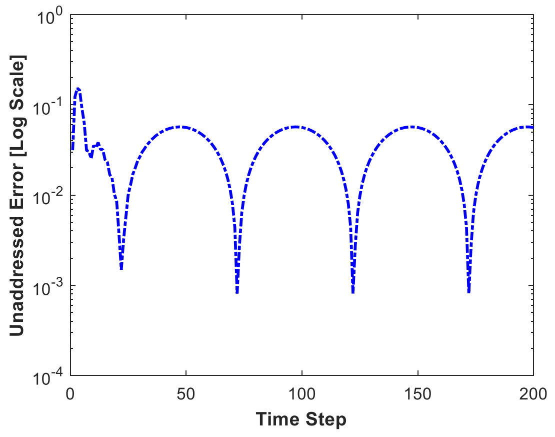


Figure 33. Tracking error at addressed time steps for JiLLL NS using *p* = 8 and 5Hz desired trajectory.

Figure 34. Tracking error at unaddressed time steps for JiLLL NS using *p* = 8 and 5Hz desired trajectory.

One feels tempted to compare this error to the error one would get halfway between addressed steps when using a single zero-order hold instead of two zero-order holds between the addressed time steps. This question does not make sense because one cannot use a single zero-order hold for a pole excess of 3 system -- it results in instability. The error grows by approximately a factor of 3.3 every time step at the times half way between the addressed time steps. Instead, consider a second order system, where there is no need to double the sample rate to produced stability for this system, but this permits to use a single zero-order hold, so one can compare the results. Pick the second order system that results from the 3rd order system considered above by deleting the first order factor, and again use 1Hz desired trajectory. Figure 35 gives the error at the time of the unaddressed time steps in the middle of the single zero-order hold, while Figure 36 is the error at unaddressed time steps when the sample rate is double. The RMS error is substantially better when doubling the sample rate, 0.0434 for Figure 35 and 0.0103 for Figure 36.



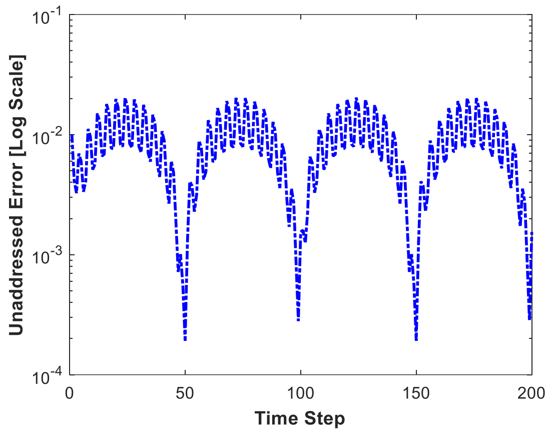


Figure 35. Tracking error half way between addressed time steps using a single zero-order hold on a second order system with 100Hz sample rate.

Figure 36. Tracking error at unaddressed time steps after doubling sample rate, when using JiLLL NS on a second order system with *p* = 8.

4.9.3 Convergence When Initial Conditions Are Far from the Desired Trajectory

Previously, the desired trajectory has been assumed to be feasible, but if the initial condition is far from the desired trajectory, one cannot ask to get zero error after only two time steps. To address this issue, one makes use of the quadratic cost. A problem is simulated where the desired trajectory is a 1Hz cosine instead of sine, which asks to be on the cosine curve at all even valued time steps after starting with zero initial condition, i.e. the initial condition is very far from the desired trajectory.

Figure 37 shows the control action vs. time step without imposing any actuator limitation. It can be seen that very large control action, with magnitude over 100, is needed to get onto the cosine after only two time steps starting from zero initial conditions, and some continued fast changes in the control action are needed. Figure 38 is the same function zoomed in so that one can see that it is sufficient to have a u_{max} of 6 in order to follow the trajectory. Using this value with the quadratic cost, adjusting the control penalty to satisfy the actuator limit each p-step batch update (and changing the batch size to p = 10), results in the addressed error history in Figure 39, and the unaddressed error history in Figure 40.

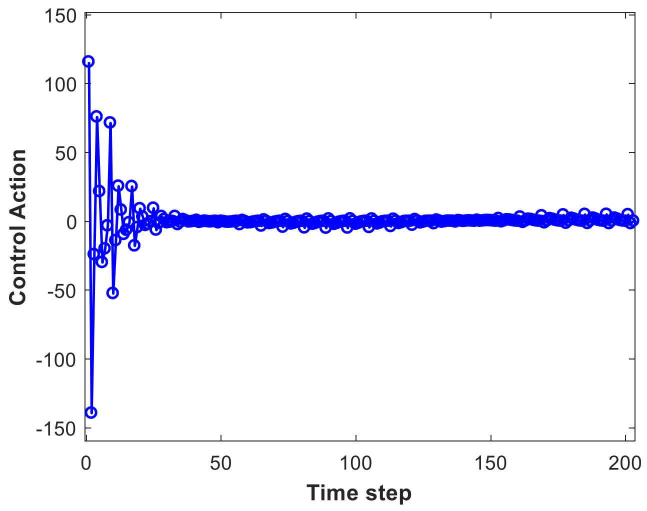


Figure 37. Control action to track cosine starting from zero initial conditions, with no actuator limit.

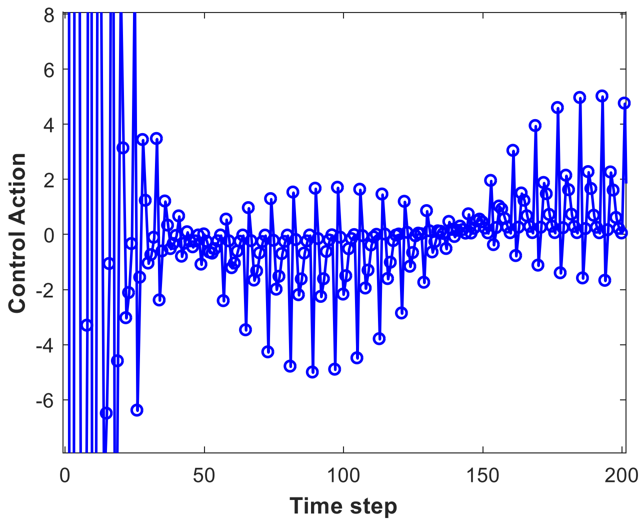


Figure 38. An enlargement of Figure 37.

It is clear that, although zero tracking error is no longer achieved immediately at the addressed time steps, the quadratic cost makes the trajectory converge fast to zero error starting far from the desired trajectory. The value of penalty r versus the p time step updates is given in

Figure 41, and the conversion to zero tracking error is almost complete at the first update, a small r is still needed at update number 2, and r is zero thereafter. Thus, there is quick and smooth transition from initial conditions to the time the output joins the desired trajectory.

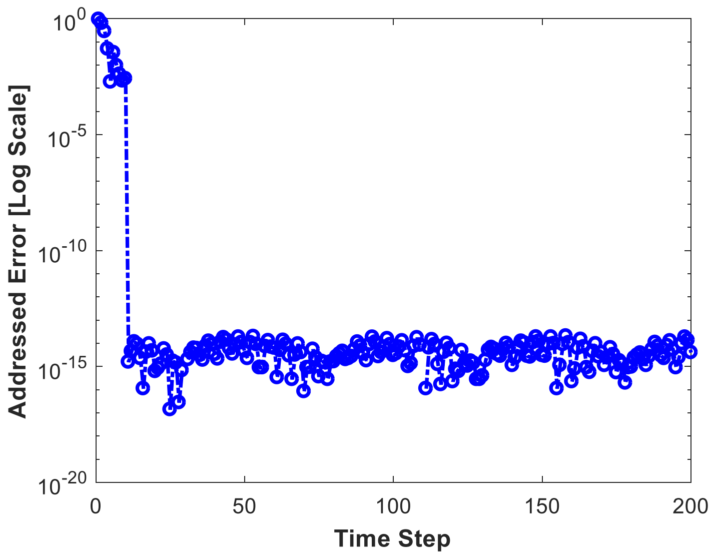


Figure 39. Convergence of tracking error vs. time steps with an actuator limit, starting with initial conditions far from the desired trajectory.

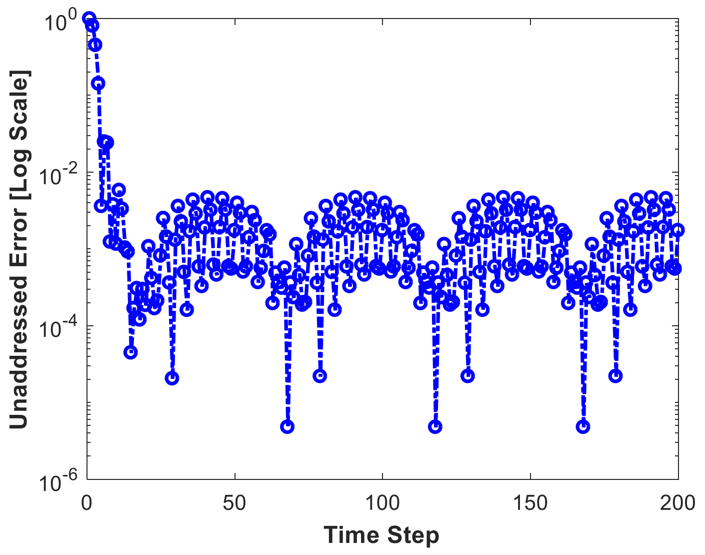


Figure 40. The tracking error at unaddressed time steps corresponding to Figure 39.

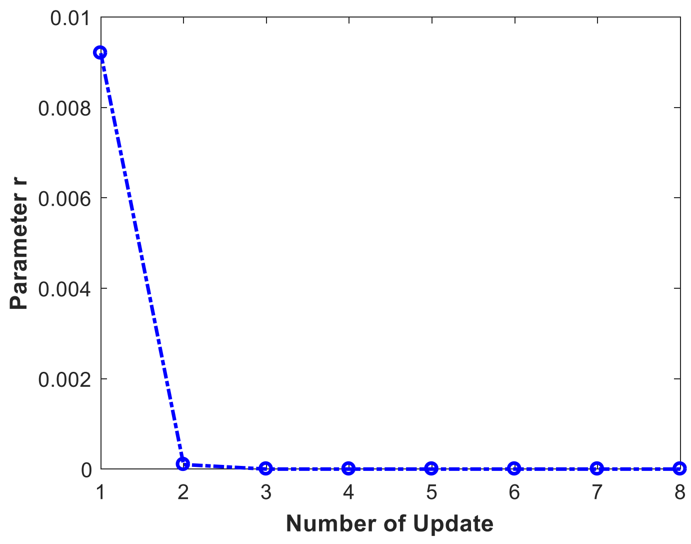


Figure 41. Transition of the control *r* penalty to zero after 3rd update.

