2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

A Comparison of ILC Architectures for Nanopositioners with Applications to AFM Raster Tracking

Jeffrey A. Butterworth, Lucy Y. Pao, and Daniel Y. Abramovitch

Abstract— **In previous work, we compared the raster tracking performance of two distinct combined feedforward/feedback control architectures while using model-inverse-based feedforward control [1], [2]. In this paper, we extend that work into the application of parallel and serial iterative learning control (ILC) architectures. These ILC architectures naturally relate to the two previously studied combined feedforward/feedback control architectures, feedforward closed-loop injection (FFCLI) and feedforward plant injection (FFPI). Experimental learning results from an atomic force microscope (AFM) raster scanner are provided as well as results comparing the FFPI and FFCLI architectures with those of the learned performance for parallel and series ILC. We show that the value of ILC over model-inverse-based feedforward methods is increased in the presence of model uncertainty or variation.**

I. Introduction

Several research groups have investigated improving upon the performance of feedback-only control by combining it with a feedforward filter [1]–[11]. In particular, two distinct combined feedforward/feedback architectures have appeared in the atomic force microscope (AFM) literature: the plant-injection (FFPI) and closed-loop-injection architectures (FFCLI) in [1], [2] and [3]. In these three references, the objective of these combined feedforward/feedback architectures is to improve the tracking of high-speed raster patterns for AFM imaging, and we have the same objective here.

Both the FFPI and FFCLI architectures can be described with the block diagram shown in Fig. 1. When using the FFCLI architecture, F_P in Fig. 1 is set to zero and the feedforward filter F_{CL} acts on the reference signal ahead of the closed-loop system. When using the FFPI architecture, F_P in Fig. 1 is designed to perform as the feedforward filter while F_{CL} is a unity-gain function with appropriate phase properties. Specifically, [1] and [3] discuss a method for the design of F_{CL} in the FFPI architecture which uses a single-parameter adaptive delay method; all FFPI experimental results shown in this paper will utilize the F_{CL} design technique of [1], [3].

When given the standard FFPI (without the adaptive delay method of [1], [3]) and FFCLI architectures, one can calculate the F_{CL} filter of the FFCLI architecture required to achieve theoretical equivalence to the FFPI architecture (and vice versa). This suggests that, in theory, there is little value in comparing the two architectures. In practice, FFCLI is easier to implement, but we have seen its performance degrade as the order of the filter grows. Implementation of FFPI is slightly more difficult, but it has the advantage of reduced numerical sensitivity (relative to FFCLI) and it retains an obvious connection to identified plant parameters which can be advantageous for applying indirect adaptive methods. Further, the single-parameter adaptive delay method of [1], [3] continually results in better tracking performance, but it cannot be applied to the FFCLI architecture.

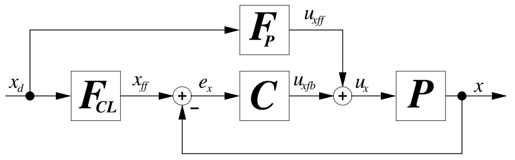


Fig. 1. A combined feedforward/feedback block diagram that can be applied to the raster direction of a piezo-based positioning system. It consists of a feedback compensator C, a plant-injection feedforward controller F_P, and a closed-loop-injection feedforward controller F_{CL}.

Model-inverse methods are an effective way to design feedforward filters. For example, when designing F_P for the FFPI architecture, we can set F_P equal to the inverse of the model of the plant, $F_P = P^{-1}$. Assuming P is exactly proper, minimum phase, and perfectly modeled, then the overall transfer function reduces to unity and we can expect perfect tracking of the reference signal x_d.

Linear model-inverse-based control can struggle in the presence of uncertainty, modeling errors, and/or nonlinearities. Our system has some uncertainty and variation that has challenged the performance of the tracking of a raster pattern. This provides substantial motivation for the investigation of a self-calibrating algorithm such as ILC to continue to improve upon raster tracking. This paper discusses the application of ILC to our system, where we focus on the use of the serial and parallel ILC architectures [12] which is a natural extension (and comparison) to our past work comparing the FFPI and FFCLI architectures of Fig. 1.

