10-1-2008

High Bandwidth Control of Precision Motion Instrumentation

Douglas A. Bristow
Missouri University of Science and Technology, dbristow@mst.edu

Jingyan Dong

Andrew G. Alleyne

Srinivasa M. Salapaka

et. al. For a complete list of authors, see http://scholarsmine.mst.edu/mec_aereng_facwork/3437

Recommended Citation

D. A. Bristow et al., "High Bandwidth Control of Precision Motion Instrumentation," *Review of Scientific Instruments*, American Institute of Physics (AIP), Oct 2008.
The definitive version is available at https://doi.org/10.1063/1.2980377

High bandwidth control of precision motion instrumentation

Douglas A. Bristow,[1] Jingyan Dong,[2] Andrew G. Alleyne,[2] Placid Ferreira,[2] and Srinivas Salapaka[2]
[1]*Department of Mechanical and Aerospace Engineering, Missouri University of Science and Technology, Rolla, Missouri 65401, USA*
[2]*Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA*

(Received 4 June 2008; accepted 20 August 2008; published online 15 October 2008)

This article presents a high-bandwidth control design suitable for precision motion instrumentation. Iterative learning control (ILC), a feedforward technique that uses previous iterations of the desired trajectory, is used to leverage the repetition that occurs in many tasks, such as raster scanning in microscopy. Two ILC designs are presented. The first design uses the motion system dynamic model to maximize bandwidth. The second design uses a time-varying bandwidth that is particularly useful for nonsmooth trajectories such as raster scanning. Both designs are applied to a multiaxis piezoelectric-actuated flexure system and evaluated on a nonsmooth trajectory. The ILC designs demonstrate significant bandwidth and precision improvements over the feedback controller, and the ability to achieve precision motion control at frequencies higher than multiple system resonances. [DOI: 10.1063/1.2980377]

I. INTRODUCTION

Many modern scientific instruments rely on precise motion generation. This is especially true at the nanoscale and below, where precision motion systems (PMSs) are used for microscopy,[1–4] manipulation,[5–7] assembly,[8,9] and information storage.[10] For precise motion at the submicron scale, actuation is typically provided by piezoelectric crystals and transmitted through flexure linkages to the tool, which is often a cantilevered probe. Sensing can be accomplished through several technologies including capacitance sensors, linear variable differential transform sensors, and the piezoelectric actuators. The controller is an algorithm, usually implemented on a digital microprocessor, that uses incoming sensed information to determine the appropriate outgoing actuation signal. A schematic of the complete PMS is shown in Fig. 1.

Primarily, the goal for the PMS is high precision with high bandwidth. High bandwidth is especially important for nonsmooth motions, such as the normalized triangle wave shown in Fig. 2. The triangle wave is important in microscopy since it is used by one axis to create the raster scanning motion. To track nonsmooth motions, which are synonymous with high accelerations, the controlled system must be capable of very high bandwidth. As seen from Fig. 2, it may be necessary for the controlled system to have a bandwidth that is one or two orders of magnitude larger than the fundamental frequency to accurately track the triangle wave. When the bandwidth is too low, the system will not duplicate the commanded accelerations resulting in rounded corners or overshoot. In the case of microscopy, tracking error will appear as image artifacts.

Traditionally, proportional-plus-integral (PI) or proportional-plus-double-integral (PII) feedback controllers have been used to provide the motion control.[1] Although these controllers are effective for convergence to step or ramp trajectories, they may be poorly suited for tracking complex trajectories or during transient motions, such as the peaks and valleys of a triangle wave. The reason for this is that they are designed based on the low-frequency behavior of nanopositioning systems. At higher frequencies, resonant modes of the flexure systems become important, or even dominate the response. More complex $\mathcal{H}_\infty$-type feedback controllers, which can account for the resonant modes of the system, have demonstrated higher bandwidth and improved transient tracking over the PI or PII type controllers.[11] Although $\mathcal{H}_\infty$ feedback controllers have a higher bandwidth, they still operate well below the first system resonance. To provide high-bandwidth tracking that can approach or exceed the system resonance, a feedforward control is necessary.[12,13]

In many applications, such as raster scanning, the reference trajectory is repetitive. The repetition of the trajectory can be leveraged in a feedforward-type control called iterative learning control (ILC).[14–17] In ILC, the feedforward signal is generated and updated from previous passes of the trajectory. The performance of this scheme is less sensitive to model inaccuracies than standard feedforward control because it is updated using the actual system response rather than relying on the system model. The reduced model sensitivity translates to higher bandwidth. In Ref. 18 an ILC design is presented for hysteretic systems, which includes atomic force microscopes. Here, we treat hysteresis as a low-frequency disturbance and delegate hysteresis compensation to the feedback controller. The primary focus of our ILC design is to maximize bandwidth. To this end, we present a high-bandwidth ILC design. For nonsmooth trajectories, an advanced ILC design is also presented that uses knowledge of the trajectory to further improve precision. Both designs are tested on a multiaxis piezopositioning system and results

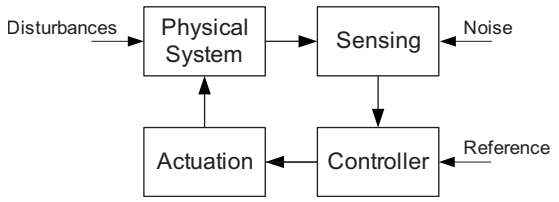

FIG. 1. PMS schematic.

show significant performance improvement versus feedback control and that precision tracking is capable beyond the first system resonance.

The remainder of this article is organized as follows. In Sec. II we describe the class of systems under consideration. An introduction to ILC is given in Sec. III. In Sec. IV, our high-bandwidth ILC design is presented. A second, advanced ILC design for nonsmooth trajectories is presented in Sec. V. Both designs are applied to a multiaxis piezopositioning system and evaluated for precision. Results are given in Sec. VI and conclusions are given in Sec. VII.

II. SYSTEM DESCRIPTION

We assume that the motion system can be described as a multi-input multioutput (MIMO) linear time-invariant (LTI) system

$$G(z) = \begin{bmatrix} G_{11}(z) & \cdots & G_{1n}(z) \\ \vdots & \ddots & \vdots \\ G_{n1}(z) & \cdots & G_{nn}(z) \end{bmatrix},$$

where $G_{ij}(z)$ is the discrete-time transfer function from the j^{th} control input to the ith axis position, n is the number of inputs and outputs, and z is complex. We use a discrete-time formulation because ILC requires measurement and storage of the output signals, which is done digitally. For serial kinematic systems, where each degree of freedom is primarily controlled by a single actuator, the diagonal elements of $G(z)$, $G_{ii}(z)$ for $i=1,\ldots,n$ provide the dominant response and off-diagonals $G_{ij}(z)$, where $i \neq j$, are comparatively small. It should be noted that at high frequencies it is not uncommon for serial mechanisms to display significant coupling. In parallel kinematic systems, such as the system discussed in Sec. VI, multiple inputs affect each output creating highly coupled dynamics. For the system used in Sec. VI, three piezoactuator inputs (A, B, and C) actuate the system, which is measured in standard Cartesian output coordinates (X, Y, and Z). The Bode plot for this is shown in Fig. 3, from which the coupling can be clearly seen.

A. Frequency regions

In the following we define three frequency regions common in nanopositioning systems. These regions are important for our discussion and control design because they provide a context for the limitations and challenges in control design. They also provide a means of normalizing across the spectrum of nanopositioning systems. Consider the transfer function from actuator UA to the Y axis of the system in Fig. 3 given by $G_{21}(z)$ (shown in greater detail in Fig. 4). For the purposes of this discussion, we can divide the Bode plot into

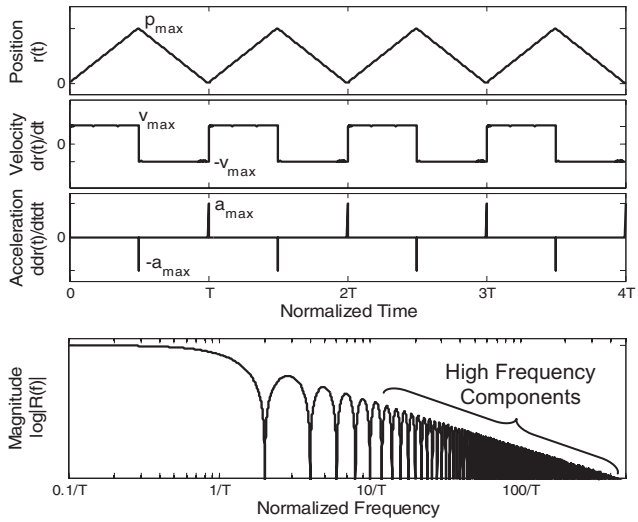

FIG. 2. Triangle trajectory (e.g., in scanning applications) with period T and range $[0, P_{\max}]$. Velocity is $\pm V_{\max}=2P_{\max}/T$ and accelerations are impulses (for digital sources, peak acceleration is $a_{\max}=2V_{\max}/t_s$, where t_s is the sample period). Trajectories with rapid velocity changes have high-frequency components and require high bandwidth for accurate tracking.

three regions. In region I the flexure system is stiff enough to quickly transmit the piezoactuator's motion to the tool. High stiffness and low-frequency piezoproperties give this region a characteristically flat response. In region II, resonant modes of the flexure system appear resulting in lightly damped poles and zeros. Region III may also have some resonant modes but has characteristically low gain due to intertia-based "roll off."

Region I is the typical region of operation for these systems since the flat response makes it relatively easy to control. Control in region II is difficult because the magnitude and phase change rapidly over short frequency ranges which can cause instability in feedback control for even small inaccuracies in the model. Control in region III is impractical and inefficient because very large control signals are necessary for comparatively small motions. Because of the potential for instability with feedback control in region II, this is primarily a domain for feedforward control.[12]

B. Feedback control

The feedback controller is assumed to be an LTI filter

$$K(z) = \begin{bmatrix} K_{11}(z) & \cdots & K_{1n}(z) \\ \vdots & \ddots & \vdots \\ K_{n1}(z) & \cdots & K_{nn}(z) \end{bmatrix},$$

where $K_{ij}(z)$ is the controller from the jth axis to the ith actuator. The feedback controlled system, shown in Fig. 5, can be designed using a variety of methods including $\mathcal{H}_\infty$-design.[11] In Fig. 5, k is the discrete-time index, $r \in \mathbb{R}^n$ is the reference, $e \in \mathbb{R}^n$ is the error, $u \in \mathbb{R}^n$ is the control, and $y \in \mathbb{R}^n$ is the position output. When z and k are mixed in equations or figures, such as Fig. 5, z should be treated as the time-shift operator, defined by $zx(k) \triangleq x(k+1)$ and $z^{-1}x(k) \triangleq x(k-1)$.