4.9.4 Changing the Desired Output After Each *p*-Step Batch Update

Note that with routine control system designs such as PID, one is free to command any change in the output one feels like every time step. But the control system will not follow immediately, needing some transient time to adjust to a new command and never succeeding to actually perform the command. For the generalized one step ahead control design developed here, it is the task of $r(j)$ to perform this transient adjustment when necessary to stay within actuator limits. The approach used here considers using maximum control effort while making the adjustment. One can tune this for slower and less aggressive convergence each batch update without asking for u_{max} every time until reaching zero tracking error.

As a guide to what is reasonable physically when one makes a p-step batch update, consider the 3rd-order differential equation whose discretization produced Eq. (4.20). A step input is a reasonable input in a control system, but Dirac delta function inputs are not. The step input on the input side of the 3rd order differential equation can make a step change in the third derivative term.

Then the second derivative can have a step change in slope, while the first derivative needs to be continuous with continuous first derivative, and finally the specified desired output should be continuous with two derivatives being continuous. These conditions are only slightly relaxed when one considers discrete time problems as is done here. It is concluded that if one changes the desired trajectory from one p-step batch interval to the next, and wants to maintain zero tracking error at each time step, one must constrain the change to correspond to a smooth transition using a function with two continuous derivatives. If this condition is not satisfied, one should expect the control action to have some fast adjustments as the next p-step period starts. If these are large enough to reach actuator saturation, the quadratic cost must be applied and penalty r used to transition to the new desired trajectory.

Figures 42 through 45 study this issue for the 3rd order system with 100Hz original sample rate, when at one of the p-step batch updates the desired trajectory changes in such a way that the desired trajectory is continuous across the update, but the first derivative is discontinuous (batch size $p = 8$ steps at the doubled rate). Before this change the desired trajectory is a 1Hz signal but with a phase lead of 0.02π. After the change the desired trajectory is $2.2 - 5t$. This is shown in Figure 42. Figure 43 shows the control action. It needs to react to the initial conditions not being on the original desired trajectory making some quick adjustment with time step. It is interesting to note that there is no apparent difficulty in adjusting to the trajectory change. Figures 44 shows numerically zero tracking error every addressed time step, and Figure 45 show the decay of error at unaddressed time steps resulting from the initial conditions, and then a rather easy conversion to the trajectory after the transition.

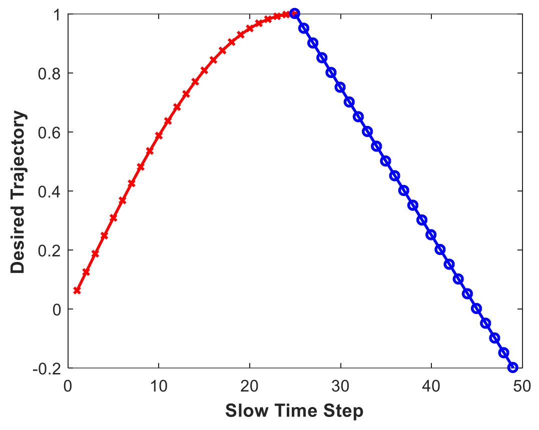


Figure 42. Desired trajectory with a discontinuous first derivative at the start of a *p*-step batch update.

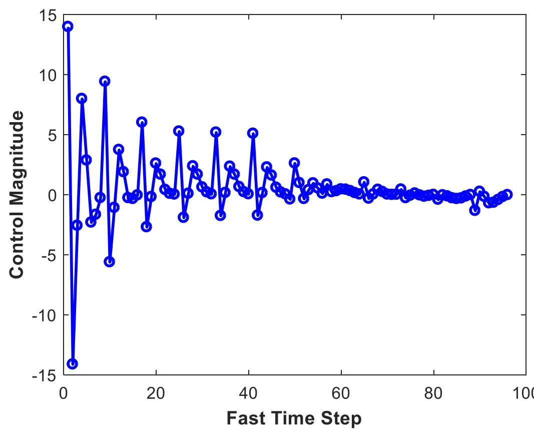


Figure 43. Control actions associated with Figure 42.

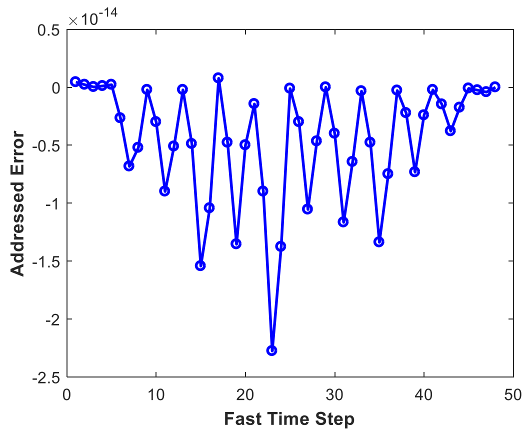


Figure 44. Numerically zero tracking error both before and after the discontinuity for the addressed time steps.

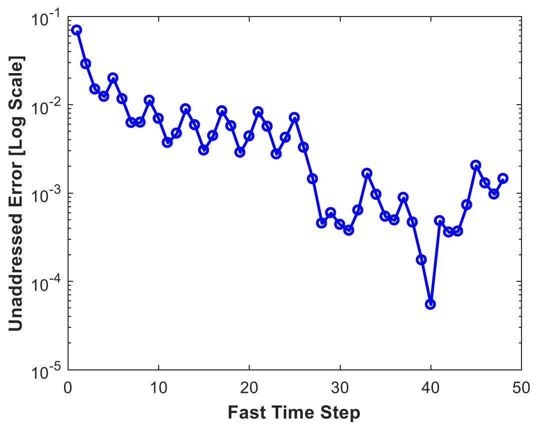


Figure 45. The error at the unaddressed time steps for the desired trajectory in Figure 42.

4.10 Conclusions

One step ahead control aims for zero tracking error every time step based on one's model. The promise of zero tracking error is very attractive. And it makes a strong contrast to any routine control law, which will never promise zero error for any trajectory. Integral control promises zero error to constant commands, but only asymptotically as time goes to infinity. The digital one step ahead control law fails in the world for the majority of physical systems, because any physical system with pole excess in continuous time of 3 or more and a reasonable sample rate has an unstable inverse. The in-house developed stable inverse theory Longman JiLLL NS, NI, FI, FS is applied to solve the unstable inverse issue involved in control. There are corresponding mathematical proofs that they produce stable inverses for sampling zeros and intrinsic zeros outside the unit circle.

This chapter generalizes the one step ahead control law to a *p*-step batch stable inverse control law using JiLLL NS and FS to produces zero tracking error at all the addressed time steps. These two use a uniform pattern of updates for each addressed time step. The stable inverse is achieved by introducing two zero-order holds between each time step for which one asks for zero error for one zero outside the unit circle. Then the *p*-step batch stable inverse control uses this stable inverse for successive *p*-time step updates rather than the original one time step update. There are initial condition transients associated with the unaddressed time steps and correspondingly with the input. An if and only if stability condition for convergence for all possible initial conditions is given and used to determine the minimum value of batch size *p* that can be used. It indicates that this number can be quite small, indicating that one only needs to specify the command for 6 future time steps at the doubled sample rate before one can pick some new command for the third order example investigated. This is accomplished using the JiLLL NS, making the generalized control law close enough towards the original one step ahead control law. On the other hand, the JiLLL FS stable inverse was found to require a much larger value of *p* to produce stability, and the error at unaddressed time steps was unacceptable. The conclusion is that when using a stable inverse to create a *p*-step batch stable inverse control, one should use the JiLLL NS instead of JiLLL FS, which is also much simpler since no factorization process is involved in JiLLL NS. The *p*-step batch stable inverse control, generalized from one-step ahead control, can also be viewed as a Model Predictive Control (MPC) achieving zero tracking error at required time steps instead of minimizing a cost function between control magnitude and output error as conventionally does.

The second practical difficulty with one step ahead control is that it does not consider any actuator saturation limitations. A quadratic cost with a control penalty adjusted to meet saturation limits is used here to move from initial conditions onto the desired trajectory. It is also used to handle changes in command made by the user from one *p*-step batch period to the next, whenever the changes are too aggressive to follow without saturating the actuator. If desired, one can adjust this penalty for more gradual convergence to zero tracking error, tuning the transient behavior.

Chapter 5 Development of Batch Stable Inverse Indirect Adaptive Control of Systems with Unstable Discrete-Time Inverse

5.1 Introduction

In classical control systems, the command to the system is the desired output. The actual output is a solution of the governing differential equation, containing the command in the forcing function. Except under very unusual circumstances, the particular solution output is not equal to the command. In addition, one must expect error in the model used in the design. Indirect adaptive control seeks to address both of these sources of error (Goodwin et al., 1980; Goodwin et al. 1984). It uses the current estimated model and asks for zero error in the next step, and also in each time step it makes use of the most recent input/output data to update the estimated model. This approach promises to produce zero tracking error after convergence, following whatever command one gives the control system. In addition to indirect adaptive control by Goodwin, there are two other big contributions to adaptive control around 1980. Narendra did direct adaptive control, finding a way to create a control law based on Lyapunov functions without identifying a model and then computing a control; instead, it went direct to the control (Narendra et al., 2012). The third contribution is self-tuning regulars, with the idea of having a PID or similar controller, and based on data of how system is performing, tunes the gains automatically (Cameron et al., 1984). Learning control, adaptive control, and later on learning-adaptive control, using an extension to the generalized secant method have been presented in the literature and compared with the self-tuning regulator as well as a few numerical optimization methods in controlling non-linear systems (Beigi et al., 1991; Beigi, 1992; Beigi, 1992; Beigi, 1997). Li et al. (1993) presented a learning self-tuning regulator based on parameter estimation for nonlinear piezo-actuator and has successfully reduced tracking error while performing repetitive tasks.