The arrangement for a parallel application of ILC is given in Fig. 2. The specifics of this block diagram will be discussed in Section III, but for now, the reader should note that the ILC command vector u_{j+1} is injected into the same location of the general feedback loop as the output of the F_P filter in the FFPI architecture. As a result, it is natural to

This work was supported in part by Agilent Technologies, Inc., the US National Science Foundation (NSF Grant CMMI-0700877), and a Richard & Joy Dorf Professorship. The authors gratefully acknowledge Professor Andrew G. Alleyne for his guidance on the application of ILC to our system.

J. A. Butterworth is a graduate student, and L. Y. Pao is a professor; both are with the Dept. of Electrical, Computer, and Energy Engineering at the University of Colorado at Boulder, Boulder, CO 80309 USA, butterwo@colorado.edu, pao@colorado.edu

D. Y. Abramovitch is a senior research engineer in the Nanotechnology Group at Agilent Laboratories, 5301 Stevens Creek Blvd., M/S: 4U-SB, Santa Clara, CA 95051 USA, danny@agilent.com

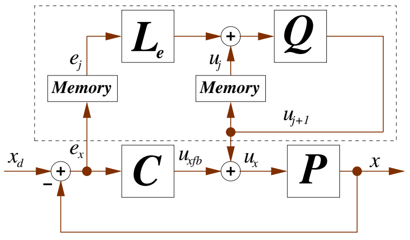


Fig. 2. A single-input single-output (SISO) block diagram representing an application of parallel ILC. The dotted line encloses the ILC system that has been added to a standard feedback control loop. L_e and Q are learning filters. This parallel ILC architecture is a cousin of the FFPI architecture of Fig. 1. Here, e_j, u_j, and u_{j+1} are vectors.

compare the parallel ILC architecture with that of the FFPI architecture. The same can be said for a serial application of ILC (Fig. 3) and the FFCLI architecture.

II. Motivation for the Application of ILC

Fig. 4 shows some FFCLI and FFPI experimental results of each architecture tracking a 152.4 Hz raster pattern when using model-inverse feedforward design. Specifically, in Fig. 4(b), $F_P = 0$ and F_{CL} is the inverse of the closed-loop system with the appropriate amount of delay added to make the filter causal. A special case of FFPI (including an adaptive delay filter for F_{CL}) [1] is used to obtain the performance shown in Fig. 4(a). In that case, F_P is the inverse of the plant model (with the appropriate amount of delay added to make the filter causal) and $F_{CL} = z^{-\tau}$ where τ is an adapted integer that can update every raster period. In these results, the plant model P is minimum phase, strictly proper and includes a 10-sample transport delay.

For each experiment in Fig. 4, the feedback filter $C(z)$ is the same 5th-order H_∞ filter. It was designed following techniques described in [13]. Feedback design objectives include maximizing bandwidth, limiting saturation, and avoiding excitation of the plant's resonance frequency. The exact $C(z)$ is provided in (1). The discrete sampling frequency of all controllers (feedback and feedforward) is 25 kHz.

$$C(z) = \frac{0.74736(z+1)(z^2-1.816z+0.8284)}{(z+0.7072)(z-0.7899)(z-1)} \times \frac{(z^2-1.959z+0.9864)}{(z^2-1.882z+0.9019)}. \quad (1)$$

Performance metrics (provided in each subcaption) are discussed in Subsection II-A and can be used to compare tracking results. That said, it is more important to note the deficiencies of the results in Fig. 4, and how ILC might be able to correct them. In particular, notice the repeatable peak error in Fig. 4(a), and repeatable error over each raster period in Fig. 4(b). Both of these repeated errors are also obscured by noise which can challenge the ILC results. Attempts at reducing these errors through more accurate models have helped somewhat. However, the model can vary slightly as we move about the range of the stage, and we will see in Section IV that having an accurate model for one particular operating point is not beneficial for all operating points.

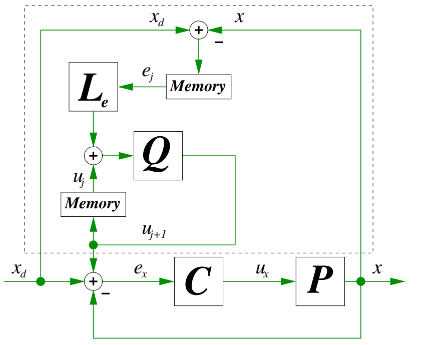


Fig. 3. A single-input single-output (SISO) block diagram representing an application of serial ILC. The dotted line encloses the ILC system that has been added to a standard feedback control loop. L_e and Q are learning filters. This series ILC architecture is a cousin of the FFCLI architecture of Fig. 1. Here, e_j, u_j, and u_{j+1} are vectors.