In $\mathcal{H}_\infty$-design the goal is to simultaneously optimize competing objectives of reference-to-error sensitivity

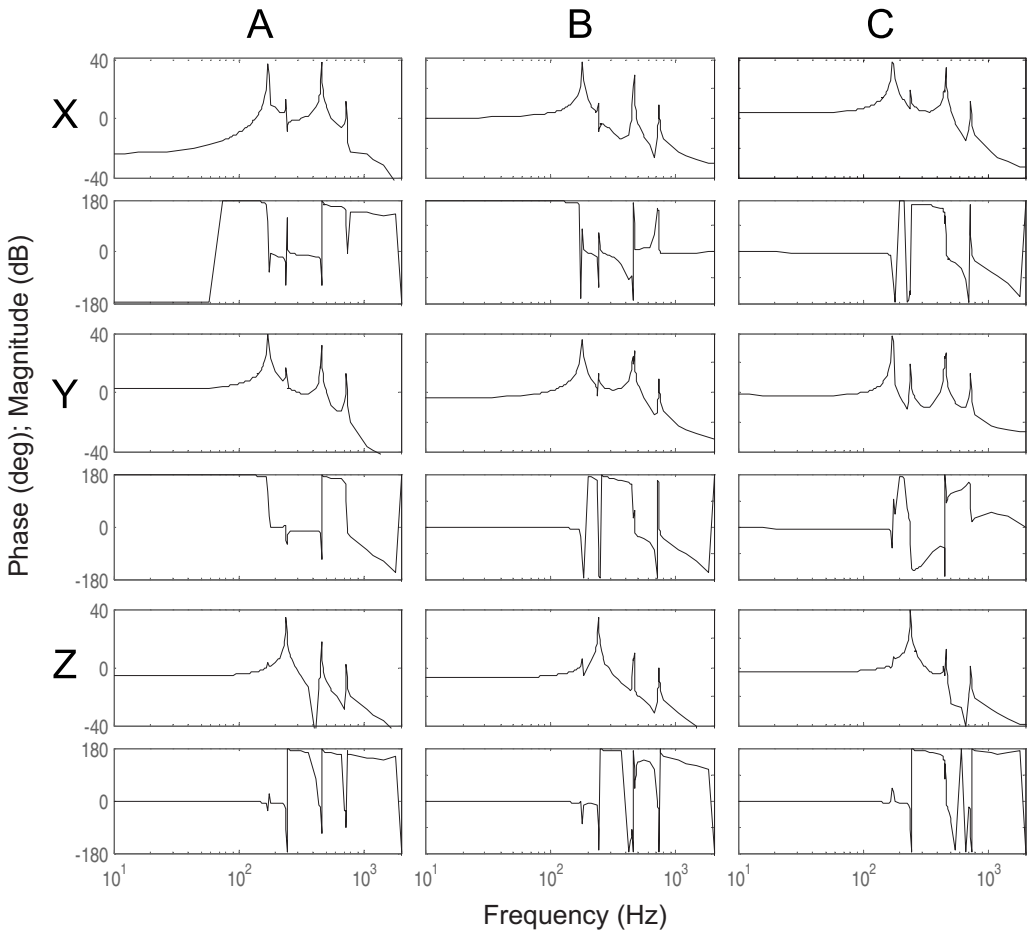

FIG. 3. Bode plot of PKM dynamics (Ref. 19). System inputs A, B, and C are piezoelectric actuators and system outputs X, Y, and Z are Cartesian coordinates. Dynamic coupling across the entire frequency spectrum can be seen in the off-diagonal blocks (X-B, X-C, Y-A, Y-C, Z-A, and Z-B).

$S(z) \triangleq [I+G(z)K(z)]^{-1}$, reference-to-output sensitivity (called complementary sensitivity) $T(z) \triangleq [I+G(z)K(z)]^{-1}G(z)K(z)$, and reference-to-control sensitivity $K(z)S(z)$. Good feedback design dictates that S be small at low frequencies for good tracking, T be small at high frequencies for noise-insensitivity, and KS be magnitude limited to prevent control saturation. To this end, $\mathcal{H}_\infty$ design uses the fact that it is numerically possible to find a $K(z)$ such that

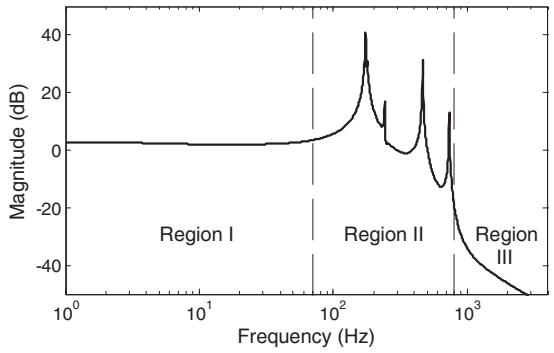

FIG. 4. Bode plot of transfer function from actuator A to the Y axis of the PKM showing three region characteristics to piezoelectric actuator flexure mechanisms. Low/midfrequency region I is flat, high-frequency region II contains resonant modes, and very-high-frequency region III has interia-based roll off.

$$\left\| \begin{bmatrix} W_1(z)S(z) \\ W_2(z)T(z) \\ W_3(z)K(z)S(z) \end{bmatrix} \right\|_\infty < \gamma_{\text{feedback}},$$

where $\gamma_{\text{feedback}} > \gamma_{\text{op}} > 0$, $\|\cdot(z)\|_\infty \triangleq \max_{\omega\in[-\pi,\pi]} \bar{\sigma}[\cdot(e^{i\omega})]$ is the $\mathcal{H}_\infty$-norm, $\bar{\sigma}$ is the maximum singular value, and γ_{op} is the optimal norm. The filters $W_1(z)$, $W_2(z)$, $W_3(z)$ are $n\times n$ LTI weighting functions designed to achieve the desired sensitivity trade-offs. The interested reader is referred to Refs. 11, 20, and 21 for details on weighting function design. Many numerical packages, such as MATLAB, contain built-in code to calculate an $\mathcal{H}_\infty$ controller.

C. Feedforward control

Here we consider two possible implementation structures for feedforward design, which we refer to as parallel and serial. In the parallel implementation the feedforward is added to the feedback control signal, and in the serial imple-

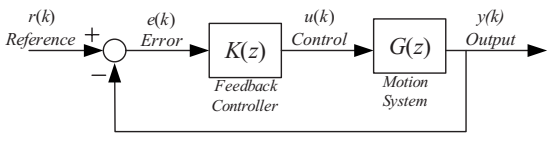

FIG. 5. Typical feedback controlled closed-loop system.

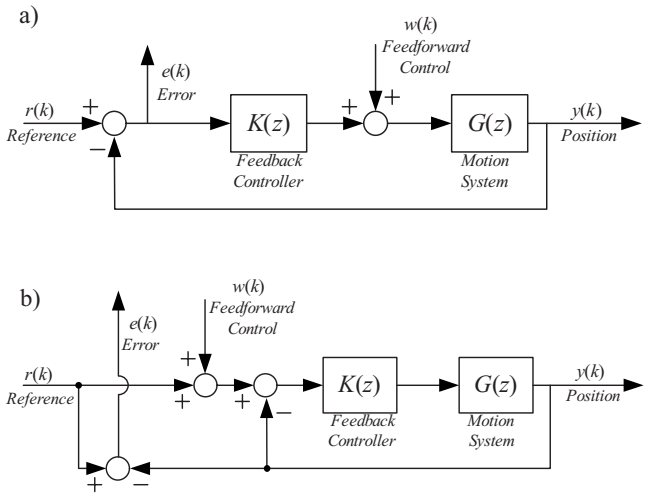

FIG. 6. (a) Parallel implementation and (b) serial implementation of feedforward control combined with feedback control. The feedforward control signal is often generated from the reference as $w(k)=F(z)r(k)$, where $F(z)$ is a designed filter.

mentation, the feedforward is added to the reference signal. Both implementations are illustrated in Fig. 6. The parallel implementation is useful because the feedforward design is largely independent of the feedback controller. This has advantages in situations where the feedback controller design is tuned frequently, or is complex, possibly containing nonlinear elements such as rate limiters. Serial implementation is more useful in systems with "closed architecture" control hardware, where the user does not have access to the control signals, but typically can modify the reference signal.

The tracking error is a combination of the feedback control and the feedforward control as

$$e(k) = \underbrace{S(z)r(k)}_{\text{feedback component}} - \underbrace{P(z)w(k)}_{\text{feedforward component}}, \quad (1)$$

where $P(z)=S(z)G(z)$ for the parallel system and $P(z)=T(z)$ for the serial system. In classical feedforward design,[12,13] the feedforward control $w(k)$ is generated from the reference, as $w(k)=F(z)r(k)$. The goal is to find the filter $F(z)$ that yields the smallest error. When $P(z)$ is invertible, it is easy to see that $F(z)=P^{-1}(z)S(z)$ gives zero error.

In practice, feedforward based on model inversion rarely achieves perfect tracking due to model inaccuracy. The feedforward controller may actually increase the error when the model is too inaccurate.[22] Best results are obtained when the feedforward controller is carefully designed to reduce sensitivity to inaccurate modes of the model by sacrificing performance. In Ref. 12, a finite-impulse-response (FIR) design for $F(z)$ is able to provide reference tracking beyond the first resonant mode of the system. Here, we use ILC in lieu of classical feedforward design, to generate the feedforward control.

