Receding Horizon Linear Quadratic Tracking Design and Implementation: A Practical Study on a Dual-stage Piezoactuator

Yuhe Chang[a], and Sean B. Andersson[a,b]
[a]Department of Mechanical Engineering, [b]Division of Systems Engineering,
Boston University, Boston, MA 02215 USA
{yuhec, sanderss}@bu.edu

***Abstract*— In this work we compare three practical versions of a receding horizon Linear Quadratic Tracking (LQT) controller in terms of practical considerations such as computational speed and online storage requirements. This work is motivated by the need for effective tracking control of one stage of a dual-stage piezoelectric actuator in non-raster scanning for atomic force microscopy; this application sets challenging requirements on the needed sample rates and tracking bandwidth but also implies that the signal to be tracked is periodic. The first and second versions of our LQT controller compute all relevant values offline, storing what is needed for replay at runtime, with the second version using the steady-state values of the controller gains and the first version using the optimal, finite horizon values. The third approach computes the feedforward term that is based on the reference signal at runtime, requiring more computation than the first two but providing significant flexibility in implementation since the reference can be changed during a run. However, the online computation implies that only a limited planning horizon can be handled. These approaches are compared through simulation. The third method was implemented and validated on a physical dual-stage system.**

I. Introduction

The atomic force microscope (AFM) measures mechanical and material properties at the nanometer-scale, including topology, material moduli, and surface potential [1]–[4]. The standard approach to construct an image of the signal of interest is to raster the tip of the AFM over the sample surface, constructing the image pixel-by-pixel. This leads to slow frame rates, typically well below one frame per second. There are in general three classes of approaches to yield high-speed AFM (HS-AFM) [5]: (1) improving system dynamics, (2) using advanced controller designs, and (3) employing alternative scan paths. From this last category, the authors have previously introduced the Local Circular Scan (LCS), illustrated in Fig. 1(b), a scheme that drives the cantilever tip in a small circle, using the measurements in real time to center the circle on and track a sample feature such as an edge [6]. This approach improves the imaging rate without increasing scanning speed or altering the underlying dynamics of the system. While LCS can be applied to standard AFMs, the pattern is especially well-suited for a particular actuator structure known as a dual-stage system.

Dual-stage actuators (DSAs), Fig. 1(a1-a3), consist of a low-speed, long-range actuator (LRA) connected serially with a high-speed, short-range actuator (SRA). This combination allows the dual-stage system to perform a long range, high bandwidth, and high resolution motion. The concept of DSA is widely implemented in hard disk drives (HDDs), optical alignment systems, and probe-based microscopes [8]–[11]. Combined with LCS, the LRA can be utilized to track the sample path, while the SRA is dedicated to track the low-amplitude, high-frequency sinusoidal signal.

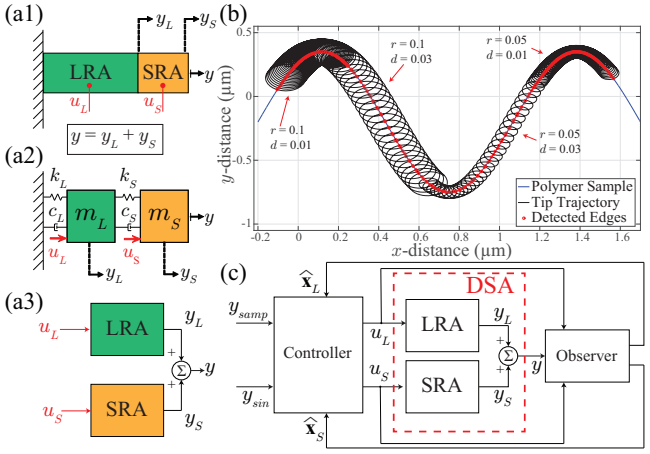


Fig. 1: Dual-stage actuator (DSA) concept: (a1) schematic of the long-range actuator (LRA) and short-range actuator (SRA) connected in series, (a2) lumped-parameter model where the SRA is attached to the LRA mass element, and (a3) dual-input single-output block diagram. (b) Local circular scan (LCS) algorithm concept, where the coarse path is defined by the sample of interest and low amplitude high-frequency circles are executed along the sample. (c) Proposed control structure for a single DSA. Figured reused from [7] with permission (©[2021] IEEE).

Despite a single, combined output, we have shown that DSAs are both observable and controllable [12]. Therefore, an observer can be designed to provide estimates of the actuator states in order to deploy controllers for the LRA and SRA (Fig. 1(c)). With this structure, a variety of controllers can be chosen, including simple proportional-integral-derivative (PID) controllers, complementary filters, and optimal controllers such as robust H_∞ control and model predictive control (MPC) [9], [10], [13], [14]. These controllers will of course have different performance characteristics in terms of behavior, robustness, simplicity, and implementability.

In [7], we developed a DSA controller specific to the

LCS setting that used a receding horizon linear quadratic tracking (LQT) controller for the SRA to accomplish the high frequency, repetitive motion, combined with a model predictive control for the LRA to track the long range, low frequency sample path. The feasibility and tracking performance of this approach were demonstrated through simulation.