This plant variation can be seen in Fig. 5 in which we plot seventeen distinct frequency response functions (FRFs) measured over the range of the x direction of the AFM while using the swept-sines system-identification technique. This figure provides a bound on plant uncertainty.

A. Performance Metrics

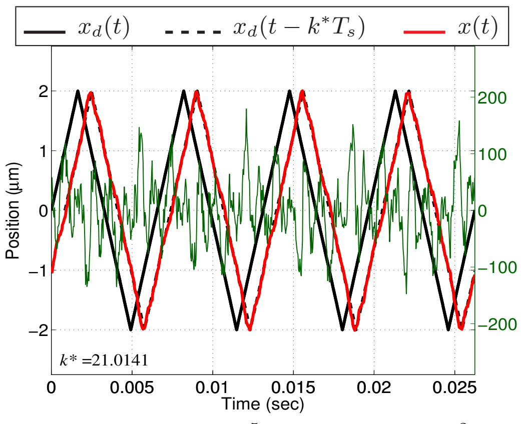
When discussing the performance of the tracking of a raster scan in AFMs, it is important to recall the overall goal of AFMs: to create a quality image in a timely manner [14]. But this goal requires a definition of a "quality image" when referring to a x-y raster scan. Focusing on the x direction, an ideal controller would cause the system output $x(t)$ to track the desired raster pattern $x_d(t)$ flawlessly. This suggests that a mean-square-error metric might be informative in defining the performance of a controller. However, *for imaging*, mean-square error might not give us the best measurement of tracking quality. This is because delay in the raster scan used for AFM imaging is not nearly as critical as consistently tracking the magnitude. Ultimately, this means that if we know the delay well, perfectly delayed tracking is better than imperfect timely tracking. So, to identify that delay, we examine several periods T of the recorded raster scan after a time t_{ss} at which all transients (from initial conditions for example) have died out, and define the variable k^* as

$$k^* = \arg\min_k \int_{t_{ss}}^{t_{ss}+NT} \big(x_d(t-kT_s) - x(t)\big)^2 dt. \quad (2)$$

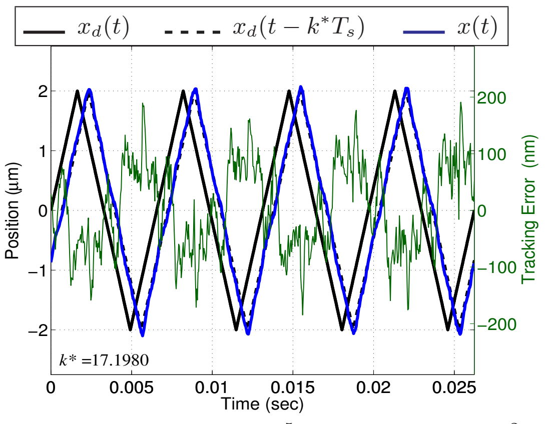
Here, k is a variable defined on $[0, \frac{T}{T_s}]$ where $T_s = 40$ μsec is the controller sample period, and N is the number of recorded periods after transients have died out. Specifically, k^* is the value in $x_d(t-k^*T_s)$ that defines the amount of delay that minimizes the mean square error with respect to the output $x(t)$. k^* can be used to define two metrics that emphasize magnitude tracking over ten periods:

$$J_e = \frac{1}{10}\int_{t_{ss}}^{t_{ss}+10T} \big(x_d(t-k^*T_s) - x(t)\big)^2 dt \quad (3)$$

$$J_m = \max_{t\in[t_{ss},t_{ss}+10T)} \big(x_d(t-k^*T_s) - x(t)\big)^2 \quad (4)$$



(a) FFPI: $J_e = 2.29 \times 10^{-5}$ & $J_m = 3.77 \times 10^{-2}$



(b) FFCLI: $J_e = 3.87 \times 10^{-5}$ & $J_m = 5.25 \times 10^{-2}$

Fig. 4. Experimental results for (a) model-inverse adaptive-delay FFPI and (b) model-inverse FFCLI control when given a 152.4 Hz raster pattern with a $\pm 2\ \mu m$ amplitude. Three curves appear in each figure: $x_d(t)$, $x_d(t-k^*T_s)$, and $x(t)$. The metric values for each experiment are provided in the corresponding sub-captions (closer to zero is superior), and the value for k^* is provided in the lower left of each sub-figure. The tracking error, $x_d(t-k^*T_s)-x(t)$ is provided on a secondary axis in green; it has an RMS of 0.0597 and 0.0767 for FFPI and FFCLI, respectively.