III. ITERATIVE LEARNING CONTROL

ILC uses several iterations, or trials, of a process to automatically generate the feedforward control. In some processes, such as scanning, the motion commands may be naturally periodic. In this case the first few periods are treated as

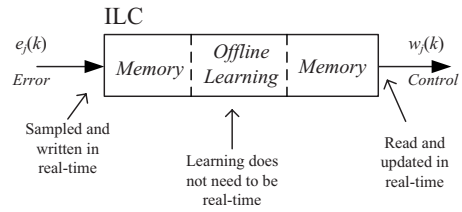

FIG. 7. ILC learning schematic. Error signal is written to memory during each iteration. Learning occurs offline, between iterations, and is stored in another memory location. A separate (possibly networked) system may be used for learning. During the next iteration, the new control is read from memory and applied to the system in real time.

learning iterations. In other processes, where there is no periodicity, several process iterations must first be run for learning. In this case the addition of learning iterations to the process may be a worthwhile trade for the improved precision.

ILC operates in the following manner. During each iteration, the tracking error is sampled and stored in memory. After the motion has completed, learning is performed in an offline mode using the tracking error stored in memory to generate a new feedforward control signal. Alternatively, instead of learning offline, a separate computer process thread can be initiated to perform the learning operation. After the new control signal is generated, it is stored in a second block of memory. On the following iteration, the new control signal is read from the memory and applied to the system in real time. This procedure is illustrated with the blocks in Fig. 7. We use the subscript j on signals to denote the signal from the jth iteration.

Because learning is performed offline, while the system is not in motion, it is possible to use high-order filters without computational concerns. Noncausal filters, which we discuss in Sec. IV, can also be used here. When the system is online and tracking, the only real-time operations required by the ILC are the sampling and storing of the error signal, and the reading and updating of the control signal. It is critical to note that processes learned by ILC must be identical from iteration to iteration. The control signal that is generated by ILC is specific to the reference signal used in the learning.

A. First-order ILC algorithm

The first-order ILC algorithm is given by

$$w_{j+1}(k) = w_j(k) + L(z)e_j(k), \quad (2)$$

where L is an $n\times n$ LTI filter called the learning filter, $j=0,1,\ldots$ is the iteration index, $k=0,1,\ldots,N-1$ is the time index, N is the iteration length. Typically, the ILC is initialized to zero for the initial pass, $w_0(k)=0$, so that the zeroth pass is controlled only by the feedback controller. The learning filter, L, determines how the error is used to change the control signal for the following iteration. In the simplest case, L could be a constant,[18] $L=\kappa$, where κ is units of control/units of error. A constant may work well for low frequencies where the piezo has a flat frequency response, but to appropriately learn at frequencies near or above the system resonances, a more sophisticated filter is necessary, as discussed in Sec. IV A.

No matter how sophisticated our filter is, however, there are typically some frequencies (especially high frequencies) where the proper learning filter design is difficult to obtain, generally because of model inaccuracy. With the practical limitation of robustness in mind, a modified first-order algorithm

$$w_{j+1}(k) = Q(z)I_{n\times n}[w_j(k) + L(z)e_j(k)], \tag{3}$$

where Q is a single-input single-output (SISO) low pass LTI filter on each control channel, is preferable to Eq. (2). Here Q is used to select the learning bandwidth. We proceed using the modified first-order algorithm.

Remark 1: In other ILC approaches[23–27] for nanopositioning systems, the ILC updating algorithm is calculated in the frequency domain, whereas here we consider it in time domain (3). Analysis and design are similar for the two approaches, although there is a significant difference in implementation. In frequency-domain updating the output signal of the system must be captured for an unactuated period before and after the trajectory.[24] In the time-domain updating used here, the output signal only needs to be captured during the trajectory, which uses less memory. Additionally, there are no actuation constraints before or after the trajectory in time-domain updating. Therefore, the system can be reset immediately upon completion of the trajectory, resulting in less downtime between iterations as compared to frequency-domain updating. Finally, frequency-domain updating limits the learning bandwidth by removing high-frequency terms.[24] In time-domain updating the Q-filter is used to limit the control bandwidth, which permits additional design flexibility that is leveraged in the advanced Q design in Sec. V.

B. ILC analysis

For the analysis in this section, we make the following assumptions:

(A1) the system G is LTI,
(A2) the ILC is implemented in parallel or serial with a feedback controller as in Fig. 7,
(A3) the disturbance signal and initial conditions are iteration invariant, and
(A4) the iteration length is infinitely long.

Although many nanopositioning stages are flexure based and thus have nonlinear dynamic components, the flexure legs are typically much longer than the motion range. Therefore, the dynamics can be accurately captured as a LTI system[28] in keeping with (A1). Piezononlinearities such as creep and hysteresis can be made small with a well designed feedback controller[11] and are here treated as external disturbances. (A3) can be relaxed to bounded disturbances and initial conditions. In this case, the ILC converges to a bounded region and asymptotic performance is contained in a bounded region.[29] In practice (A4) is never true, but is often assumed in ILC analysis to permit the use of frequency-domain tools.[17] (A4) can be relaxed to use the finite length, N, and time-domain analysis performed instead,[17] but the frequency-domain analysis is used here for simplicity.

1. Convergence

The closed-loop learning dynamics are obtained from Eqs. (1) and (3) as

$$w_{j+1}(k) = Q(z)[I - L(z)P(z)]w_j(k) + Q(z)L(z)S(z)r(k). \tag{4}$$

Taking the Fourier transform of Eq. (4) we can determine that learning converges if (details in Ref. 17)

$$\max_{\omega\in[0,\pi]} |Q(e^{i\omega})|\bar{\sigma}\{[I - L(e^{i\omega})P(e^{i\omega})]\} < 1. \tag{5}$$

If the ILC is convergent, then we can find the converged control as $w_\infty(k) = \lim_{j\to\infty} w_j(k)$, or

$$w_\infty(k) = \{I - Q(z)[I - L(z)P(z)]\}^{-1}Q(z)L(z)S(z)r(k). \tag{6}$$

Note that it is impractical to use Eq. (6) to precisely calculate $w_\infty(k)$ in practice because we never know $P(z)$ with perfect accuracy. However, Eq. (6) usually gives a good estimate.

2. ILC asymptotic performance

If the ILC converges, we can calculate the asymptotic tracking from Eqs. (1) and (6) as

$$e_\infty(k) = S(z)r(k) - P(z)w_\infty(k) = [I - P(z)\{I - Q(z)[I - L(z)P(z)]\}^{-1}Q(z)L(z)]S(z)r(k). \tag{7}$$

If $Q(z)=1$ and Eq. (5) is true, then Eq. (7) reduces to $e_\infty(k)=0$ and perfect tracking is achieved. Unfortunately, it is generally difficult to design $L(z)$ such that Eq. (5) is true when $Q(z)=1$ because of $P(z)$ model inaccuracy. Instead, we use a low pass filter for $Q(z)$ and, taking the Fourier transform for Eq. (7) we obtain

$$E_\infty(e^{i\omega}) = \{I - P[I - Q(I - LP)]^{-1}QL\}SR(e^{i\omega}),$$

where we have dropped the $e^{i\omega}$ argument on the systems for compactness. If our $Q(z)$ is an ideal low pass filter with $Q(e^{i\omega})=1$ for $\omega<\Omega$ and $Q(e^{i\omega})=0$ for $\omega>\Omega$, where Ω is the bandwidth, then

$$E_\infty(e^{i\omega}) = \begin{cases} 0, & \omega \le \Omega \\ E_0(e^{i\omega}), & \omega > \Omega, \end{cases}$$

where $E_0(e^{i\omega})=S(e^{i\omega})R(e^{i\omega})$ is the error on the initial pass. We conclude that the asymptotic tracking bandwidth for ILC is approximately the Q-filter bandwidth. For the best asymptotic performance, we would like our Q-filter bandwidth to be as large as possible, subject to stability constraint Eq. (5).

IV. HIGH-BANDWIDTH LEARNING ALGORITHM DESIGN

A. Learning filter L

Here, we design the learning filter $L(z)$ as

$$L(z) = \gamma\hat{P}_{\text{inv}}(z), \tag{8}$$

where $\hat{P}_{\text{inv}}(z)$ is the left-inverse dynamics of the plant model $\hat{P}(z)$ and $0<\gamma\le 1$ is the learning rate gain. If $\hat{P}_{\text{inv}}(z)$ is a left inverse of $P(z)$, then the stability condition in Eq. (5) becomes

$$\max_{\omega\in[0,\pi]} |Q(e^{i\omega})|(1-\gamma) < 1,$$

which is true for any choice of Ω. Therefore, the bandwidth can be selected arbitrarily large for best performance. In practice, however, the system model is never accurate so $\hat{P}_{\text{inv}}(z)$ becomes an approximate left inverse of $P(z)$. In the absence of detailed models of uncertainty bounds, which is challenging to obtain in practice, the inverse model is the optimal design for $L(z)$ to maximize bandwidth. The highest attainable bandwidth is obtained by tuning. In the following, we present the mathematical details to construct Eq. (8) from $\hat{P}(z)$.

1. Construction of $\hat{P}_{\text{inv}}(z)$

Let a minimal state-space realization of the model $\hat{P}(z)$ be given by

$$\hat{P}:\begin{cases} x_P(k+1) = A_P x_P(k) + B_P \zeta(k) \\ \theta(k) = C_P x_P(k) + D_P \zeta(k), \end{cases} \tag{9}$$

where $\zeta(k)$ and $\theta(k)$ are input and output signals, respectively, and A_P, B_P, C_P, and D_P are appropriately sized real matrices. Note that the filter description of $\hat{P}$ is given by $\hat{P}(z)=C_P(zI-A_P)^{-1}B_P+D_P$. A sufficient condition for invertibility is that $D_P=0$ and $\text{rank}(C_PB_P)=n$, where the number of control inputs and system outputs are the same and equal to n. This is the usual case for digitally controlled positioning systems because the sample-and-hold circuitry prevents direct feedthrough, and also the number of inputs typically equals the number of outputs. For general invertibility conditions the interested reader is referred to Ref. 30.