However, convergence requires that the inverse of the system transfer function be asymptotically stable, which has limited the applications of both one step ahead control and indirect adaptive control because this condition is not satisfied in a majority of systems one would like to control (Åström et al., 1984). The adaptive updates need a discrete time model. When a differential equation is fed by a zero-order hold, one can replace the differential equation by a difference equation without any approximation, i.e. it produces the same output as the differential equation at each time step. The process of converting to a difference equation introduces zeros into the transfer function, and for a reasonable sample rate, and at least 3 more poles than zeros in the original differential equation, there will be zeros introduced that are outside the unit circle, making the digital system inverse unstable. For systems with pole excess of 3 or more, the zeros introduced during discretization can remain within the unit circle only if the sampling rate is slow enough, which results in aliasing and is not practical for applications.

To overcome the instability when inverting a system with naturally unstable inverse, different kinds of stable inverse theory have been developed to obtain a feasible stable inverse. Devasia's research group developed a stable inverse theory that gives a stable inverse for a given time history, but this theory requires a pre-actuation period in advance of the zero tracking error trajectory (Devasia et al., 1996). Generalized Sampled-Data Hold Function (GSHF) has been applied to handle the unstable inverse nature of the system caused by non-minimum phase zeros in (Ibeas et al., 2010). In fact, GSHF is similar to the commonly applied zero-order hold (ZOH), but GSHF divides the original ZOH sampling period T into N segments, i.e. GSHF fast sampling period $T' = T/N$ with $N \geq n$ where n is the order of the continuous time transfer function denominator. The sensitivity and robustness of GSHF are studied and presented in Kabamba (1987) and Freudenberg et al. (1997), where Kabamba (1987) solved a few problems including optimal noise rejection and Freudenberg et al. (1997) argued that using GSHF for zero shifting might lead to unacceptable intersample behavior and sensitivity in systems. As presented in the previous chapters, a new stable inverse theory named Longman JiLLL have been developed, where JiLLL stands for the researchers involved in the development of this stable inverse theory (Ji et al., 2019; Longman et al., 2017; Li et al., 2010; LeVoci et al., 2004). This in-house developed stable inverse theory divides the original sampling period into $n_O + 1$ segments where n_O is the number of non-minimum phase zeros, and JiLLL has been applied into many controls, such as linear model predictive control and one step ahead control, to address the instability in the inverse of high-order systems whose continuous-time pole excess is three or more (Zhu et al., 2017; Wang et al., 2018).

The work in Ibeas et al. (2010) used Least Squares estimation algorithm for the adaptive control cases, whereas in this chapter Projection Algorithm, which has the advantages of relatively simplicity and capable of tracking time-varying parameters automatically, are applied for parameter updates in model estimation. Projection-based update procedures have also been applied to Iterative Learning Control (Avrachenkov et al., 2002). This chapter makes use of batch stable inverse control presented in Chapter 4 generalized from one step ahead control that takes advantage of new in-house developed stable inverse theory for such systems, and make appropriate adjustments to the projection algorithm. Mathematical proof and experiment validation are presented to show that, with the help of in-house developed stable inverse theory, the batch stable inverse indirect adaptive control can gradually achieve zero tracking error with a bounded feasible control action sequence for systems whose discrete time transfer functions have zeros located outside the unit circle, and the parameters in the estimated model converge as time goes on.

5.2 Zeros of Discretized Systems

Digital adaptive control has two basic building blocks, the one step ahead control algorithm and the algorithm for real time model updates such as projection algorithm and least square algorithm. One-step ahead control is the fundamental part of the adaptive control. For illustration purposes, consider a third-order differential equation system with Laplace transfer function as Eq. (5.1)

$$\frac{d^3y(t)}{dt^3} + \alpha_1 \frac{d^2y(t)}{dt^2} + \alpha_2 \frac{dy(t)}{dt} + \alpha_3 y(t) = \beta_1 u(t)$$

$$Y(s) = \frac{\beta_1}{s^3+\alpha_1 s^2+\alpha_2 s+\alpha_3} U(s) = G(s)U(s) \tag{5.1}$$

When the input comes through a zero-order hold, this can be replaced by a difference equation with identical solution at the sample times, and associated *z*-transfer function as Eq. (5.2)

$$y(k+1) + a_1 y(k) + a_2 y(k-1) + a_3 y(k-2) = b_1 u(k) + b_2 u(k-1) + b_3 u(k-2)$$

$$Y(z) = \frac{b_1 z^2+b_2 z+b_3}{z^3+a_1 z^2+a_2 z+a_3} U(z) = G(z)U(z) \tag{5.2}$$

by using the conversion $G(z) = (1 - z^{-1})Z[\frac{G(s)}{s}]$, where Z indicates the *z*-transform of the unit step response specified in the square bracket.

One way to view the inverse problem is to write Eq. (5.2) for each time step. At the time step k, $u(k-1)$ and $u(k-2)$ are known, $y(k-1)$ and $y(k-2)$ are past known measurements. When $y(k)$ arrives, one can solve for the required control action $u(k)$ in order to produce $y^*(k+1)$ at the next time step, as presented in Eq. (5.3):

$$u(k) = \frac{1}{b_1}[y^*(k+1) + a_1 y(k) + a_2 y(k-1) + a_3 y(k-2) - b_2 u(k-1) - b_3 u(k-2)]$$

(5.3)

However, there are situations when the accuracy of the model might not be reliable. In these situations, the control action $u(k)$ is computed from the past data and the current model, as presented in Eq. (5.4). After computing $u(k)$, it is applied to the actual unknown system and the actual output is measured. The model parameters are then updated based on real time output and input data before computing the next control action $u(k+1)$. This is, in fact, an adaptive one-step ahead control, which aims to gradually achieve zero tracking error while updating the estimated model.

$$u(k) = \frac{1}{\hat{b}_1}\left[y^*(k+1) + \hat{a}_1 y(k) + \hat{a}_2 y(k-1) + \hat{a}_3 y(k-2) - \hat{b}_2 u(k-1) - \hat{b}_3 u(k-2)\right]$$

(5.4)

Substitute the desired output $y^*(k)$ into the left-hand side of Eq. (5.2) produces a known forcing function $f(k)$ to the resulting difference Eq. (5.2), and the input history needed to produce this desired output history is the solution of the nonhomogeneous difference equation $b_1 u(k) + b_2 u(k-1) + b_3 u(k-2) = f(k)$ (LeVoci et al., 2004). The solution is the sum of a particular solution and two linearly independent solutions of the homogeneous equation as Eq. (5.5):

$$u(k) = u_p(k) + C_1(z_1)^k + C_2(z_2)^k \tag{5.5}$$

The characteristic equation whose roots determine these two solutions are the roots of the zeros polynomial. The zeros introduced during the discretization process are a function of the pole excess in the differential equation, i.e. the number of poles minus the number of zeros. The number of extra zeros introduced is the number needed to produce one more pole than zero in the discrete time transfer function, so that there is a one time step delay from a change in the zero-order hold input, until one observes a resulting change in the sampled output. Thus, for a third-order differential equation with no zeros as above (or with 3 more poles than zeros) there will be two zeros introduced. The asymptotic locations, as sample time tends to zero, of zeros introduced outside the unit circle are (Åström et al., 1980): there will be a zero located at -3.7321 for the system with pole excess of 3; for the system with pole excess of 4, the outside-unit-circle zero will be located at -9.899; for the system with pole excess of 11, there will be five zeros located outside the unit circle, with two largest magnitude zeros located at -1958.6431 and -59.9893 respectively. These zeros located outside the unit circle all have magnitudes greater than 1, and, as illustrated by Eq. (5.5), they make the input control action grow exponentially as time goes on when one wants to achieve zero tracking error at all the time steps. For example, if one wants to track a unit magnitude sine wave of 1 Hz with zero tracking error, the magnitude of control action required for the pole excess of 3 system after one second will be at the level of 10^{50}; for the pole excess of 4 system, the control magnitude will be at 10^{100} level; and for the pole excess of 11 system, the control action will be at 10^{300} level. These exponentially growing input actions are for sure too aggressive, and therefore impossible to implement.

It is this instability of the inverse problem that one seeks to eliminate so that indirect adaptive control can have stable control inputs that produce zero tracking error at addressed time steps.

5.3 Batch Inverse Control

The indirect adaptive control of Goodwin et al. (1980) and Goodwin et al. (1984) needs recursive real time system inverse computation -- the one step ahead control which is often unstable. The in-house developed stable inverse theory Longman JiLLL, which has 4 formats in total, can produce new system inverses that are stable. There exists a previous literature that presents stable inverses (Devasia et al., 1996). Both methods address the problem of zeros outside the unit circle introduced by discretization, and both are batch computations, producing an inverse over a prescribed finite time interval. The next section briefly illustrates on how a new method building on Longman JiLLL NS can be converted to real time, making a p-step batch stable inverse control in place of the original one step ahead control. The method related to Devasia et al. (1996) is not amenable to the conversion because they need to append pre-actuation to the desired trajectory.

The stable inverse uses a model that presents the relation of the whole history of inputs to the whole history of outputs. Convert the third order scalar difference equation of Eq. (5.2) to state space form, and form

$$\bar{y}(k+1) + a_1\bar{y}(k) + a_2\bar{y}(k-1) + a_3\bar{y}(k-2) = u(k) \tag{5.6}$$

Then, superposition says the output is given in terms of the defined state variables as

$$y(k) = b_1\bar{y}(k) + b_2\bar{y}(k-1) + b_3\bar{y}(k-2) = Cx(k) \tag{5.7}$$

$$x(k) = [\bar{y}(k-2) \quad \bar{y}(k-1) \quad \bar{y}(k)]^T,\ C = [b_3 \quad b_2 \quad b_1]$$

Eq. (5.2) in state variable form becomes

$$x(k+1) = Ax(k) + Bu(k) \qquad A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.8}$$

$$y(k) = Cx(k)$$

Of course, by taking *z*-transforms, one can retrieve the original scalar difference equation from the *z*-transfer function, using $Y(z) = [C(zI - A)^{-1}B]U(z)$. Propagating the solution to the state space equation for *p*-time steps produces the relationship between the input and output *p*-time step histories, and the initial conditions

$$\underline{y} = P\underline{u} + \bar{A}x(0) \qquad P = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \cdots & CB \end{bmatrix} \qquad \bar{A} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^p \end{bmatrix} \tag{5.9}$$

where $\underline{y} = [y(1) \quad y(2) \quad \cdots \quad y(p)]^T$ and $\underline{u} = [u(0) \quad u(1) \quad \cdots \quad u(p-1)]^T$ are the output and input vector respectively. $\bar{A}$ is an observability matrix, and the Toeplitz matrix P relates the output vector with the input vector. Note that the entries in the output vector start with time step one, and input start with time step 0, as it is normal to assume there is a one-step delay between output and input.