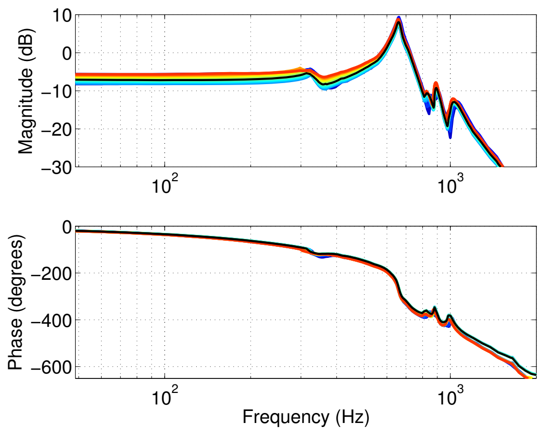


Fig. 5. Seventeen distinct FRFs measured over the range of the x direction of a nPoint NPXY100A x-y piezoscanner stage while using the swept-sines system identification method. Each FRF was generated while injecting $\pm 5\ \mu m$ amplitude sinusoids; they each differ by their offset value within the range of the stage. The thin black line represents the "centered" FRF measured at 0 μm on the $\pm 50\ \mu m$ range.

In general, J_e can be considered to be the mean-square error averaged over ten raster periods while J_m is the max error over those ten periods. When comparing metrics from two different experiments, smaller values of J_e and J_m indicate superior performance.

For further clarification, Fig. 4 shows experimental results provided in in which the actual 152.4 Hz raster scan input $x_d(t)$ is solid black, and the shifted 152.4 Hz input $x_d(t-k^*T_s)$ is dashed black. The system output $x(t)$ is shown in either solid red or blue. On a green secondary axis, we also provide the tracking error $x_d(t-k^*T_s)-x(t)$.

III. The ILC Design

Both serial and parallel ILC architectures [12] are also investigated for use on our AFM nanopositioner. We chose to implement a 1st-order ILC law [12],

$$u_{j+1} = Q(u_j + L_e e_j), \tag{5}$$

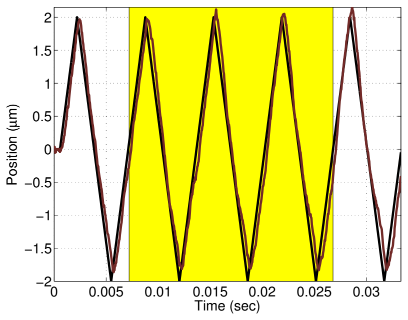
as it is a natural extension to our previous work in that L_e can be designed as the inverse of our plant or closed-loop system as appropriate. In (5), j is the learning iteration index and Q and L_e are learning filters designed by the user. e_j is the error *vector* of the j-th iteration while u_j is the j-th iteration ILC command *vector*. Generally, Q is a low-pass filter, and L_e is the inverse of the system to be learned. Learning is done over a repeated trajectory (in this case a few periods of the raster pattern), and (5) is calculated between each learning iteration in order to determine the next iteration's ILC command vector, u_{j+1}. In short, each new ILC command u_{j+1} is calculated in a batch process. Despite the fundamental architectural differences, the basis of the ILC algorithm is the same regardless of the serial or parallel application. Fundamental differences arise in the design of L_e, and the point of injection of the ILC command signal. We next discuss the parallel ILC and FFPI architectures.

A. Parallel ILC and FFPI

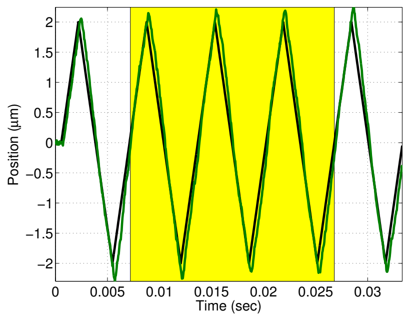
When using a 1st-order ILC law as in (5), a good first design choice for L_e is the plant inverse, $P^{-1}(z)$ [12]. However, this results in a noncausal L_e when $P(z)$ is not exactly proper. As with the FFPI F_P design, additional delay must be added in order to make L_e causal. Unfortunately, when using such a L_e filter, the learning procedure failed to converge. We found success using noncausal learning with 14 samples of "preactuation". Recall that noncausal learning is possible as the calculation of (5) is a batch process in which the full data vectors e_j and u_j are available.