Our construction of $\hat{P}_{\text{inv}}(z)$ is composed of three steps: construction of a delayed inverse filter, stabilization, and noncausal delay correction. Each step is treated individually as follows.

d-step delayed inversion. A d-step delayed inverse filter $\hat{P}_d^{-1}(z)$ is any causal filter such that $\hat{P}_d^{-1}(z)\hat{P}(z)=z^{-d}I$. Such a filter is not unique.[30] For example, consider a delayed inverse filter $\hat{P}_d^{-1}(z)$. Then, a $d+1$-step delayed inverse filter is given by $\hat{P}_{d+1}^{-1}(z)=z^{-1}\hat{P}_d^{-1}(z)$. Although we will correct the delay in a following step, our approach here is to choose the smallest delay for which we can find a delayed inverse filter. For our assumed case, $D_P=0$ and $\text{rank}(C_PB_P)=n$, one can show that the smallest delay is $d=1$. Using standard algorithms,[30] a one-step delayed inverse filter for Eq. (9) is given by

$$\hat{P}_1^{-1}:\begin{cases} x_{i1}(k+1) = A_{i1}x_{i1}(k) + B_{i1}\theta(k) \\ \zeta(k) = C_{i1}x_{i1}(k) + D_{i1}\theta(k), \end{cases}$$

where

$$A_{i1} = A_P - B_P(C_PB_P)^{-1}C_PA_P, \quad B_{i1} = B_P(C_PB_P)^{-1},$$

$$C_{i1} = -(C_PB_P)^{-1}C_PA_P, \quad D_{i1} = (C_PB_P)^{-1}.$$

Stabilization of the inversion. In some cases the d-step delayed inverse filter is unstable, which can be determined by calculating the eigenvalues of A_{id}. If any eigenvalue is outside the unit disk, then the filter is unstable. An unstable filter is problematic because it will lead to exponentially growing control signals, and therefore we must construct a comparable filter that is stable. Suitable approaches for SISO LTI systems are given in Ref. 31. Here, we extend those results to MIMO LTI systems.

To stabilize $\hat{P}_d^{-1}$ we begin by separating the stable modes from the unstable modes as

$$\hat{P}_d^{-1}:\begin{cases} \begin{bmatrix} x_{\text{id,s}}(k+1) \\ x_{\text{id,us}}(k+1) \end{bmatrix} = \begin{bmatrix} A_{\text{id,s}} & 0 \\ 0 & A_{\text{id,us}} \end{bmatrix}\begin{bmatrix} x_{\text{id,s}}(k) \\ x_{\text{id,us}}(k) \end{bmatrix} + \begin{bmatrix} B_{\text{id,s}} \\ B_{\text{id,us}} \end{bmatrix}\theta(k) \\ \zeta(k) = \begin{bmatrix} C_{\text{id,s}} & C_{\text{id,us}} \end{bmatrix}\begin{bmatrix} x_{\text{id,s}}(k) \\ x_{\text{id,us}}(k) \end{bmatrix} + D_{\text{id}}\theta(k), \end{cases} \tag{10}$$

where $x_{\text{id,s}} \in \mathbb{R}^{n_s}$, $x_{\text{id,us}} \in \mathbb{R}^{n-n_s}$, n_s is the number of stable modes, and $A_{\text{id,s}}$, $A_{\text{id,us}}$, $B_{\text{id,s}}$, $B_{\text{id,us}}$, $C_{\text{id,s}}$, and $C_{\text{id,us}}$ are appropriately sized real matrices. We assume that all of the eigenvalues of $A_{\text{id,s}}$ lie on the unit disk and all of eigenvalues of $A_{\text{id,us}}$ lie strictly outside the unit circle. Equation (10) can be obtained for any system using an appropriate state transformation.[32]

A well known result in system theory is that modes that are unstable in positive time are stable in negative time,[33] as illustrated in Fig. 8. Therefore, we can stabilize the inverse filter by replacing the positive time, or causal, unstable modes with dynamically equivalent negative time, or anticausal, modes. The anticausal dynamics are, in fact, the transpose of the system adjoint.[33] That is, given the causal unstable filter,

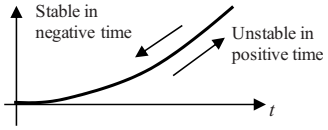

FIG. 8. An unstable mode in positive time is stable in negative time. Simulating the mode in positive time yields exponentially diverging outputs, while simulating in negative time yields exponentially converging outputs.

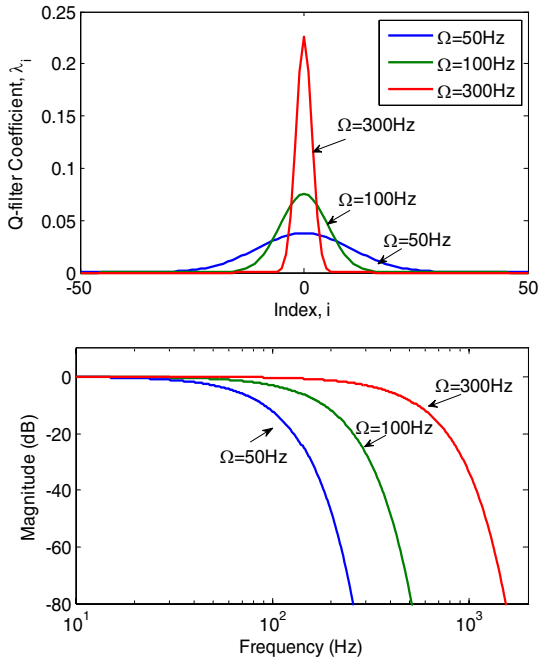

FIG. 9. (Color online) Gaussian filter with $S=4$ kHz in the (a) time domain and (b) frequency domain. The Gaussian filter is a low pass, zero-phase, noncausal filter.

$$x_{\mathrm{id,us}}(k+1) = A_{\mathrm{id,us}}x_{\mathrm{id,us}}(k) + B_{\mathrm{id,us}}\theta(k),$$

$$\zeta_{\mathrm{us}}(k) = C_{\mathrm{id,us}}x_{\mathrm{id,us}}(k),$$

the dynamically equivalent anticausal filter is given by

$$x_{\mathrm{id,nt}}(k-1) = A_{\mathrm{id,nt}}x_{\mathrm{id,nt}}(k) + B_{\mathrm{id,nt}}\theta(k),$$

$$\zeta_{\mathrm{us}}(k) = C_{\mathrm{id,nt}}x_{\mathrm{id,nt}}(k) - D_{\mathrm{id,nt}}\theta(k), \tag{11}$$

where

$$A_{\mathrm{id,nt}} = A_{\mathrm{id,us}}^{-1}, \quad B_{\mathrm{id,nt}} = A_{\mathrm{id,us}}^{-1}B_{\mathrm{id,us}},$$

$$C_{\mathrm{id,nt}} = -C_{\mathrm{id,us}}A_{\mathrm{id,us}}^{-1}, \quad D_{\mathrm{id,nt}} = -C_{\mathrm{id,us}}A_{\mathrm{id,us}}^{-1}B_{\mathrm{id,us}}.$$

Note that $A_{\mathrm{id,nt}}$ has all eigenvalues inside the unit circle because it is the inverse of $A_{\mathrm{id,us}}$, whose eigenvalues are strictly outside the unit circle. Therefore, the anticausal filter Eq. (11) is stable.

Combining the above results, we have that an *unstable* d-step delayed inverse filter can always be written as the *stable* d-step delayed inverse filter

$$\hat{P}_{d,\mathrm{s}}^{-1}:\begin{cases} x_{\mathrm{id,s}}(k+1) = A_{\mathrm{id,s}}x_{\mathrm{id,s}}(k) + B_{\mathrm{id,s}}\theta(k), & x_{\mathrm{id,s}}(0) = 0 \\ x_{\mathrm{id,nt}}(k-1) = A_{\mathrm{id,nt}}x_{\mathrm{id,nt}}(k) + B_{\mathrm{id,nt}}\theta(k), & x_{\mathrm{id,nt}}(N) = 0 \\ \zeta(k) = C_{\mathrm{id,s}}x_{\mathrm{id,s}}(k) + C_{\mathrm{id,nt}}x_{\mathrm{id,nt}}(k) + (D_{\mathrm{id}} + D_{\mathrm{id,nt}})\theta(k), \end{cases} \tag{12}$$

where we have selected zero initial and final conditions on the causal and anticausal dynamics, respectively. A consequence of the stabilization is that we require knowledge of the entire input signal $\theta(k)$, $k=0,\ldots,N$ to calculate the filter output $\zeta(k)$ at any time k. This is possible because we will be using our filter in the ILC updating Eq. (3), which is done offline between iterations so that the entire error signal from the previous iteration is known.

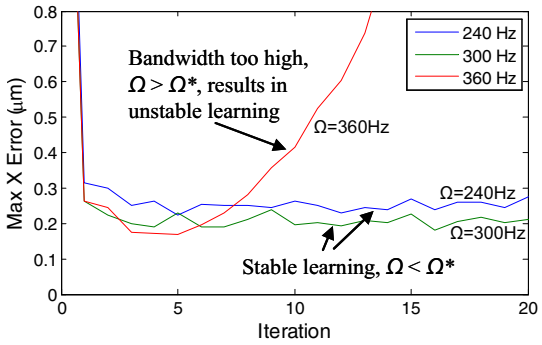

FIG. 10. (Color online) Q-filter tuning results. Bandwidths at or below 300 Hz result in convergent solutions, while bandwidths at or above 360 Hz result in unstable solutions.