Eq. (5.9) shows that a *p*-step history of input control action will produce a *p*-step history of output, where *p* needs to be big enough so that steady state frequency response will dominate, and furthermore the stability could be guaranteed. Using this model the solution of the inverse problem to find the $\underline{u}^*$ that produces a desired output $\underline{y} = \underline{y}^*$, is given by

$$\underline{u}^* = P^{-1}(\underline{y}^* - \bar{A}x(0)) \tag{5.10}$$

However, due to the fact that P^{-1} is ill-conditioned and unstable, the resulting control action grows exponentially with time and it is impossible to implement such a kind of control action.

5.4 Stable Batch Inverse Control

Using Eq. (5.2), it is obvious how to recognize an unstable inverse, simply look for roots of the zeros polynomial in the numerator of the transfer function below Eq. (5.2). Longman et al.

(2017) explains how to recognize an unstable inverse by examining the singular value decomposition $P = USV^T$. For p large enough, the singular values converge to Eq. (5.2)'s magnitude frequency response at the frequencies that can be seen in *p* time steps. However, a zero outside the unit circle introduces a singular value with its own right and left singular vectors. The signature in the *P* matrix corresponding to an unstable inverse system is: (1) A linear decay on a log scale of a singular value as a function of matrix size *p*. (2) A corresponding pair of input and output singular vectors which have opposite slopes, with the input singular vector component magnitudes growing linearly with time step on a log scale, and the output singular vector component magnitudes decaying with the same slope. The latter can be understood by noting that to correct an error near the beginning of the p-step process expressed in terms of the output singular vector, the control action needed, given in terms of the input singular vector, will grow exponentially with time step.

There are studies working on obtaining a stable inverse for systems whose inverses are naturally unstable. As described in Chapter 4, there are four versions of in-house developed stable inverse theory Longman JiLLL in total, i.e., FI, FS, NI, NS. Both JILLL FI and JiLLL FS factor the *P* matrix into a matrix containing zeros outside the unit circle and a matrix containing the other zeros, and the poles (all of which are inside the unit circle as a system is assumed to be stable). JiLLL FI says delete the number of initial time steps in the desired trajectory equal to the number of zeros outside the unit circle. Then use Moore-Penrose pseudo inverse of the factor associated with the zeros outside to solve the resulting underspecified set of equations for the control input. The result is zero error at the remaining time steps, the addressed time steps. JiLLL FS says to double the sample rate, but ask for zero error only at the original sample times – for the case of one zero outside (for two zeros outside, introduce two intermediate sample times). Observe that this can be thought of as using a generalized hold. JiLLL NI and JiLLL NS analogous the FI and FS respectively have been developed, but they two do not factor the P matrix. Instead of using a pseudo inverse of the factor containing the zeros outside, one simply takes the pseudo inverse of P itself after deleting rows related to unaddressed steps. It is proved that these not only produce stable inverses, but also are significantly more convenient because no factorization process is required. In addition, the research establishes that the performance at the newly introduced sample times in JiLLL NS is significantly better than JiLLL FS. Note that JiLLL NS, similar to JiLLL FS, has a uniform spacing of addressed time steps, so it is natural to try to use it in a stable batch inverse control algorithm with batch size of p-time steps at fast sampling rate, as will be done here.

To use the inverse JiLLL NS, first write P for all time steps after increasing the sample rate. Then group the rows associated with the original sample times into a matrix P_a, and the rows associated with the unaddressed time steps into matrix P_u. Similarly, group the output into $\underline{y}_a$ and $\underline{y}_u$ and the desired trajectory is denoted by $\underline{y}_a^*$, prescribed only at the original time steps before increasing the sample rate. The inverse model proved to be stable according to the above criteria is then given by

$$\underline{u}^* = P_a^\dagger(\underline{y}_a^* - \bar{A}x(0)) \tag{5.11}$$

5.5 Recursive Batch Stable Inverse Control

The one step ahead control of adaptive control theory inverts the system, and does so in a recursive way, updating each time step. The stable inverse presented in the previous section is a batch computation, giving a history of inputs to produce a history of outputs. One needs to find a way to make this batch stable inverse into a recursive process. The resulting batch stable inverse indirect adaptive control will need to know the desired output for the next p time steps, instead of just the next time step. Unlike the one step ahead adaptive control with which one can change the desired output arbitrarily every step, updates will be limited to once every p time steps.

As detailed in Chapter 4, the third-order system by Eq. (5.2) can be generalized into batch stable inverse control, and the dynamics from batch to batch is

$$\begin{bmatrix} \underline{y}_a(p(j+1)) \\ \underline{y}_u(p(j+1)) \\ \underline{u}(p(j+1)) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ P_u P_a^{\dagger} P_{2aa} - P_{2ua} & P_u P_a^{\dagger} P_{2au} - P_{2uu} & P_{1u} - P_u P_a^{\dagger} P_{2aa} \\ P_a^{\dagger} P_{2aa} & P_a^{\dagger} P_{2au} & -P_a^{\dagger} P_{1a} \end{bmatrix} \begin{bmatrix} \underline{y}_a(pj) \\ \underline{y}_u(pj) \\ \underline{u}(pj) \end{bmatrix} + \begin{bmatrix} I \\ P_u P_a^{\dagger} \\ P_a^{\dagger} \end{bmatrix} \underline{y}_a^*\big(p(j+1)\big) \qquad (5.12)$$

For the case of one zero outside the unit circle, the first partition vector on the left for the addressed steps is of dimension $p/2$ as is the second partition vector for unaddressed time steps, while the third partition vector for the control has dimension p, for a total dimension of $2p$ total time steps at the fast sample rate.

In the recursive batch stable inverse control, the time steps in the current period, are needed as initial conditions for the next period, and they keep being changed every update, coupling the equations from period to period. Although the error at the addressed time steps are always zero, the output at the unaddressed time steps can exhibit transients from batch to batch. In order for the recursive batch update to be stable, it is necessary that the square system matrix in Eq. (5.12) have all eigenvalues less than one in magnitude. How much less than one determines how fast the behavior at the unaddressed time steps converge with the batch updates.

One can use this stability condition to determine if, based on one's current model, the transients of the recursive updates will go to zero as updates j progress. This can also be used to determine what the value of p is needed to produce stability. Examine the first row partition of the equation, which has 3 zero entries, to see that the converged solution has $\underline{y}_a(p(j+1)) = \underline{y}_a^*(p(j+1))$ producing zero error at every addressed time step in the batch update, for all updates, after the true system model has been identified.

Note that it is certain that there exists a value of p that creates a stable dynamic process: If one samples so slowly that the system reaches steady state response to the zero-order hold input by the end of the time step, then all initial condition terms become negligible. Matrices P_1, P_2 that contain the initial conditions for the next batch can be set to zero, and the second and third row partitions become zero as well as the first row partition, producing all zero eigenvalues.

The effect of model error should also be taken into account. The control computation in Eq. (4.28) using JiLLL NS is made using the current model. Denote the estimated coefficients in this case by introducing a hat on top of symbols to make them different from the real world system coefficients that are unknown in adaptive control cases

$$\underline{u}(p(j+1)) = \hat{P}_a^\dagger[\underline{y}_a^*(p(j+1)) - \hat{P}_{1a}\underline{u}(pj) + \hat{P}_{2aa}\underline{y}_a(pj) + \hat{P}_{2au}\underline{y}_u(pj)] \qquad (5.13)$$

Substituting this control action computed from the estimated model to be updated into the unknown real world system Eq. (4.26), and Eq. (5.12) becomes

$$\begin{bmatrix} \underline{y}_a(p(j+1)) \\ \underline{y}_u(p(j+1)) \\ \underline{u}(p(j+1)) \end{bmatrix} = \begin{bmatrix} P_a\hat{P}_a^\dagger\hat{P}_{2aa} - P_{2aa} & P_a\hat{P}_a^\dagger\hat{P}_{2au} - P_{2au} & P_{1a} - P_a\hat{P}_a^\dagger\hat{P}_{1a} \\ P_u\hat{P}_a^\dagger\hat{P}_{2aa} - P_{2ua} & P_u\hat{P}_a^\dagger\hat{P}_{2au} - P_{2uu} & P_{1u} - P_u\hat{P}_a^\dagger\hat{P}_{1a} \\ \hat{P}_a^\dagger\hat{P}_{2aa} & \hat{P}_a^\dagger\hat{P}_{2au} & -\hat{P}_a^\dagger\hat{P}_{1a} \end{bmatrix} \begin{bmatrix} \underline{y}_a(pj) \\ \underline{y}_u(pj) \\ \underline{u}(pj) \end{bmatrix} +$$

$$\begin{bmatrix} P_a\hat{P}_a^\dagger \\ P_u\hat{P}_a^\dagger \\ \hat{P}_a^\dagger \end{bmatrix} \underline{y}_a^*(p(j+1)) \qquad (5.14)$$

It can be seen from Eq. (5.14) that only the addressed part of the estimated model is involved, as all the matrices with hats have subscripts of “a” not “u”.

Both one step ahead control and batch stable inverse control can be too aggressive in practical applications, and Chapter 4 develops methods to intelligently smooth the convergence process when the current trajectory is far from the desired trajectory. Some similar approach may be adopted for the adaptive recursive batch stable inverse updates.

5.6 Batch Stable Inverse Indirect Adaptive Control

There are situations where the initially estimated model is not reliable, and one feels like to improving the model based on real time measurements. In the following batch stable inverse indirect adaptive control, after getting the *p*-step history output from the actual system, the estimated model is updated via Projection Algorithm with the input and output history and previous estimated model parameters before computing the next *p*-step history of input control actions. In this way, the estimated model gets improved while aiming to achieve zero tracking error.