In the process of tuning the learning algorithm, the filter Q was eventually chosen to be a 4th-order Butterworth low-pass filter with a cut-off frequency of 3.25 kHz. A summary of the learning results are provided in Figs. 6(a) and 7. The algorithm took 21 iterations to converge and learn the 5-period 152.4 Hz raster pattern.

With respect to implementation, there is some inconsistency with the length of this reference signal to be learned. In particular, most AFM image scans will require on the



(a) Parallel ILC



(b) Serial ILC

Fig. 6. A summary of the experimental ILC learning for the parallel and serial arrangement. Here, the 154.2 Hz reference signal $x_d(k)$ is black, and $x_f(k)$, the final output after 30 and 10 (respectively) iterations of learning is brown and green, respectively. Learning was done over 5 periods. The shaded yellow areas indicate the raster periods to be averaged to create a final learned ILC feedforward command.

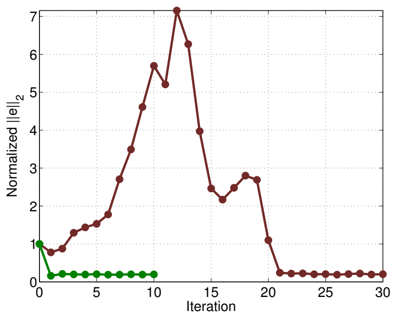


Fig. 7. A plot of the normalized error $||e||_2$ as a function of learning iteration for the ILC learning in the parallel (brown) and serial (green) arrangement. The error was normalized to the error at the first iteration (when using only the feedback controller).

order of 100 or more raster periods to obtain an image, but here, we are only learning over 5 periods. This is because learning over 100 raster periods would exhaust the available memory of the system and take too long to converge. This problem can be circumvented if we take advantage of the periodicity of the reference and assume that there will not be any further disturbances interacting with the raster over the course of multiple periods.

This can be done by ignoring the first and last periods of the learned command input, collating the remaining periods and averaging them. Chopping off the first and last periods of the learned command input help avoid any issues with transients and zero-padding from the noncausal learning. In this particular case, the raster periods to be averaged are shaded in yellow in Fig. 6. The result is a vector describing the average learned command input over the course of one period that can be accessed in a look-up-table by a microprocessor for as many times as the learned raster pattern is required. This process is described by Fig. 8(a).

The inclusion of averaging has the added advantage of averaging out some variation and noise. Noise, in particular is difficult for ILC to address as it is not repeatable. Clearly, learning more than five raster periods would improve the effectiveness of averaging out noise and variation, but there is a balance to consider. First, more raster periods in the learning signal require more memory and computation time. Second, longer learning signals require more learning iterations. Third, it appears that when using ILC, the more we "push" the performance of the nanopositioner with respect to frequency, the shorter the learning signal it is able to learn.

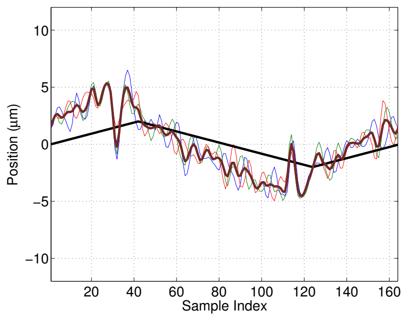
The result of injecting the average learned command input of Fig. 8(a) can be seen in Fig. 9(a), and should be compared with the FFPI results of Fig. 4(a). With respect to our performance metrics, the standard FFPI architecture beats the learned ILC results. Further, the tracking error is not reduced.

While, the "fixed" feedforward control of the FFPI architecture outperforms the ILC results, the FFPI architecture is functioning under circumstances that show it in the best light. In particular, it is functioning in the operating region of the identified plant model. We expect ILC to perform better outside of this range. In the next Subsection, we discuss the series ILC and FFCLI architectures.