Noncausal delay correction. The final step in constructing the learning filter is to correct for the d-step delay. Mathematically, this is achieved by setting $\hat{P}_{\mathrm{inv}}(z) = z^d\hat{P}_{d,\mathrm{s}}^{-1}(z)$, which results in $\hat{P}_{\mathrm{inv}}(z)\hat{P}(z) = I$, as desired. That is, we add d forward-time shifts to our filter to cancel the d-step inversion delay. In practice, it is usually easier to directly apply the forward time shifts to the input signal. The result is the following modified, model-inversion first-order ILC update algorithm:

$$w_{j+1}(k) = Q(z)I_{n\times n}[w_j(k) + \gamma\hat{P}_{d,\mathrm{s}}^{-1}(z)e_j(k+d)]. \tag{13}$$

2. Learning rate selection

The learning rate, γ, determines how quickly the ILC will converge. $\gamma=1$ converges quickly, while $\gamma=0^+$ converges slowly. γ also affects the noise sensitivity of the ILC. If $\gamma=1$, then sensor noise will pass through the update algorithm, into the control. For example, consider the noisy error signal $e(k)=\tilde{e}(k)+n(k)$, where $\tilde{e}(k)$ is the actual error, e is the measured error and n is the noise. Then, the learning opera-

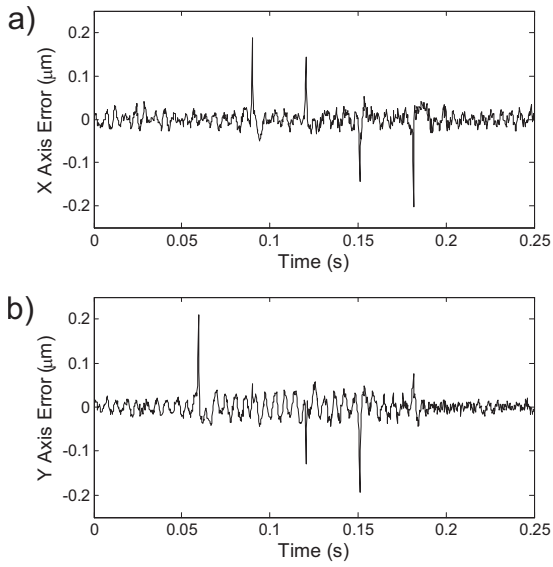

FIG. 11. Tracking results with a fixed 300 Hz bandwidth Q-filter on the "star" pattern presented in Sec. VI. The time series of tracking errors for the (a) X axis and (b) Y axis are shown. Rapid accelerations in the reference trajectory result in error "spikes" in both axes.

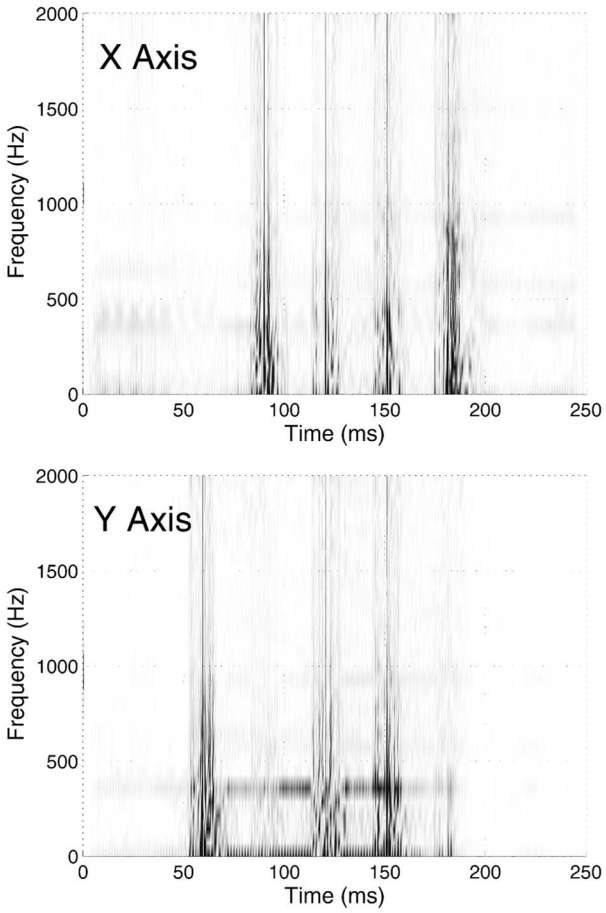

FIG. 12. WVTF decomposition of signals in Fig. 11. Error spikes seen in Fig. 11 have high-frequency content over short periods of time, called α-segments. The time-frequency decomposition is used to select parameters for the Q-filter design.

tion Eq. (3) becomes $w_{j+1}(k)=Q(z)[w_j(k)+\gamma\hat{P}_{d,s}^{-1}(z)\tilde{e}(k)]+Q(z)\gamma\hat{P}_{d,s}^{-1}(z)n(k)$, where $Q(z)\gamma\hat{P}_{d,s}^{-1}(z)n(k)$ is noise on the control signal. Using a smaller γ will reduce the amount of noise transmitted to the control at the expense of convergence rate.

Remark 2: In Refs. 23–25 γ is used to reduce sensitivity of the learning algorithm to model uncertainty, especially for inaccuracies in the phase of the model. A small γ reduces sensitivity and can extend the learnable frequency range, although at the expense of convergence rate. The same approach can be applied here, and may result in a higher Q-filter bandwidth.

TABLE I. Identified α-segments for the time-frequency decomposition shown in Fig. 12.

X-axis α-segments		Y-axis α-segments		Combined α-segments	
Time (ms)	Range (Hz)	Time (ms)	Range (Hz)	Time (ms)	Range (Hz)
		59.5	2000	59.5	2000
90	2000			90	2000
120.5	2000	~120	2000	120.5	2000
151	2000	151	2000	151	2000
181.5	2000			181.5	2000

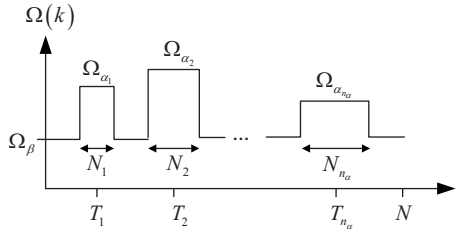

FIG. 13. Parametrized bandwidth profile $\Omega(k)$ for a time-varying Q-filter. α-segment times T_i are obtained from time-frequency decomposition. Shape parameters Ω_{α_i}, N_i, and Ω_β are obtained from numerical optimization.

B. LTI *Q* design

1. Zero-phase filter construction

For best performance, the Q-filter should be a zero-phase, zero-delay filter.[34] Zero phase and zero delay necessitate the use of a symmetric, noncausal filter, although this is not problematic because the Q-filtering is done offline with the complete data set. We use a FIR Gaussian filter because it is natively defined as a symmetric filter and the filter coefficients can be written explicitly as a function of the bandwidth, which is convenient for implementation of the advanced ILC design discussed in Sec. V. The Gaussian Q-filter with bandwidth Ω Hz is given by

$$Q_\Omega(z)=\lambda_{-N_Q}(\Omega)z^{N_Q}+\cdots+\lambda_{-1}(\Omega)z+\lambda_0(\Omega)+\lambda_1(\Omega)z^{-1}+\cdots+\lambda_{N_Q}(\Omega)z^{-N_Q}, \tag{14}$$

$$\lambda_i(\Omega)=\frac{1}{\sum_{r=-N_Q}^{N_Q}\exp\left[-\frac{r^2(2\pi\Omega)^2}{S^2\ln 4}\right]}\exp\left[-\frac{i^2(2\pi\Omega)^2}{S^2\ln 4}\right], \tag{15}$$

where S is the sample frequency in hertz and N_Q is the support. The Gaussian filter is shown in Fig. 9 for $S=4000$ Hz and $N_Q=50$.

As an alternative to the Gaussian filter, any causal filter (Butterworth, Chebychev, Bessel, etc.) could be used to construct the FIR symmetric Q-filter as follows. Let $h(k)$, $k=0,\ldots,N_Q$ be the truncated impulse response of a causal filter with bandwidth Ω. Then symmetric Q-filter coefficients are calculated as $\lambda_k=h(k)*h(-k)$, where $*$ is the convolution operator. Although more complex to implement than the Gaussian filter, constructing the Q-filter from a causal filter may result in sharper roll off and better performance.

2. Bandwidth selection

From Eq. (5) we have

$$\max_{\omega\in[-\pi,\pi]}|Q(e^{i\omega})|\bar{\sigma}[I-L(e^{i\omega})P(e^{i\omega})]<1.$$

In most cases plant model uncertainty will imply that there exists some smallest frequency Ω^*, where $\bar{\sigma}[I-L(e^{i\Omega^*})P(e^{i\Omega^*})]\geq 1$. To satisfy stability condition Eq. (5), we must choose our bandwidth $\Omega<\Omega^*$. The maximum bandwidth, Ω^*, can be determined experimentally as follows. Choose any trajectory that excites the system at all frequencies, such as a step function. Start with a low-frequency bandwidth for Q and iterate until convergence. Signal norms

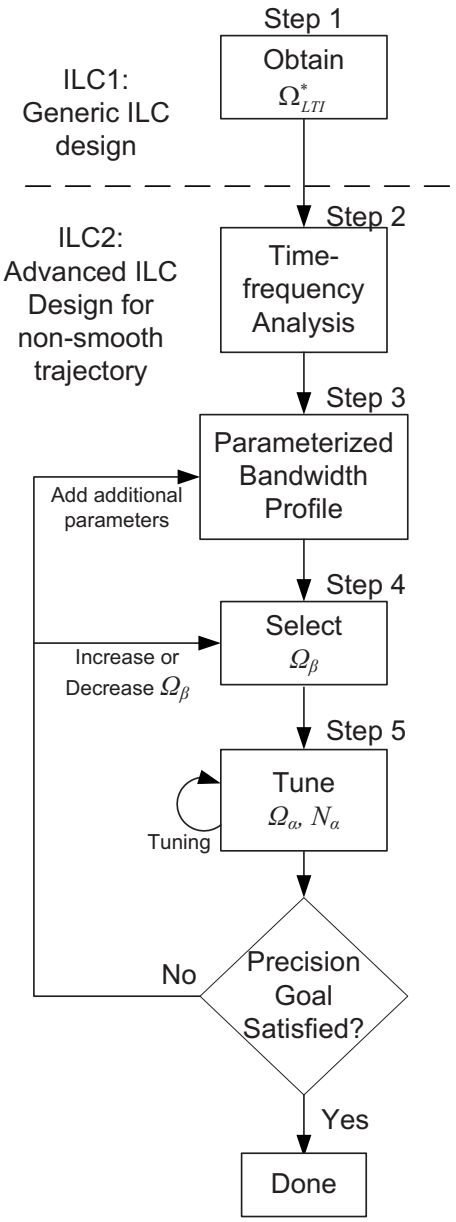

FIG. 14. Tuning process flowchart for bandwidth profile $\Omega(k)$. The high-bandwidth design presented in Sec. VI is the starting point for the advanced design.

such as maximum error or rms error are useful for determining when convergence has occurred. After convergence occurs, increase the bandwidth for Q and restart the learning. Repeat until divergence is observed, as illustrated in Fig. 10. The highest stable bandwidth should be selected.