Define a matrix containing all the parameters of the real-world system in Eq. (5.12) $\theta_0^T = [P \quad P_1 \quad P_2]$, which can be considered as composed of the addressed part $\theta_{0a}^T = [P_a \quad P_{1a} \quad P_{2a}]$ and unaddressed part $\theta_{0u}^T = [P_u \quad P_{1u} \quad P_{2u}]$. The estimated addressed part of the model parameters, which is different from the unknown real-world system, is denoted as $\hat{\theta}_a(j)^T = [\hat{P}_a(j) \quad \hat{P}_{1a}(j) \quad \hat{P}_{2a}(j)]$, and the i^{th} row of matrix $\hat{\theta}_a(j)$ is denoted as vector $\underline{\hat{\theta}}_{a,i}(j)$. Define another vector $\underline{\varphi}(j)^T = [\underline{u}(j)^T \quad \underline{u}(j-1)^T \quad -\underline{y}(j-1)^T]$ containing the *p*-step input for the present iteration $\underline{u}(j)^T$, the *p*-step input from the past iteration $\underline{u}(j-1)^T$, and the *p*-step system actual output from the past iteration $\underline{y}(j-1)^T$. The batch size *p* has been dropped in the following formula denotations for simplicity, but one should always be reminded that every iteration update from *j* to *j*+1 will change all the *p* time steps at fast sampling rate in the batch. So the actual system output in the present iteration is expressed as $\underline{y}(j) = \theta_0^T \underline{\varphi}(j)$, and the *i*th addressed output of $\underline{y}(j)$ is expressed as $y_{a,i}(j) = \underline{\varphi}(j)^T \underline{\theta}_{a,i}$.

Assumptions for the batch stable inverse indirect adaptive control: (1) the delay between input and output is known. (2) the upper bound for the system order *n* is known. (3) A batch size *p* big enough is chosen for the purpose of stability. The steps of the batch stable inverse indirect adaptive control algorithm are presented as follows:

Step 1: Compute the desired input $\underline{u}(j)$ aiming to produce zero tracking error from the estimated model via Eq. (5.16):

$$\underline{u}(j) = \hat{P}_a^{\dagger}(j-1)\left[\underline{y}_a^*(j) - \hat{P}_{1a}(j-1)\underline{u}(j-1) + \hat{P}_{2a}(j-1)\underline{y}(j-1)\right] \qquad (5.15)$$

Step 2: Form $\underline{\varphi}(j)$ with $\underline{u}(j)$ from Step 1:

$$\underline{\varphi}(j)^T = \begin{bmatrix}\underline{u}(j)^T & \underline{u}(j-1)^T & -\underline{y}(j-1)^T\end{bmatrix} \qquad (5.16)$$

Step 3: Apply the control action from Eq. (5.15) to the real-world system and record the corresponding output:

$$\underline{y}(j) = P\underline{u}(j) + P_1\underline{u}(j-1) - P_2\underline{y}(j-1) \qquad (5.17)$$

Step 4: Update every row of the addressed estimated model parameters using Projection Algorithm:

$$\underline{\hat{\theta}}_{a,i}(j) = \underline{\hat{\theta}}_{a,i}(j-1) + a\frac{\underline{\varphi}(j)}{c+\underline{\varphi}(j)^T\underline{\varphi}(j)}\left[y_{a,i}(j) - \underline{\varphi}(j)^T\underline{\hat{\theta}}_{a,i}(j-1)\right] \qquad (5.18)$$

Note that the initial estimation $\hat{\theta}(0)$ is known and given. For the 4 steps above, the update iteration $j \geq 1$, and a is a gain constant influencing the parameter convergence rate with $\varepsilon < a < 2 - \varepsilon$ where $0 < \varepsilon < 1$.

If the batch stable inverse indirect adaptive control (5.15)-(5.18) is applied to the real world system whose MPC format is expressed by Eq. (5.17) recursively under the assumptions mentioned above, then:

(I) Estimated parameters $\{\hat{\theta}_a(j)\}$ in the model will converge, although not necessarily to θ_{0a}.

(II) $\{\underline{y}(j)\}$ and $\{\underline{u}(j)\}$ are bounded.

(III) $\lim_{j\to\infty} |y_{a,i}(j) - y^*_{a,i}(j)| = \lim_{j\to\infty} |e_{a,i}(j)| = 0$

5.6.1 Projection Algorithm

The projection algorithm updates the estimate of coefficients from $\hat{\underline{\theta}}_{a,i}(j-1)$ to $\hat{\underline{\theta}}_{a,i}(j)$, in such a way as to minimize the square of the Euclidean norm of the change, while making the new coefficients predict the latest measurement $y_{a,i}(j)$. To do this one generates the cost function

$$J = \frac{1}{2}\left\|\hat{\underline{\theta}}_{a,i}(j) - \hat{\underline{\theta}}_{a,i}(j-1)\right\|^2 + \lambda\left[y_{a,i}(j) - \underline{\varphi}(j)^T \hat{\underline{\theta}}_{a,i}(j)\right] \tag{5.19}$$

The λ is a Lagrange multiplier to enforce the new coefficients $\hat{\underline{\theta}}_{a,i}(j)$ will predict the latest data $y_{a,i}(j)$. To find the update, compute $\frac{\partial J}{\partial \lambda} = 0$ and solve for λ. Then substitute into $\frac{\partial J}{\partial \hat{\underline{\theta}}_{a,i}(j)} = 0$ to obtain

$$\hat{\underline{\theta}}_{a,i}(j) = \hat{\underline{\theta}}_{a,i}(j-1) + a\frac{\underline{\varphi}(j)}{c + \underline{\varphi}(j)^T \underline{\varphi}(j)}\left[y_{a,i}(j) - \underline{\varphi}(j)^T \hat{\underline{\theta}}_{a,i}(j-1)\right] \tag{5.20}$$

The square bracket term is the prediction error using $\hat{\theta}(k-1)$ coefficients before update. The c can be introduced to guard against dividing by zero, and the a allows one to adjust the convergence rate, the initial estimation $\hat{\theta}(0)$ is given.

Parameter convergence can be analyzed as follows. Define the parameter estimation error as $\underline{\tilde{\theta}}_i(j) = \underline{\hat{\theta}}_i(j) - \underline{\theta}_{0,i}$. Subtract $\underline{\theta}_{0,i}$ from both sides of Eq. (5.20), and compute $\left\|\underline{\tilde{\theta}}_{a,i}(j)\right\|^2 = \underline{\tilde{\theta}}_{a,i}(j)^T \underline{\tilde{\theta}}_{a,i}(j)$, after that compute $\left\|\underline{\tilde{\theta}}_{a,i}(j)\right\|^2 - \left\|\underline{\tilde{\theta}}_{a,i}(j-1)\right\|^2$ as in Eq. (5.21)

$$\left\|\underline{\tilde{\theta}}_{a,i}(j)\right\|^2 - \left\|\underline{\tilde{\theta}}_{a,i}(j-1)\right\|^2 = \left\|\underline{\hat{\theta}}_{a,i}(j) - \underline{\theta}_{0a,i}\right\|^2 - \left\|\underline{\hat{\theta}}_{a,i}(j-1) - \underline{\theta}_{0a,i}\right\|^2$$

$$= a \cdot \left[-2 + a\frac{\underline{\varphi}(j)^T\underline{\varphi}(j)}{c+\underline{\varphi}(j)^T\underline{\varphi}(j)}\right] \cdot \frac{\underline{\tilde{\theta}}_{a,i}(j-1)^T\underline{\varphi}(j)\underline{\varphi}(j)^T\underline{\tilde{\theta}}_{a,i}(j-1)}{c+\underline{\varphi}(j)^T\underline{\varphi}(j)} \leq 0 \qquad (5.21)$$

Observe Eq. (5.21), it can be seen that this difference in the Euclidean norm of the error is monotonically decreasing, or it can stay the same, but it cannot increase, indicating the Euclidean norm of the vector $\underline{\tilde{\theta}}_{a,i}(j)$ is a non-increasing function. Since $\left\|\underline{\tilde{\theta}}_{a,i}(j)\right\|^2$is a non-negative bounded non-increasing function, it converges, proving that the coefficients will converge, but not necessarily to zero error. It can fail to converge to the true system parameters, if some part or mode of the system dynamics is not excited, i.e. not observed in the data. In this case, the difference equation with wrong coefficients can still predict the output, provided the mode remains un-excited.

5.6.2 Uniform Boundedness Condition

Continue with Eq. (5.21), it can be seen that $a \cdot \left[-2 + a\frac{\underline{\varphi}(j)^T\underline{\varphi}(j)}{c+\underline{\varphi}(j)^T\underline{\varphi}(j)}\right]$ will never be zero. Therefore, using summation of Eq. (5.21), it can be concluded that

$$\lim_{j\to\infty} \frac{\underline{\tilde{\theta}}_{a,i}(j-1)^T\underline{\varphi}(j)\underline{\varphi}(j)^T\underline{\tilde{\theta}}_{a,i}(j-1)}{c+\underline{\varphi}(j)^T\underline{\varphi}(j)} = 0$$

which leads to Eq. (5.22)

$$\lim_{j\to\infty} \frac{\underline{\varphi}(j)^T \underline{\tilde{\theta}}_{a,i}(j-1)}{\left[c+\underline{\varphi}(j)^T \underline{\varphi}(j)\right]^{1/2}} = 0 \tag{5.22}$$

Recall that the tracking error at addressed time steps is defined as: $e_{a,i}(j) = y_{a,i}(j) - y^*_{a,i}(j) = -\underline{\varphi}(j)^T \underline{\tilde{\theta}}_{a,i}(j-1)$. Substituting this expression of addressed tracking error into Eq. (5.22) gives Eq. (5.23):

$$\lim_{j\to\infty} \frac{-e_{a,i}(j)}{\left[c+\underline{\varphi}(j)^T \underline{\varphi}(j)\right]^{1/2}} = 0 \tag{5.23}$$

and Eq. (5.23) produces Eq. (5.24)

$$\lim_{j\to\infty} \frac{e^2_{a,i}(j)}{c+\underline{\varphi}(j)^T \underline{\varphi}(j)} = 0 \tag{5.24}$$

Now it is clear that Eq. (5.24) is in the format of

$$\lim_{j\to\infty} \frac{s^2(j)}{b_1(j)+b_2(j)\underline{\sigma}(j)^T \underline{\sigma}(j)} = 0$$

where $b_1(j) = c$, $b_2(j) = 1$, $s(j) = e_{a,i}(j)$, and $\underline{\sigma}(j) = \underline{\varphi}(j)$ corresponding to the expression by the Lemma in Goodwin et al. (1980). Obviously, the uniform boundedness condition $0 < b_1(j) < K < \infty$, $0 \le b_2(j) < K < \infty$ is satisfied.

5.6.3 Linear Boundedness Condition

The aim of this subsection is to illustrate how stable inverse theory Longman JiLLL NS can enable the linear boundedness condition to be satisfied.