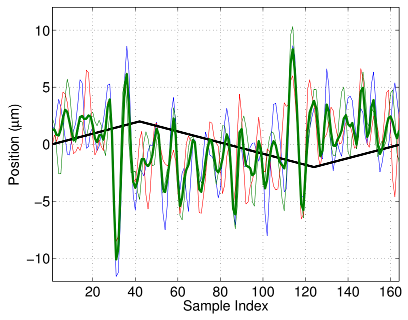
B. Serial ILC and FFCLI

As in the parallel case, a 1st-order ILC law was used for learning in the series architecture, where L_e is designed to be an approximate inverse of the closed-loop system rather than of the plant. Since $C(z)$, the feedback filter is minimum phase, stable, and exactly proper, we were able to use a noncausal ILC algorithm with the same 14 samples of "preactuation". Q was again tuned as a 4th-order Butterworth low-pass filter with a cut-off frequency of 4.25 kHz. A summary of the learning results are provided in Figs. 6(b) and 7.

The dynamics of learning in the serial ILC architecture in Fig. 7 are different from parallel ILC of the previous section. Compared to the parallel architecture, the series architecture converged faster, but the learning resulted in poorer tracking of the raster pattern (Fig. 9(b)). In general, Fig. 7 indicates the difference in learning in the parallel and serial architectures. Specifically, we note that the learning dynamics of the parallel architecture is more complicated because it is searching for balance with $C(z)$. In contrast, the learning dynamics of the serial architecture is similar to applying ILC to a stable open-loop system.

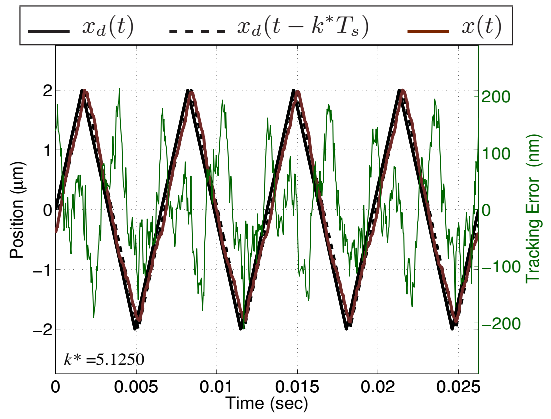


(a) Parallel ILC

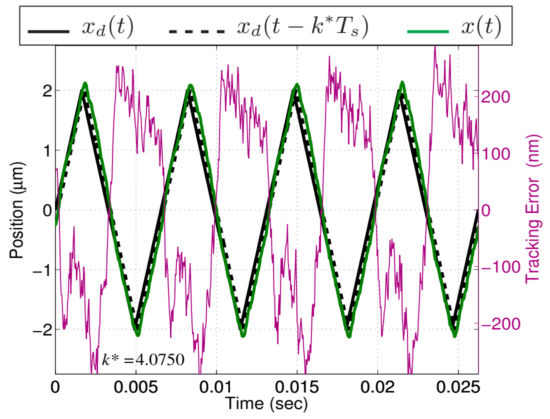


(b) Serial ILC

Fig. 8. A plot showing $u_{j:ave}(k)$, the average learned command input (bold brown and green, respectively) of the parallel and serial ILC architectures over one raster period. Averages were calculated from the learned ILC command input with the first and last raster periods removed; the shaded yellow areas in Fig. 6 indicate this area. The colored curves behind the average are the signals prior to averaging. Note the variation in these unaveraged signals. For comparison, these plots have been plotted on the same scale and $x_d(k)$ is provided in black.



(a) Parallel ILC: $J_e = 4.70 \times 10^{-5}$ & $J_m = 4.51 \times 10^{-2}$



(b) Serial ILC: $J_e = 1.92 \times 10^{-4}$ & $J_m = 1.10 \times 10^{-1}$

Fig. 9. Experimental results for injecting the average learned parallel and serial feedforward command input for a 152.4 Hz raster pattern with a ± 2 μm amplitude. Similar to Fig. 4, three curves appear in the figure: $x_d(t)$, $x_d(t-k^*T_s)$, and $x(t)$. The tracking error, $x_d(t-k^*T_s)-x(t)$ is provided on a secondary axis in green (parallel) and purple (serial); it has an RMS of 0.0843 and 0.1705 for parallel and serial, respectively.

As with the parallel architecture, we averaged the learned ILC command input (Fig. 8(b)). This step revealed signal variation issues of the series architecture. As a result, it is difficult to say that the average signal in Fig. 8(b) is a good representative of the signal the system actually requires. The variation of the signals of Fig. 8(b) also indicates a chance of numerical sensitivity issues that we have seen before with our system under architectures of this style.