Any changes to the motion system that impacts dynamics, such as loading changes, changes in environmental conditions, or actuator wear may require retuning of the Q-filter bandwidth. If the change to the system is sufficiently dramatic, it may be necessary to reidentify the system model and recalculate the learning filter.

V. AN ADVANCED *Q* DESIGN FOR NONSMOOTH TRAJECTORIES

As discussed in Sec. I, many common precision motion trajectories are nonsmooth. Characteristically, these trajectories have long periods of constant or nearly constant velocity cruises with short periods of rapid acceleration. An example of this type of trajectory is the triangle wave shown in Fig. 2. For these types of references, the bandwidth required to precisely track during rapid accelerations may be orders of magnitude higher than the bandwidth necessary during constant velocity cruises. Therefore, precision tracking during short, rapid accelerations represents both a common and particularly difficult challenge in precision motion control. In this section, we present an advanced design for the Q-filter that is tailored to improve performance this type of trajectory.

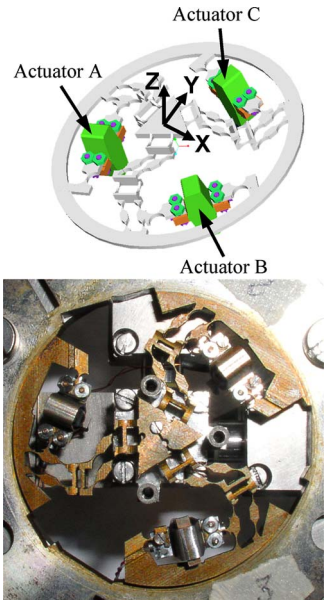

FIG. 15. (Color online) PKM (Refs. 19 and 28) schematic and photograph. Actuators A, B, and C are piezoelectric and outputs X, Y, Z are Cartesian coordinates. Parallel kinematics give higher stiffness (faster response) and balanced dynamics at the expense of dynamic coupling.

A. Time-varying *Q*-filter

Our goal is to design a linear time-varying (LTV) Q-filter that results in improved performance beyond traditional LTI Q-filters by increasing bandwidth during short portions of the trajectory where rapid accelerations occur. We consider our Q-filter to be a FIR-type filter as before, Eq. (14), except that we now consider a time-varying bandwidth, $\Omega(k)$, and thus the filter coefficients [Eq. (15)] change with time. The time-varying bandwidth is designed for a particular trajectory, and any change to the trajectory will require a new design for $\Omega(k)$. Although we lose generality with this design, the benefit is improved performance.

B. Time-frequency analysis

Our design process begins with the largest fixed bandwidth obtained in Sec. IV B 2, which we refer to as Ω^*_{LTI}. We set the Q-filter bandwidth to Ω^*_{LTI}, iterate until convergence, and record the converged error signal. For example, the X-axis error signal for the system in Section VI is shown in Fig. 11. From the error signal we note several large spikes in

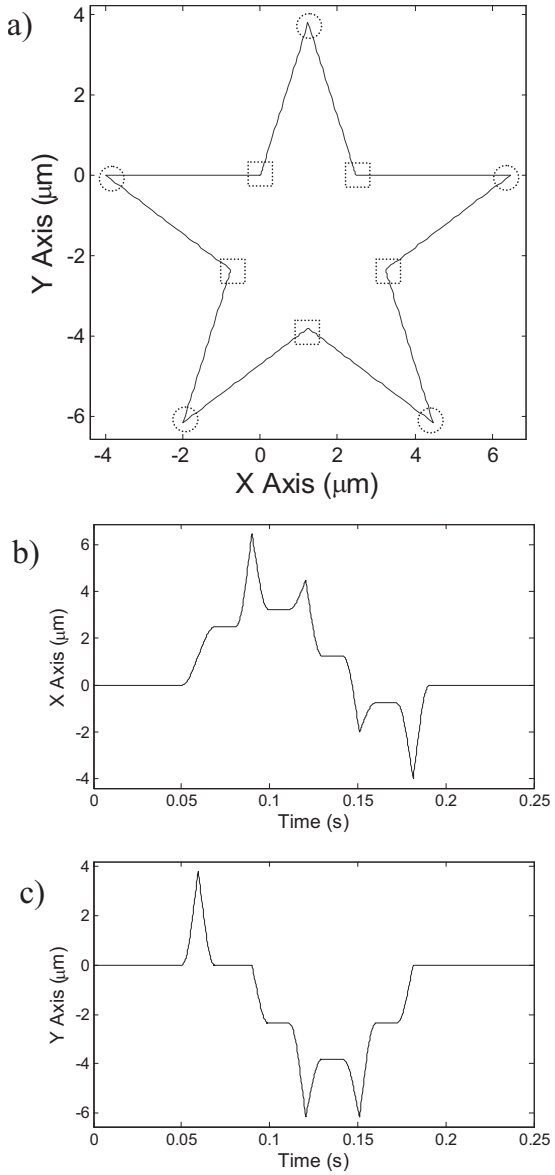

FIG. 16. Star reference (a) in the *X*-*Y* plane, (b) time series for the *X* axis, and (c) time series for the *Y* axis. Dashed squares designate corners with 10 cm/s^2 acceleration and dashed circles designate corners with 4.6 m/s^2 acceleration. Maximum velocity is 600 $\mu m/s$.

the error. These error spikes correspond to rapid accelerations in the reference signal that are beyond the bandwidth of the Q-filter.

To isolate temporal locations of high-frequency content, we use a time-frequency analysis. Here, the Wigner–Ville time-frequency (WVTF) distribution is used because it is the simplest among Cohen's class of distribution functions.[35] Any distribution function from Cohen's class or wavelet decomposition, could be used in this step. The WVTF decomposition is given by

$$\mathrm{WVTF}(k,\omega) \triangleq \frac{1}{\pi}\sum_{\tau=-N_W}^{N_W} e(k-\tau)e(k+\tau)e^{(-i2\omega\tau)},$$

where N_W is the window size. The distribution for the error signals in Fig. 11 is shown in Fig. 12. The decomposition shows frequency spikes at the same times as the magnitude spikes from Fig. 11. We refer to segments of the trajectory that result in spikes in error magnitude and frequency as α-segments. The temporal location and approximate frequency range of each α-segment is recorded. We combine the α-segments identified on each axis because the system is coupled, and therefore bandwidth must be increased simultaneously on all of the system inputs. Table I lists the location and range for the α-segments identified from Fig. 12. The remaining segments of the error with low magnitude and/or frequency are referred to as β-segments. The designation of α-segments is critical because our design will focus on performance improvement only in these sections.

The reader may be tempted to forgo the time-frequency analysis because in the above example, the same α-segments can be identified directly from the time-domain error signal. This is, however, not true in general. For some complex trajectories it may be difficult to precisely locate α-segments from the time-domain signal alone.

C. Filter shaping

In this step we construct a parametrized time-varying bandwidth profile using the α-segments found in the time-frequency analysis. For simplicity, we use the same bandwidth for all β-segments, denoted by Ω_β. Centered at each α-segment we consider a variable bandwidth $\Omega_{\alpha,i}$ for a period of $N_{\alpha,i}$. The parametrized time-varying bandwidth profile is shown in Fig. 13.

To minimize the number of variables, α-segments with similar frequency ranges in the time-frequency decomposition can share the same tuning variables. For the time-frequency decompositions shown in Fig. 12, it is reasonable to use the same bandwidth Ω_α and period N_α for all α-segments.

D. Parameter tuning

Theoretical results indicate that there is a trade-off between β-segment bandwidth and α-segment bandwidth and width.[36] Rigorous bounds on these trade-offs[36] take the form

$$f_\beta[\Omega_\beta, W(z)] + \max_{i=1,\ldots,n_\alpha}\{f_\alpha[\Omega_{\alpha_i}, N_{\alpha_i}, W(z)]\} + f_{\mathrm{coupling}}[\min_{i=1,\ldots,n_\alpha-1}(T_{i+1}-T_i), W(z)] < 1,$$

where $\bar{\sigma}[W(e^{i\omega})] \geq \bar{\sigma}\{\hat{P}^{-1}(e^{i\omega})[P(e^{i\omega}) - \hat{P}(e^{i\omega})]\}$ bounds the model uncertainty and $f_\beta[\cdot, W(z)]$, $f_\alpha[\cdot,\cdot,W(z)]$, and $f_{\mathrm{coupling}}[\cdot, W(z)]$ are nonlinear monotonic functions of $\cdot$ given in Ref. 36. That is, increasing or widening an α-segment bandwidth will increase f_α, and thus the β-segment bandwidth must be lowered to reduce f_β. This may be expected since, intuitively, performance cannot be increased everywhere without trade-off. The coupling term f_{coupling} decreases exponentially with the length of time between α-segments[36] so that α-segments very close together are penalized heavily, but this effect dissipates rapidly as α-segments are separated. Since the trade-off functions f_β, f_α, and f_{coupling} depend on $W(z)$, which is time consuming and challenging to obtain in practice, our approach here is to tune the profile parameters to obtain optimal results. Our approach uses the trade-off