By assumption, the real-world system itself is stable, i.e., no poles located outside the unit circle in discrete time, so Eq. (5.25) holds, where m_{11} and m_{21} are of finite value

$$|y_{k1}(j-1)| \leq m_{11} + m_{21} \max_{1\leq\tau\leq p} \|u_\tau(j-1)\| \text{ for all } 1 \leq k1 \leq p \tag{5.25}$$

The purpose of requiring a stable inverse is to ensure the obtained control action to produce the desired output is bounded. As shown in Eq. (5.13), the control action is computed from the estimated model, so the inverse problem is only involved in the estimated model but not in the real-world unknown system. With the in-house developed stable inverse theory, the inverse of the estimated model will have a stable inverse, which means there exists m_{12} and m_{22} of finite value that ensure

$$|u_{k2}(j)| \leq m_{12} + m_{22} \max_{1\leq\tau\leq p/2} \left\|\hat{y}_{a,\tau}(j)\right\| \text{ for all } 1 \leq k2 \leq p \tag{5.26}$$

where $\hat{y}_{a,\tau}(j)$ is the output of the model, and, as in Eq. (5.13), it has been set equal to $y^*_{a,i}(j)$ to compute the required input control sequence $\{\underline{u}(j)\}$.

Yet the proof above only shows the linear boundedness condition is satisfied in one single batch, and the linear boundedness condition for batch to batch updates is still under investigation. Once this becomes available, the addressed tracking error $\lim_{j\to\infty} e_{a,i}(j) = y_{a,i}(j) - y^*_{a,i}(j) = 0$ can be easily proved followed by that $\{\|\underline{\varphi}(j)\|\}$ is bounded. Noting that the boundedness of $\{\|\underline{\varphi}(j)\|\}$ ensures the boundedness of the output $\{|\underline{y}(j)|\}$ and input $\{|\underline{u}(j)|\}$.

5.7 Numerical Results

Numerical experiments to validate and compare the performances of the generalized one step ahead control and batch stable inverse indirect adaptive control are conducted on the third-order real world system whose continuous time transfer function is $G(s) = (\frac{a}{s+a})(\frac{\omega_0^2}{s^2+2\xi\omega_0 s+\omega_0^2})$

with $a = 8.8$, $\omega_0 = 37$, and $\xi = 0.5$. It is then sampled at 100 Hz into discrete time domain, and the desired trajectory to be tracked is a unit magnitude sine wave of 4 Hz.

First of all, the smallest p needs to be determined from the stability condition in Eq. (5.12) to guarantee stability. Numerical results show that when p=6, all the eigenvalues of matrix are within the unit circle, which is a satisfyingly small number and makes the generalized one step ahead control close to the original one step ahead control. After figuring out that p should be no smaller than 6, the performance of the generalized one step ahead control is evaluated using p=8. As presented in Fig. 46, tracking errors are at the level of 10^{-14}, confirming that zero tracking error has been achieved at all the addressed time steps from the very beginning. And the tracking error at the unaddressed time steps presented in Fig. 47 has an RMS of 0.0098, which is an acceptably small error.

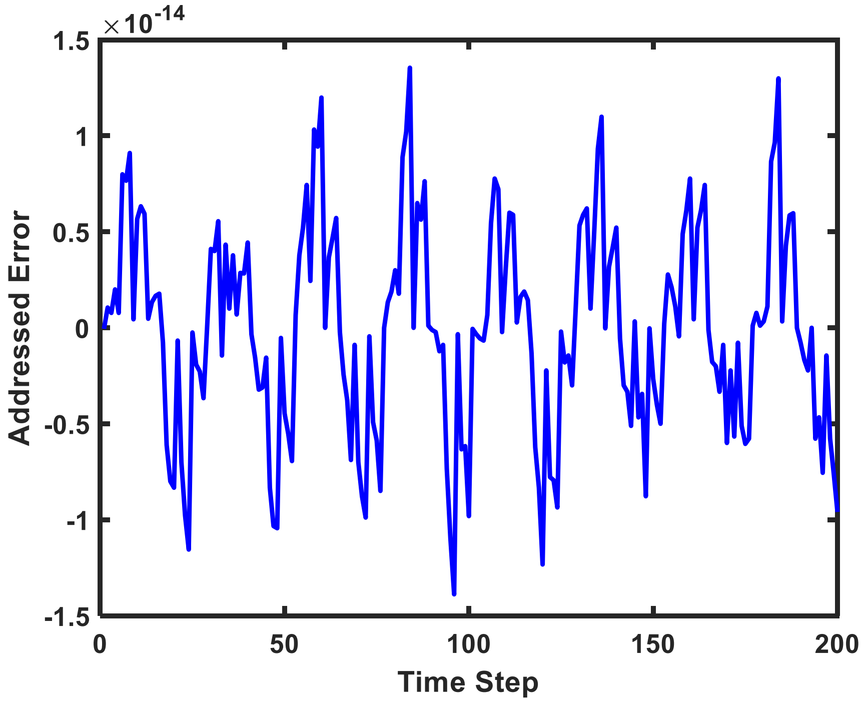


Figure 46. Tracking error at addressed time steps for JiLLL NS using *p* = 8 and 4Hz desired trajectory using generalized one step ahead control.

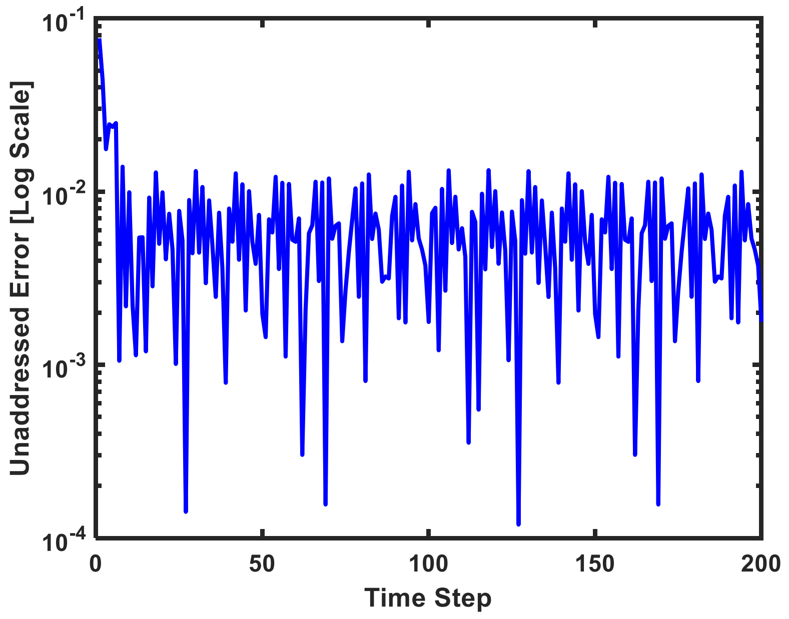


Figure 47. Tracking error at unaddressed time steps for JiLLL NS using *p* = 8 and 4Hz desired trajectory using generalized one step ahead control.

To see the performance differences between the generalized one step ahead control and batch stable inverse indirect adaptive control, the initial estimated model is biased from the real world system, and the estimated model has a continuous time transfer function of $\hat{G}(s) = (\frac{\hat{a}}{s+\hat{a}})(\frac{\hat{\omega}^2}{s^2+2\hat{\xi}\hat{\omega}s+\hat{\omega}^2})$ with $\hat{a} = 10$, $\hat{\omega} = 40$, and $\hat{\xi} = 0.9$. It is then also sampled at 100 Hz. The performance of the batch stable inverse indirect adaptive control in tracking the same unit magnitude sine wave of 4 Hz using *p*=10 is presented in Fig. 48 and 49. Unlike the generalized one step ahead control presented before, the batch stable inverse indirect adaptive control achieves zero tracking error at addressed time steps gradually, as illustrated in Fig. 48. And the tracking error at the unaddressed time steps presented in Fig. 49 converges at a level between 0.01 and 0.001, which is also at an acceptable level.

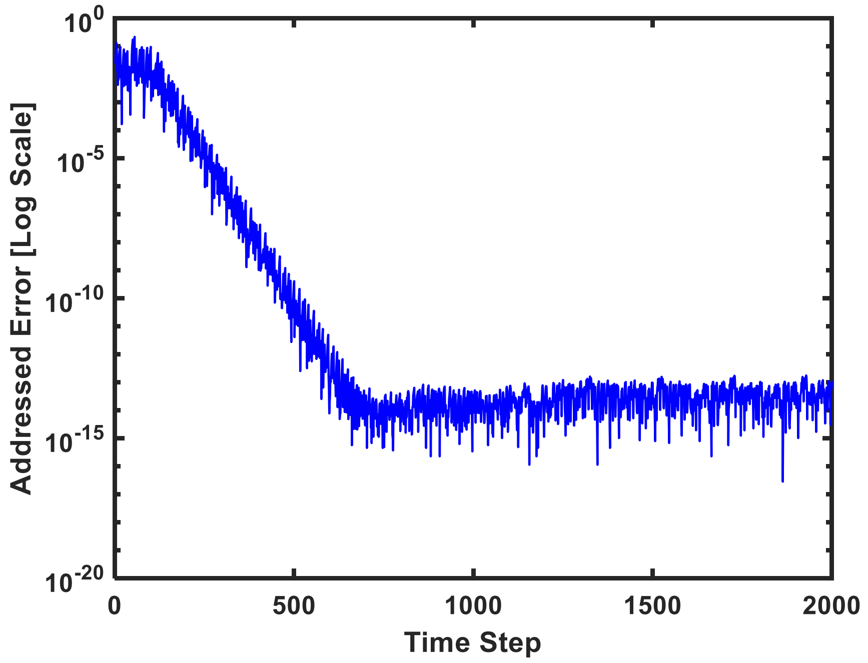


Figure 48. Tracking error at addressed time steps for JiLLL NS using *p* = 10 and 4Hz desired trajectory using batch stable inverse indirect adaptive control.