The standard fixed FFCLI architecture performs better than the learned ILC results. This is not surprising based on the struggles of the serial ILC learning. It is interesting to note that the overshoot in Figs. 6(b) and 9(b) looks similar to the overshoot that has been seen when using the FFPI architecture *without* the adaptive-delay algorithm described in [1] and used here with the FFPI results.

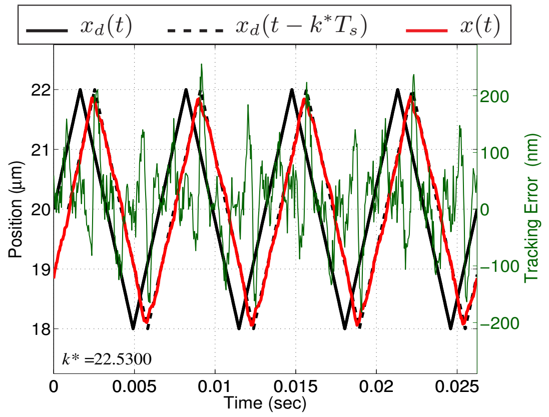
IV. In the Presence of Model Uncertainty

From the results, it is clear that ILC struggles to beat the model-inverse-based FFCLI and FFPI methods when the nanopostitioner is operating in the range at which the plant model was identified. However, Fig. 5 suggests that the results will deteriorate if we offset the raster away from the 0 μm centerline. Here, we leave behind the FFCLI and series methods and focus on FFPI and parallel methods, we shift the raster pattern by 20 μm in Fig. 10 and note the increase in J_e, J_m, and tracking error relative to Fig. 4(a).

Fig. 5 describes the variation in the system FRF due to changes such as these, and our system model is no longer accurate enough to be used for model-inverse-based feedforward control. ILC is not affected by such variations and the results in Fig. 11 show the parallel ILC architecture accommodating the shift and beating the performance of the shifted FFPI results in Fig. 10. As the "centered" results of Figs. 4(b) and 9(b) suggest, both the serial ILC and FFCLI architectures struggle at $+20$ μm.

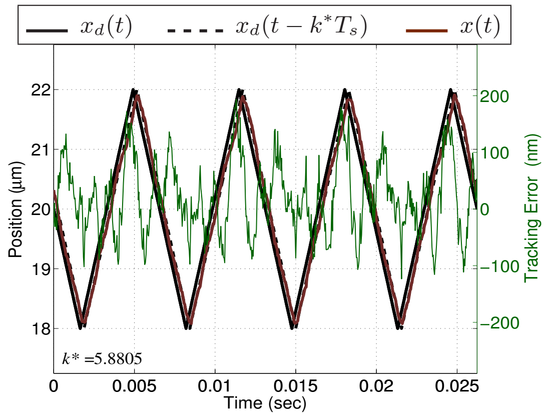
V. Conclusions and Future Work

We investigated the use of the serial and parallel ILC architectures on an AFM nanopositioner. This investigation is a natural extension and comparison to previous work using model-inverse based feedforward control in the FFPI and FFCLI architectures. In general, the parallel ILC architecture always outperformed the serial ILC architecture, although it did take longer to converge.



(a) $+20\mu$m FFPI: $J_e = 4.09\times10^{-5}$ & $J_m = 7.02\times10^{-2}$

Fig. 10. Experimental results for the adaptive-delay algorithm tracking of a 152.4 Hz raster pattern offset by 20 μm. The tracking error, $x_d(t{-}k^*T_s){-}x(t)$ has an RMS of 0.0780. We note the degradation in performance when compared to Fig. 4(a) that can be attributed to known system variations associated with offset changes such as those shown in Fig. 5.



(a) $+20\mu$m Parallel ILC: $J_e = 3.09\times10^{-5}$ & $J_m = 4.39\times10^{-2}$

Fig. 11. Experimental results for injecting the average learned parallel feedforward command input for a 152.4 Hz raster pattern offset by 20 μm. The tracking error, $x_d(t{-}k^*T_s){-}x(t)$ has an RMS of 0.0680. We note that in the presence of model uncertainty, ILC can beat the pure model-inverse-based feedforward controller results of Fig. 10.