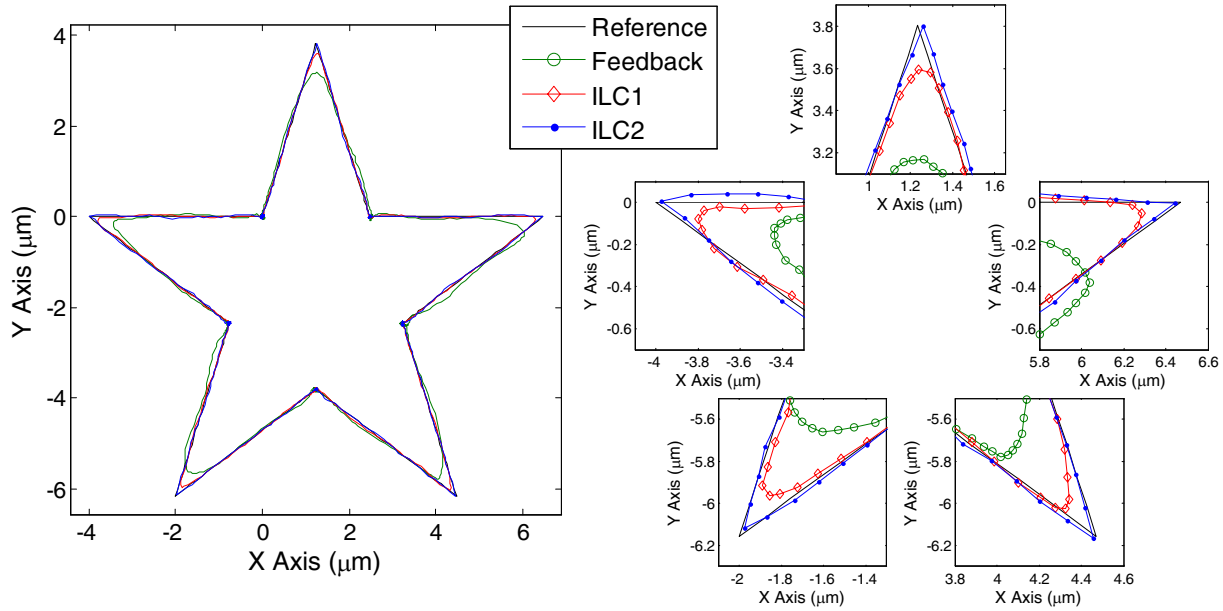

FIG. 17. (Color online) Experimental contour tracking results. Plot (a) shows the entire *X*-*Y* plane and plot (b) shows an exploded view of the outside corners (measured data points are shown with dots). Feedback control and ILC1 do not have sufficient bandwidth to precisely track the trajectory, resulting in rounded corners. ILC2 tracks corners accurately.

relationships between f_β, f_α, and f_{coupling} to guide the tuning, but we do not calculate the value of those functions. Additionally, the focus of the tuning will be on Ω_β, Ω_{α_i}, and N_{α_i} which have trade-offs through f_β and f_α. We assume that the reference trajectory cannot be modified, and thus the f_{coupling} term is not modified by our tuning procedure.

Parameter tuning is performed as follows. First, the β-segment bandwidth Ω_β is set smaller than Ω^*_{LTI} (10%–20% reduction is recommended). By setting Ω_β lower than Ω^*_{LTI}, the α-segments can be increased. Next, α-segment width N_α and bandwidth Ω_α are tuned to find the best achievable performance for the current Ω_β. For each set $\{N_\alpha, \Omega_\alpha, \Omega_\beta\}$, several iterations of learning should be performed to reliably determine convergence and converged performance. Next, a new β-segment bandwidth Ω_β is selected and the optimal α-segment parameters are again obtained through tuning. The process is repeated until the optimal bandwidth parameters are obtained. If results are unsatisfactory, additional tuning parameters can be included by adding more α-segments. The complete tuning process is shown in the schematic in Fig. 14.

VI. APPLICATION: PARALLEL KINEMATIC MECHANISM

In this section we design two ILC algorithms using the methodologies presented in Secs. IV and V. The ILC algorithms are designed for the *XYZ* parallel kinematic mechanism[19,28] (PKM) that is shown in Fig. 15. In PKMs the end effector is connected by multiple independent kinematic chains in a parallel scheme.[19] As compared to serial kinematic mechanisms, PKMs have lower inertia, higher bandwidths, a balanced mechanical structure, and no linkage error accumulation.[19]

Actuation of the PKM used here is provided by three piezos, labeled as A, B, and C in Fig. 15. Position measurement in Cartesian axes *X*, *Y*, and *Z* is provided by capacitance sensors mounted below the stage. The motion range is approximately 85 μm in each axis and position accuracy is approximately 4 nm.[19] System noise results in a machine precision level of approximately 60 nm on each axis. Control algorithms are run on a DS1104 dSPACE DSP controller board with a sampling rate of 4 kHz.

To evaluate the ILC algorithms, the *XY* star trajectory shown in Fig. 16 is used. Although the star trajectory is designed solely for the purpose of evaluation, the reader may consider this trajectory analogous to the triangle wave function shown in Fig. 2. Rather than extending the triangle wave down one axis, the star is essentially a triangle wave that is wrapped radially around the center point. The maximum velocity along the legs of the star is 600 μm/s. A relatively low acceleration of 10 cm/s^2 is used on the inside corners of the star to demonstrate tracking on smooth trajectories. At the outside corners the direction change is nonsmooth with a high acceleration of 4.6 m/s^2.

A. Feedback control results

To provide a base line by which to compare the ILC algorithms, we first present tracking results using only the feedback controller. $\mathcal{H}_\infty$ feedback controllers are designed in Ref. 20 resulting in 40 Hz closed-loop bandwidth. Tracking results for the feedback controller are shown in Figs. 17 and 18. From the contour plot in Fig. 17 we see that the feedback controller is able to provide decent tracking during the low accelerations and constant velocity cruises. The most significant contour errors occur at the outside corners of the star where acceleration is highest. Although the outside corner is intended to have a 4.6 m/s^2 acceleration, the feedback controller only generates between 0.8 and 1.0 m/s^2 of acceleration. We also see from Fig. 18 that although the star contour

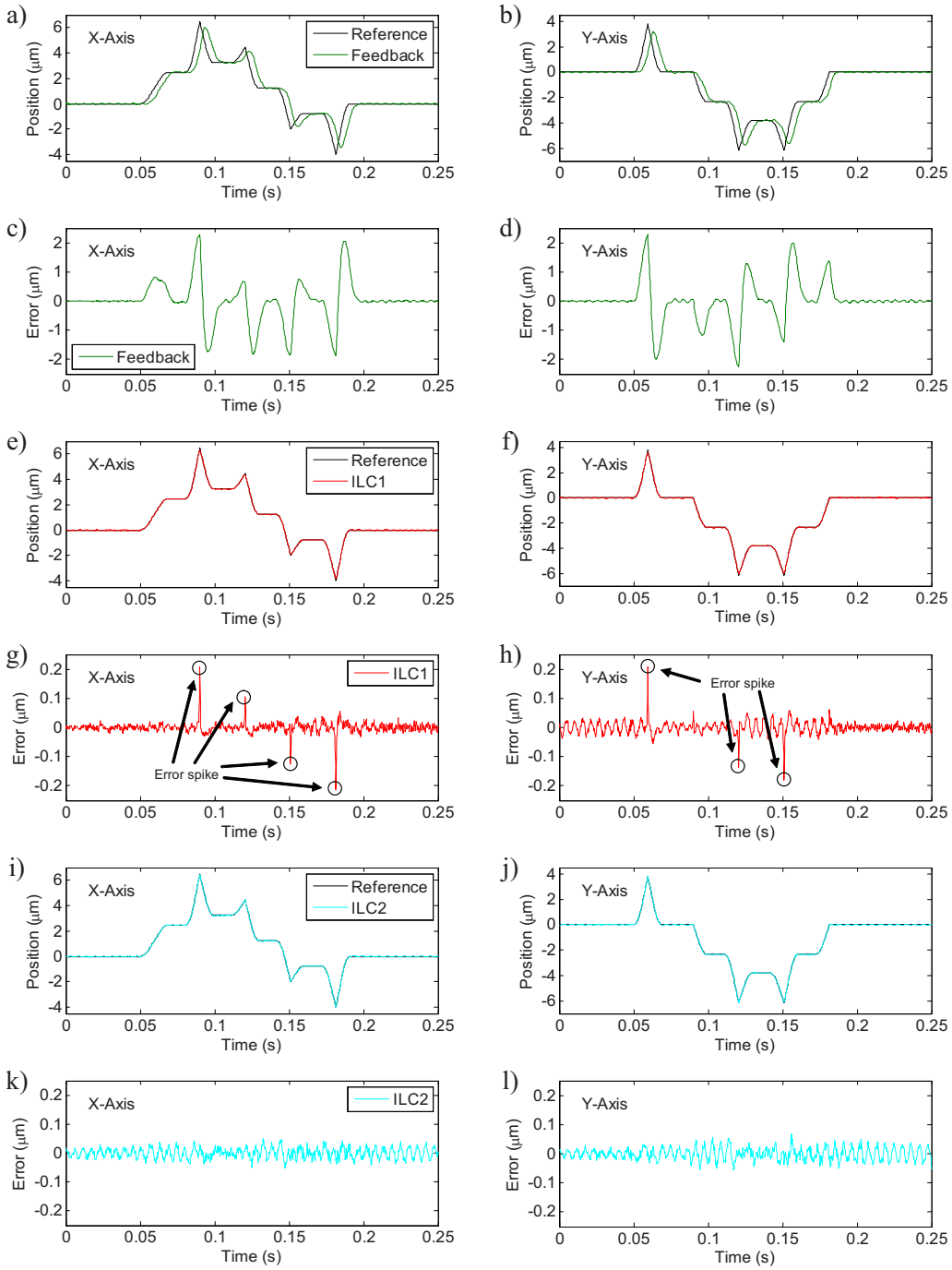

FIG. 18. (Color online) Experimental time series of tracking results. Left column shows the *X* axis results and right column shows the *Y* axis results. Feedback control tracking is shown in (a) and (b), and error is shown in (c) and (d). ILC1 tracking is shown in (e) and (f), and error is shown in (g) and (h). ILC2 tracking is shown in (i) and (j) and error is shown in (k) and (l).

is tracked reasonably well, there is a significant lag in each axis. In a process where timing is critical, this lag may be problematic.

B. ILC1: LTI *Q*-filter

To improve tracking performance, an ILC is designed using a constant bandwidth, which we refer to as ILC design 1, or ILC1. The learning filter is constructed as described in Sec. IV A. The *Q*-filter bandwidth is tuned to obtain the optimal bandwidth. A few results of the tuning process are shown in Fig. 10, and the highest convergent bandwidth is obtained as $\Omega^*_{\mathrm{LTI}} \cong 300$ Hz. Figure 19 shows that ILC1 converges in approximately five iterations.