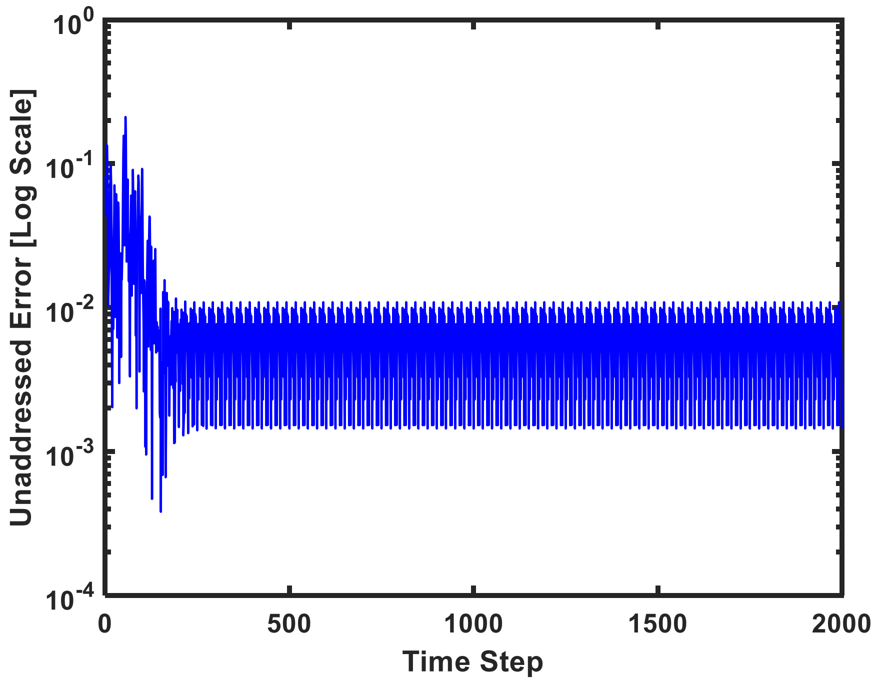


Figure 49. Tracking error at unaddressed time steps for JiLLL NS using *p* = 10 and 4Hz desired trajectory using batch stable inverse indirect adaptive control.

The numerical experiments above is consistent with the mathematical proof presented in the previous sections, confirming that both generalized one step ahead control and batch stable inverse indirect adaptive control give satisfying performances in trajectory tracking.

5.8 Conclusions

One step ahead control and indirect adaptive control have been of limited applications because they fail to track a chosen trajectory with a bounded and feasible control action sequence for discrete time systems with zeros outside the unit circle. The majority of physical systems usually have pole excess of 3 or more in continuous time, and, for reasonable sampling rates, they will have zeros outside the unit circle introduced during the process of discretization. These zeros become poles located outside the unit circle when one inverts the problem and tries to solve for the control action sequence needed to produce a known output trajectory, resulting in an issue of unstable inverse.

With the in-house developed stable inverse theory Longman JiLLL, one step ahead control, the fundamental part of indirect adaptive control, has been generalized into batch stable inverse control, where batch size *p* could be a very small number tending towards one and it is determined by the stability condition. When the model for computing the control action is of poor accuracy, or when it is preferred to improve the estimated model, batch stable inverse indirect adaptive control is applied to correct the model with real time input and output data while still achieving zero tracking error quickly.

Numerical results, consistent with the mathematics proof, show that the batch stable inverse control (also termed as generalized one step ahead control) can achieve zero tracking error immediately with a bounded control action input in tracking a chosen trajectory. Batch stable inverse indirect adaptive control, on the other hand, can gradually achieve zero tracking error while improving the estimated model.

In this chapter, it is the parameters in MPC format are estimated and updated without taking advantage of the original six-coefficient difference equation. In the future work, the parameters in the original third-order difference equation are expected be estimated and updated directly, which could reduce the computation burden involved in parameter update.

Chapter 6 Conclusions

Iterative Learning Control (ILC) aims to produce high precision tracking of a trajectory learning from repeating trials. After each trial, the command is updated based on the previous recorded error, aiming to converge to zero error. For long trajectories or with fast sample rates, the ILC computation and memory needs can be substantial as there are many time steps in one iteration. This burden can be reduced using basis functions spanning the most important error components and asking these components to converge to zero. The advantages include reduced computation, avoiding the common difficulty of an unstable inverse for many digital systems, avoiding difficulty from unmodeled high frequency dynamics, and avoiding the need for a zero-phase low-pass filter for robustification.

The ILC problem asks for zero tracking error every time step in a finite time problem, hence transients are expected in the output. Chapter 2 develops ILC making use of matched input-output basis functions for fast learning and a new kind of basis function designed to quickly eliminate error components related to transients. This can further reduce computation by producing a reduced error for the same number of basis functions applied, and can also help with tracking of the finite time signal when the desired trajectory is largely within the system settling time. Since transients are not part of the forced response, the new kind of basis functions developed here are matching initial conditions to transients. It is shown that if the desired finite time trajectory has a smooth transition of all derivatives at the start of the trajectory from before the initial time and then entering the desired trajectory, these new basis functions are not needed. The conditions under which they are needed are analyzed, and methods are developed to ensure feasibility of the learning process in such cases.

Yet basis function ILC restricts the input to a subspace, and aims for zero error in a subspace of the output, leaving the remaining error in the unaddressed output space. Chapter 3 identifies a potentially serious issue, that while the ILC is converging to zero error in the chosen or addressed part of the output error space, error can be accumulating in the unaddressed part of the space. A formula is derived to analyze the accumulation, and give the final value of the error. A method to pick the basis functions is presented that can avoid this accumulation. The concept of matched basis functions is presented, and if the model used is correct, there is no accumulation. The basis functions are chosen from the input and the output singular vectors of the singular value decomposition of the input-output matrix of Markov parameters. These basis functions are orthogonal and mapped one to one. They are related to the system frequency response which helps guide the designer in the choice of which singular vectors to include. The design can be made by finding the Markov parameters using the OKID algorithm directly from data, and there is no need to identify a transfer function model.

For a majority of desired applications, there are zeros introduced outside the unit circle producing instability of the inverse. In addition to using basis functions, another way to address the unstable inverse issue is to use an in-house developed stable inverse theory Longman JiLLL, which has not only been incorporated into ILC, but also into other controls such as Linear Model Predictive Control, One Step Ahead Control, and Indirect Adaptive Control.

Typical feedback control systems make no attempt to produce zero error tracking the desired trajectory. On the other hand, one step ahead control is a digital feedback control law that aims to produce zero error every time step. However, this control law is not practical and has been of limited use in real world applications for two reasons. First it is inverting a discrete time transfer function model of the system differential equation, and for any system with a pole excess of three or more, this produces an unstable control action for reasonable sample rates. And second, it makes no attempt to address actuator saturation limits. Chapter 4 addresses the first shortcoming by making use of two versions of Longman JiLLL and creates a *p*-step batch inverse control that avoids instability. To address the second concern, the approach is coupled with a quadratic cost whose control penalty is tuned to meet saturation limits every *p* time steps, converging to zero tracking error when the desired trajectory is feasible. A stability criterion is derived that establishes the number of time steps *p* needed to make a stable feedback control law. Numerical examples are presented showing that the methods give zero tracking error, and that the batch size *p* of time steps ahead can be quite small, making the generalized one step ahead control close enough to the original one step ahead control. The approach requires increasing the sample rate to produce the equivalent of a generalized hold between the original sample times, and the tracking error between the original sample times is studied. It is not zero as for the original time steps, but it can be quite small. Also, coupling with the quadratic cost produced fast convergence to zero tracking error when actuator limits are initially limiting the control action based on the initial tracking error. This generalized version of one step ahead control can also be viewed as a Model Predictive Control, updating the control batch by batch every *p* time steps, that can achieve zero tracking error at all the original time steps with input bounded by the actuator limits.

Indirect discrete time adaptive control creates a control law that promises to converge to zero tracking error for any commanded output. A major limitation is that the theory only guarantees convergence if the system inverse is asymptotically stable. Because it is necessarily discrete time, the continuous time governing equation must be converted to the equivalent difference equation, and this conversion normally introduces zeros. The basic indirect adaptive control relies on the one step ahead control law followed by the projection algorithm to update the model based on the current inverse. Chapter 5 presents generalized versions of these building blocks that address the unstable inverse issue. The one step ahead control is replaced by a *p*-step batch inverses control law as developed in Chapter 4 where the number of steps is chosen to produce stability of the inverse according to a new stable inverse theory. The needed computations to form the projection algorithm in this new formulation are presented.

References

1. H.S. Ahn, Y. Chen, and K.L. Moore, "Iterative Learning Control: Brief Survey and Categorization." *IEEE Transactions on Systems Man and Cybernetics, Part C Applications and Reviews*. Vol. 37, No. 6, 2007, p. 1099-1121.

2. E.S. Ahn, R.W. Longman, J.J. Kim, and B.N. Agrawal, "Evaluation of Five Control Algorithms for Addressing CMG Induced Jitter on a Spacecraft Testbed." *The Journal of the Astronautical Sciences.* Vol. 60, Issue 3, 2015, pp. 434-467.

3. S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering Operation of Robots by Learning." *Journal of Robotic Systems*. Vol. 1, No. 2, 1984, pp. 123–140.

4. K.J. Åström, P. Hagander, and J. Strenby, "Zeros of Sampled Systems." *Proceedings of the 19th IEEE Conference on Decision and Control*, 1980, pp. 1077-1081.

5. K.J. Åström, P. Hagander, and J. Sternby, "Zeros of Sampled Systems," *Automatica*, Vol. 20, No. 1, 1984, pp. 31-38.

6. K.E. Avrachenkov, H.S. Beigi, and R.W. Longman, "Updating Procedures for Iterative Learning Control in Hilbert Space." *Intelligent Automation and Soft Computing Journal (Special Issue on Learning and Repetitive Control)*, Vol. 8, No. 2, 2002, pp. 183-189.

7. J. Bao, and R.W. Longman, "Enhancements of Repetitive Control Using Specialized FIR Zero-Phase Filter Designs (AAS 07-340)," *Advances in the Astronautical Sciences*, Vol. 129, No. 2, 2008, p.1413.

8. J. Bao, and R.W. Longman, "Unification and Robustification of Iterative Learning Control Laws," *Advances in the Astronautical Sciences*, Vol. 136, 2010, pp.727-745.

9. H.S. Beigi, "A Parallel Network Implementation of the Generalized Secant Learning-Adaptive Controller," *Proceedings of Canadian Conference on Electrical and Computer Engineering*, Toronto, Canada, Sep. 13-16, 1992, Vol. 2, pp. 1.1-4.