Results comparing the serial ILC architecture with FFCLI and the parallel ILC architecture with FFPI indicates the superiority of the nonlearning methods when implemented in a region which reflects the same range at which a model for the plant was identified. However, given the plant variation described by Fig. 5, the ILC approach outperforms the model-inverse methods when operating in regions where the identified plant model used is not fully accurate.

As with the FFPI and FFCLI architectures, there is theoretical equivalence between the parallel and serial ILC architectures once learning is complete. For example, the learned command signal u_{j+1} for the serial architecture can be filtered such that it can be applied in the parallel architecture (just after the feedback controller). In practice this will likely not provide the same performance as this equivalence does not capture the learning dynamics, nor does it reflect the amount of noise in the learned command signal in Fig. 8. If Q for parallel and serial ILC is the same (which is not the case in the work here), the expressions describing equivalence are simplified considerably. In future work, we will explore a design for the feedback controller that may be more appropriate for each ILC architecture.

Another item to note when comparing Fig. 9(a) and Fig. 4(a) is the difference in delay of the output $x(k)$ with respect to the input $x_d(k)$. As discussed in Subsection II-A, delay is not a concern as long as the magnitude of the signal is tracked well. That said, we investigated the use of a time shift in the parallel ILC signals such that learning can be done in such a way that ILC's tendency to correct for time delay is not required. The performance gain was negligible, but the result did emphasize the use of the feedback filter more. This may be beneficial in some applications, but does not appear to be in this one.

References

[1] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "Adaptive-delay combined feedforward/feedback control for raster tracking with applications to AFMs," in *Proc. Amer. Ctrl. Conf.*, June 2010, pp. 5738–5744.

[2] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "A comparison of control architectures for atomic force microscopes," *Asian J. Ctrl.*, vol. 11, no. 2, pp. 175–181, Mar. 2009.

[3] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "A single-parameter combined feedforward/feedback adaptive-delay algorithm with applications to piezo-based raster tracking," *submitted to: IEEE Trans. Ctrl. Sys. Tech.*, 2010, *preprint available at:* `http://gemini.colorado.edu/∼butterwo/`.

[4] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "Analysis and comparison of three discrete-time feedforward model-inverse control techniques for nonminimum-phase systems," *to be submitted to: Mechatronics, A Journal of IFAC*, 2011, *preprint available at:* `http://gemini.colorado.edu/∼butterwo/`.

[5] D. Croft and S. Devasia, "Vibration compensation for high speed scanning tunneling microscopy," *Rev. Sci. Instr.*, vol. 70(12), pp. 4600–4605, 1999.

[6] D. Croft, G. Shedd, and S. Devasia, "Creep, hysteresis, and vibration compensation for piezoactuators: Atomic force microscopy application," *ASME J. Dyn. Sys., Meas., & Ctrl.*, vol. 123, pp. 35–43, 2001.

[7] G. Schitter, R. W. Stark, and A. Stemmer, "Fast contact-mode atomic force microscopy on biological specimen by model-based control," *Ultramicroscopy*, vol. 100, pp. 253–257, 2004.

[8] G. Schitter and A. Stemmer, "Identification and open-loop tracking control of a piezoelectric tube scanner for high-speed scanning-probe microscopy," *IEEE Trans. Ctrl. Sys. Tech.*, vol. 12, pp. 449–454, 2004.

[9] Q. Zou and S. Devasia, "Preview-based optimal inversion for output tracking: Application to scanning tunneling microscopy," *IEEE Trans. Ctrl. Sys. Tech.*, vol. 12, no. 3, pp. 375–386, May 2004.

[10] B. P. Rigney, L. Y. Pao, and D. A. Lawrence, "Nonminimum phase dynamic inversion for settle time applications," *IEEE Trans. Ctrl. Sys. Tech.*, vol. 17, no. 5, pp. 989–1005, Sept. 2009.

[11] B. P. Rigney, L. Y. Pao, and D. A. Lawrence, "Nonminimum phase adaptive inverse control for settle performance applications," *Mechatronics*, vol. 20, pp. 35–44, 2010.

[12] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Ctrl. Sys. Mag.*, pp. 96–114, June 2006.

[13] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 2nd ed. John Wiley and Sons, Ltd., 2005.

[14] D. Y. Abramovitch, S. B. Andersson, L. Y. Pao, and G. Schitter, "A tutorial on the mechanisms, dynamics, and control of atomic force microscopes," in *Proc. Amer. Ctrl. Conf.*, July 2007, pp. 3488–3502.