The *X*- and *Y*-axis tracking errors using ILC1 are shown in Figs. 17 and 18. Unlike the feedback control, no lag is discernable with ILC1. In fact, the error is near machine precision except at a few locations labeled "error spike" in Fig. 18. The error spikes occur at the outside corners of the star, and appear seen as rounded corners on the contour plot in Fig. 17. Although ILC1 has much lower error at the outside corners than the feedback controller, the actual accelera-

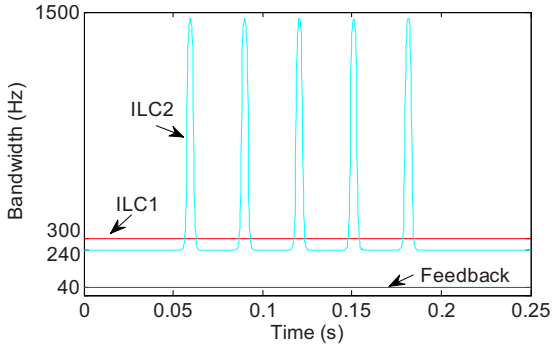

FIG. 19. (Color online) Plot of feedback, ILC1 and ILC2 controller bandwidths. ILC2 uses much higher bandwidth during accelerations.

tion generated by the system is between 1.0 and 1.2 m/s^2, well short of the desired 4.6 m/s^2. To track the high acceleration and achieve low error at the outside corners, higher bandwidth is needed.

C. ILC2: LTV *Q*-filter

In this section we present results using an LTV Q-filter. Design begins by selecting the β-segment bandwidth as 240 Hz, or 20% lower bandwidth than Ω^*_{LTI}. The α-segment bandwidth and width are tuned over several trials to obtain a final result with approximately 1500 Hz peak bandwidth and a width of $N_\alpha=7$ samples (1.75 ms). Because a large change in bandwidth may result in a discontinuity in the control signal, we smooth $\Omega(k)$ using a low pass filter so that the bandwidth transitions gradually from using Ω_β to Ω_α. The final bandwidth profile is shown in Fig. 20.

Tracking results for ILC2, shown in Figs. 17 and 18, demonstrate that best performance is obtained using a LTV Q-filter. The error spikes evident in ILC1 are eliminated using ILC2 due to the increased bandwidth at those locations. The peak acceleration generated using ILC2 at the outside corners is between 3.5 and 4.5 m/s^2, much closer than feedback or ILC1 to the desired 4.6 m/s^2 acceleration used in the reference. Perhaps most interesting is the exploded view of the outside corners in Fig. 17. In this view, dots represent actual data points and the lines connecting them are added as a visual aid. Clearly, ILC2 is able to provide high precision even when the sampling rate is slow compared to velocities and accelerations commanded.

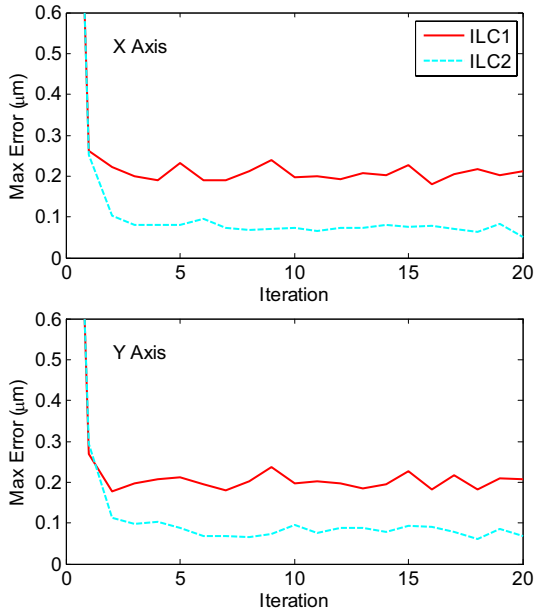

FIG. 20. (Color online) Convergence profile for ILC1 and ILC2. Convergence occurs in approximately three to seven iterations.

TABLE II. Performance for star trajectory.

Algorithm	Maximum X error (nm)	% X reduction	Maximum Y error (nm)	% Y reduction
Feedback	2287		2288	
ILC1 (LTI)	204	92	201	91
ILC2 (LTV)	72	97	82	96

Tracking performance results are summarized in Table II where we find approximately 91% and 96% improvements in peak tracking error over feedback control with ILC1 and ILC2, respectively. The improvement from ILC1 to ILC2 can be calculated as approximately 60% on both axes.

VII. CONCLUSIONS

This article considers the problem of high-bandwidth precision motion control of scientific instrumentation. For repeating trajectories, such as in raster scanning, ILC can be used to provide control bandwidths above the first system resonance, where feedback control may have difficulty. Two ILC designs for multiaxis systems are presented. The first design is a high-bandwidth design that uses a time-domain solution to the model inversion for its learning function. The Q-filter bandwidth is tuned to the maximum stable bandwidth. The second design uses a time-varying Q-filter bandwidth with short segments of very high bandwidth separated by longer segments with low bandwidth. The time-varying bandwidth improves tracking performance on nonsmooth trajectories, such as the triangle function used in raster scanning.

Both designs are applied to a three axis PKM and evaluated on a nonsmooth trajectory. The first ILC design increases system bandwidth to 300 from 40 Hz with feedback control alone. The second ILC design increases system bandwidth to 1500 Hz for 1.75 ms, which is sufficient to track the 4.6 m/s^2 reference acceleration with machine-level precision.

ACKNOWLEDGMENTS

This work was supported by the University of Illinois Nano-CEMMS Center, NSF Award No. 0328162.

[1] D. Y. Abramovitch, S. B. Andersson, L. Y. Pao, and G. Schitter, Proceedings of the American Control Conference, 2007 (unpublished), pp. 3488–3502.
[2] A. Alessandrini and P. Facci, Meas. Sci. Technol. **16**, R65 (2005).
[3] F. J. Giessibl, Rev. Mod. Phys. **75**, 949 (2003).
[4] J. Loos, Adv. Mater. (Weinheim, Ger.) **17**, 1821 (2005).
[5] Q. Yang and S. Jagannathan, Int. J. Nanotechnol. **3**, 527 (2006).
[6] M. Guthold, G. Matthews, A. Negishi, R. M. I. Taylor, D. Erie, F. P. Brooks, Jr., and R. Superfine, Proceedings of the Conference on Development and Industrial Application of Scanning Probe Methods, 1999 (unpublished), pp. 437–443.

[7] M. Sitti and H. Hashimoto, IEEE/ASME Trans. Mechatron. **5**, 199 (2000).
[8] A. A. G. Requicha, Proc. IEEE **91**, 1922 (2003).
[9] H. Chen, N. Xi, and G. Li, IEEE. Trans. Autom. Sci. Eng. **3**, 208 (2006).
[10] E. Eleftheriou, T. Antonakopoulos, G. K. Binnig, G. Cherubini, M. Despont, A. Dholakia, U. Durig, M. A. Lantz, H. Pozidis, H. E. Rothuizen, and P. Vettiger, IEEE Trans. Magn. **39**, 938 (2003).
[11] S. Salapaka, A. Sebastian, J. P. Cleveland, and M. V. Salapaka, Rev. Sci. Instrum. **73**, 3232 (2002).
[12] Y. Li and J. Bechhoefer, Rev. Sci. Instrum. **78**, 013702 (2007).
[13] K. K. Leang and S. Devasia, IEEE Trans. Control Syst. Technol. **15**, 927 (2007).
[14] S. Arimoto, S. Kawamura, and F. Miyazaki, J. Rob. Syst. **1**, 123 (1984).
[15] K. L. Moore, *Iterative Learning Control for Deterministic Systems* (Springer, Berlin, 1993).
[16] Z. Bien and J.-X. Xu, *Iterative Learning Control: Analysis, Design, Integration and Applications* (Kluwer, Dordrecht, 1998).
[17] D. A. Bristow, M. Tharayil, and A. G. Alleyne, IEEE Control Syst. Mag. **26**, 96 (2006).
[18] K. K. Leang and S. Devasia, Mechatronics **16**, 141 (2006).
[19] Q. Yao, J. Dong, and P. M. Ferreira, Precis. Eng. **32**, 7 (2008).
[20] J. Dong, S. M. Salapaka, and P. M. Ferreira, Proceedings of the IEEE Conference on Decision and Control, 2008 (unpublished), pp. 4495–4500.
[21] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design* (Wiley, New York, 1996).
[22] S. Devasia, IEEE Trans. Autom. Control **47**, 1865 (2002).
[23] Q. Zou, C. V. Giessen, J. Garbini, and S. Devasia, Rev. Sci. Instrum. **76**, 023701 (2005).
[24] S. Tien, Q. Zou, and S. Devasia, IEEE Trans. Control Syst. Technol. **13**, 921 (2005).
[25] K.-S. Kim, Q. Zou, and C. Su, Proceedings of the American Control Conference, 2007 (unpublished), pp. 4227–4233.
[26] Y. Li and J. Bechhoefer, Proceedings of the American Control Conference, 2008 (unpublished), pp. 2703–2709.
[27] K.-S. Kim and Q. Zou, Proceedings of the American Control Conference, 2008 (unpublished), pp. 2710–2715.
[28] J. Dong, Q. Yao, and P. M. Ferreira, Precis. Eng. **32**, 20 (2008).
[29] M. Norrlof, *Iterative Learning Control: Analysis, Design, and Experiments* (Linkoping Studies in Science and Technology, 2000) Chap. 4, pp. 31–66.
[30] M. K. Sain and J. L. Massey, IEEE Trans. Autom. Control **14**, 141 (1969).
[31] T. Sogo, Proceedings of the 41st IEEE Conference on Decision and Control, 2002 (unpublished), pp. 3730–3735.
[32] C.-T. Chen, *Linear System Theory and Design* (Oxford University Press, Oxford, 1999).
[33] M. A. Dahleh and I. J. Diaz-Bobillo, *Control of Uncertain Systems: A Linear Programming Approach* (Prentice-Hall, Englewood Cliffs, NJ, 1995).
[34] R. W. Longman, Int. J. Control **73**, 930 (2000).
[35] L. Cohen, *Time-Frequency Analysis* (Prentice-Hall, Englewood Cliffs, NJ, 1995).
[36] D. A. Bristow and A. G. Alleyne, IEEE Trans. Autom. Control **53**, 582 (2008).