10. H.S. Beigi, "An Adaptive Control Scheme Using the Generalized Secant Method." *Proceedings of Canadian Conference on Electrical and Computer Engineering*, Toronto, Canada, Sep. 13-16, 1992, Vol. 2, pp. 21.1-4.

11. H.S. Beigi, "New Adaptive and Learning-Adaptive Control Techniques Based on an Extension of the Generalized Secant Method," *Intelligent Automation and Soft Computing*, Vol. 3, No. 2, 1997, pp. 171-184.

12. H.S. Beigi, C.J. Li, and R.W. Longman, "Learning Control Based on Generalized Secant Methods and Other Numerical Optimization Methods," *Sensors, Controls, and Quality Issues in Manufacturing,* ASME: Atlanta, Vol. 55, December 1991, pp. 163-175.

13. Z. Bien, and J.-X. Xu, *Iterative Learning Control, Analysis, Design, Integration and Applications*. Kluwer Academic Publishers, 1998.

14. D.A. Bristow, M. Tharayil, and A.G. Alleyne, "A Survey of Iterative Learning Control." *IEEE Control Systems Magazine*. Vol. 26, No. 3, 2006, pp. 96–114.

15. F. Cameron, D.E. Seborg, "A self-tuning controller with a PID structure. Real Time Digital Control Application." *Pergamon*, 1984, pp. 613-622.

16. H.-J. Chen, R.W. Longman, and B.N. Agrawal, "Approaches to Matched Basis Function Repetitive Control." *Advances in the Astronautical Sciences*. Vol. 109, 2002, pp. 931-950.

17. K. Chen, and R.W. Longman, "Creating A Short Time Equivalent of Frequency Cutoff for Robustness in Learning Control." *Advances in the Astronautical Sciences*, Vol. 114, 2003, pp. 95-114.

18. K. Chen, R.W. Longman, and M.Q. Phan, "On the Relationship Between Repetitive Control and Model Predictive Control," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2006, p. 6525.

19. S. Devasia, D. Chen, and B. Paden, "Nonlinear Inversion-Based Output Tracking." *IEEE Transactions on Automatic Control*, Vol. 41, No. 7, 1996, pp. 930-942.

20. S.G. Edwards, B.N. Agrawal, M.Q. Phan, and R.W. Longman, "Disturbance Identification and Rejection Experiments on an Ultra Quiet Platform." *Advances in the Astronautical Sciences*. Vol. 103, 1999, pp. 633-651.

21. J.S. Freudenburg, R. H. Middleton, and J. H. Braslavsky, "Robustness of Zero Shifting via Generalized Sampled-Data Hold Functions." *IEEE Transactions on Automatic Control*, Vol. 42, No. 12, 1997, pp. 1681-1692.

22. J.A. Frueh, and M.Q. Phan, "Linear Quadratic Optimal Learning Control (LQL)." *International Journal of Control*. Vol. 73, No. 10, 2000, pp. 832-839.

23. G.C. Goodwin, P.J. Ramadge, and P.E. Caines, "Discrete-Time Multivariable Adaptive Control." *IEEE Transactions on Automatic Control*, Vol. AC-25, No. 3 June 1980, pp. 449-456.

24. G.C. Goodwin, and K.W. Sin, *Adaptive Filtering Prediction and Control*, Prentice Hall, NJ, 1984.

25. A. Ibeas, M. de la Sen, P. Balaguer, R. Vilanova, C. Pedret, "Digital Inverse Model Control using Generalized Holds with Extension to the Adaptive Case." *International Journal of Control, Automation and Systems*, Vol 8, No 4, 2010, pp. 707-719.

26. T. Inoue, M. Nakano, T. Kubo, S. Matsumoto, and H. Baba, "High Accuracy Control of a Proton Synchrotron Magnet Power Supply." *IFAC Proceedings*, Vol. 14, Issue 2. 1981, pp. 3137-3142.

27. X. Ji, T. Li, and R. W. Longman, "Proof of a New Stable Inverse of Discrete Time Systems." *Proceedings of the AAS/AIAA Astrodynamics Conference*, Stevenson, WA, August 20-23, 2017.

28. X. Ji, and R.W. Longman, "New Results for Stable Inverses of Discrete Time Systems." *Proceedings of the 19th Yale Workshop on Adaptive and Learning Systems*, Narendra Editor, June 2019.

29. P. Kabamba, "Control of Linear Systems using Generalized Sampled-Data Hold Functions." *IEEE Transactions on Automatic Control*, Vol. 32, No. 9, 1987, pp. 772-783.

30. C. Kempf, W.C. Messner, M. Tomizuka, and R. Horowitz, "A comparison of Four Discrete-Time Repetitive Control Algorithms." *Proceedings of the 1992 American Control Conference*, Chicago, Illinois. 1992, pp. 2700-2704.

31. P.A. LeVoci, and R.W. Longman, "Intersample Error in Discrete Time Learning and Repetitive Control." *Proceedings of the 2004 AIAA/AAS Astrodynamics Specialist Conference*, Providence, RI, August 2004.

32. C.J. Li, H.S. Beigi, S. Li, and J. Liang, "Nonlinear Piezo-Actuator Control by Learning Self-Tuning Regulator," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 115, No. 4, December 1993, pp. 720-723.

33. T. Li, and R. W. Longman, "Designing Iterative Learning Control of Non-Minimum Phase Systems to Converge to Zero Tracking Error." *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference*, Long Beach, CA, September 2016.

34. Y. Li, and R. W. Longman, "Characterizing and Addressing the Instability of the Control Action in Iterative Learning Control." *Advances in the Astronautical Sciences*, Vol. 136, 2010, pp. 1967-1985.

35. R.W. Longman, "Iterative Learning Control and Repetitive Control for Engineering Practice." *International Journal of Control*, Vol. 73, No. 10, 2000, pp. 930-954.

36. R.W. Longman, "On the Theory and Design of Linear Repetitive Control Systems." *European Journal of Control*, Special Section on Iterative Learning Control, Guest Editor Hyo-Sung Ahn. Vol. 16, No. 5, 2010, pp. 447-496.

37. R.W. Longman, R. Akogyeram, A. Hutton, and J.-N. Juang, "Stability of Matched Basis Function Repetitive Control." *Advances in the Astronautical Sciences*. Vol. 105, 2000, pp. 33-52.

38. R.W. Longman, and T. Li, "On a New Approach to Producing a Stable Inverse of Discrete Time Systems." *Proceedings of the 18th Yale Workshop on Adaptive and Learning Systems*, Narendra Editor, June 2017.

39. R.W. Longman, J.-N. Juang, and M.Q. Phan, "On the Ill-Conditioning in MPC when used to Address the Repetitive Control Problem." *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2008, p. 6452.

40. R.H. Middleton, G.C. Goodwin, and R.W. Longman, "A Method for Improving the Dynamic Accuracy of a Robot Performing a Repetitive Task." *International Journal of Robotics Research*, Vol. 8, No. 5, October 1989, pp.67-74.

41. M. Nagashima, and R.W. Longman, "The Effect of Averaging in Matched Basis Function Real Time Repetitive Control." *Advances in the Astronautical Sciences*. Vol. 114, 2003, pp. 75-94.

42. M. Nagashima, and R.W. Longman, "Stability and Performance Analysis of Matched Basis Function Repetitive Control in the Frequency Domain." *Advances in the Astronautical Sciences*. Vol. 119, 2005, pp. 1581-1600.

43. K.S. Narendra, and A.M. Annaswamy, *Stable Adaptive Systems*, Courier Corporation, 2012.

44. T. Omata, M. Nakano, and T. Inoue, "Applications of Repetitive Control Method to Multivariable Systems." *Transactions of SICE*. Vol. 20, No. 9, 1984, pp. 795-800.

45. B. Panomruttanarug, and R.W. Longman, "Repetitive Controller Design Using Optimization in the Frequency Domain," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2004, p. 5299.

46. M.Q. Phan, and J.A. Frueh, "Learning Control for Trajectory Tracking using Basis Functions." *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, December 1996.

47. M.Q. Phan, and J.A. Frueh, "Model Reference Adaptive Learning Control with Basis Functions," *Proceedings of the 38th IEEE Conference on Decision and Control* (Cat. No. 99CH36304). Vol. 1, 1999, pp. 251-257.

48. Y. Shi, R.W. Longman, and M. Nagashima, "Small Gain Stability Theory for Matched Basis Function Repetitive Control." *Acta Astronautica*. Vol. 95, 2014, pp. 260-271.

49. M. Tomizuka, T.-C. Tsao, and K.-K. Chew, "Analysis and Synthesis of Discrete-Time Repetitive Controllers." *Journal of Dynamic Systems, Measurement, and Control*. Vol. 111, 1989, pp. 353-358.

50. M. Uchiyama, "Formation of High-Speed Motion Pattern of a Mechanical Arm by Trial (in Japanese)." *Transactions of the Society of Instrument and Control Engineers.* Vol. 14, 1978, pp. 706–712.

51. B. Wang, and R.W. Longman, "Generalized One Step Ahead Control Made Practical by New Stable Inverses." *Proceedings of the AIAA/AAS Space Flight Mechanics Meeting*, 2018, p. 0203.

52. B. Wang, and R.W. Longman, "On the Development of Batch Stable Inverse Indirect Adaptive Control of Systems with Unstable Discrete-Time Inverse." *Proceedings of the 7th International Conference on High Performance Scientific Computing*, Hanoi, Vietnam (2018).

53. B. Wang, R.W. Longman, and M.Q. Phan, "Development of Transient Basis Functions to Improve Basis Function Iterative Learning Control." *Advances in the Astronautical Sciences*, Vol. 167, 2019, pp. 2599-2615.

54. B. Wang, R.W. Longman, and M.Q. Phan, "Basis Function Iterative Learning Control: Limiting Accumulation in Unaddressed Error Space with Singular Vector Basis Functions." *Advances in the Astronautical Sciences*, 2019, pp. 1651-1668.

55. H.-P. Wen, M.Q. Phan, and R.W. Longman, "Bridging Learning Control and Repetitive Control using Basis Functions." *Advances in the Astronautical Sciences*. Vol. 99, 1998, pp. 335-354.

56. Q. Zou, and S. Devasia, "Preview-Based Stable-Inversion for Output Tracking of Linear Systems." *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 12, 1999, pp. 625-630.

57. J. Zhu, and R.W. Longman, "Repetitive Model Predictive Controller Design Based on Markov Parameters." *Advances in the Astronautical Sciences*, 2017, pp. 2819-2830.