In this work, we move from simulation to physical experiment, focusing on the LQT controller for the SRA. Our controllers are deployed on a Field Programmable Gate Array (FPGA) programmed in the National Instruments LabVIEW® environment. When designing the controller for real-world implementation, one must consider several practical factors, including the available computational power relative to the necessary sample rate (especially given the high frequency resonant behavior of the SRA), the amount of memory available for implementing the code, and storage space available when considering, for example, using table-lookup approaches. To explore these considerations, here we consider three different versions of the LQT controller which make different tradeoffs in terms of online resource requirements. We compare their relative performance using simulations and then demonstrate feasibility through experiments with a physical DSA. Note that while we are focused here on the SRA, we also implemented a simple PI controller for the LRA to hold its position constant and reduce interactions between the two systems.

The remainder of this paper is organized as follows. Derivations of the proposed receding LQT controller are provided in Section II. Section III describes simulations selected to compare control performances for three different approaches. The experimental results in Section IV demonstrates the validation of selected method to achieve the receding LQT controller on SRA of a dual-stage nanopositioner. The conclusion of this paper was summarized in Section V.

II. Practical LQT: three approaches

Dual-stage systems are multi-input, single-output (MISO) systems. For simplicity, we model two actuators in the system as independent linear time invariant (LTI) single-input, single-output (SISO) systems with a single joint output. Note that dual stage systems are typically designed to yield strong decoupling between the subsystems. Coupling can, of course, be introduced, but at the cost of a more complex controller design. We take a discrete-time, state space modeling approach. Letting l denote the LRA and s the SRA, the dynamics of the full MISO system of the DSA are

$$\begin{bmatrix}\boldsymbol{x}_l(k+1)\\ \boldsymbol{x}_s(k+1)\end{bmatrix}=\begin{bmatrix}A_l & \mathbf{0}\\ \mathbf{0} & A_s\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_l(k)\\ \boldsymbol{x}_s(k)\end{bmatrix}+\begin{bmatrix}B_l & \mathbf{0}\\ \mathbf{0} & B_s\end{bmatrix}\begin{bmatrix}u_l(k)\\ u_s(k)\end{bmatrix},$$

$$y(k)=y_l(k)+y_s(k)=\begin{bmatrix}C_l & C_s\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_l(k)\\ \boldsymbol{x}_s(k)\end{bmatrix}. \quad (1)$$

Since there is only one joint output for the two actuators, it is necessary to construct an observer to determine the state dynamics of the individual systems. Given that observability and controllability were established in [12], we use a standard discrete-time Luenberger observer given by

$$\hat{\boldsymbol{x}}(k+1)=[A-LC]\hat{\boldsymbol{x}}(k)+B\boldsymbol{u}(k)+Ly(k), \quad (2)$$

where the observer gain L is designed to ensure $A-LC$ is Hurwitz with eigenvalues that are significantly faster than those of the DSA dynamics. Note that the matrices A, B and C represent the joint DSA state-space matrices in (1), and the column vectors $\hat{\boldsymbol{x}}$ and $\boldsymbol{u}$ represent the stacked LRA and SRA states and control signals, respectively.

$$\hat{y}_l=C_l\hat{\boldsymbol{x}}_l, \quad \hat{y}_s=C_s\hat{\boldsymbol{x}}_s. \quad (3)$$

Under LCS, the SRA should follow a simple sinusoidal pattern. To achieve this, we utilize an LQT controller. The LQT controller is an extension of the linear quadratic regulator and is employed when the system output needs to track a desired reference trajectory and reject disturbances while also minimizing the (weighted) control signal. In practice, the control signals are bounded in amplitude by the drive hardware; under the simple sinusoidal pattern, the choice of an LQT is generally sufficient for ensuring the bounds are met through proper choice of the weights in the cost function, even if such a result is not theoretically guaranteed. An LQT controller is designed for a fixed, time horizon of N steps and uses *a priori* knowledge of the reference signal to achieve the tracking [15]–[17]. The cost function in discrete time is given as

$$\min_{u_s}\ \frac{1}{2}\|C\hat{\boldsymbol{x}}_s(N)-r_s(N)\|^2_{\bar{P}} + \frac{1}{2}\sum_{k=0}^{N}\left(\|C\hat{\boldsymbol{x}}_s(k)-r_s(k)\|^2_Q+\|u_s(k)\|^2_R\right),$$

where r_s is a sinusoidal reference signal and $\bar{P}$, Q, and R, are weighting matrices for the terminal state, the tracking performance, and the input effort. (Note that in this work, we take these matrices to be a simple scalar multiplying the identity matrix.) Following the standard derivations, the state feedback control law at the k^{th} step is

$$u_s^*(k)=-K_{fb}(k)\hat{\boldsymbol{x}}_s(k)+K_{ff}(k)v(k+1). \quad (4)$$

The gain matrices and feedforward terms are found by solving the matrix Riccati equation and vector difference equation over the finite time horizon,

$$K_{fb}(k)=[B_s^TP(k+1)B_s+R]^{-1}B_s^TP(k+1)A_s, \quad (5a)$$
$$P(k)=A_s^TP(k+1)[A_s-B_sK_{fb}(k)]+C_s^TQC_s, \quad (5b)$$
$$v(k)=[A_s-B_sK_{fb}(k)]^Tv(k+1)+C_s^TQr_s(k), \quad (5c)$$
$$K_{ff}(k)=[B_s^TP(k+1)B_s+R]^{-1}B_s^T, \quad (5d)$$

with $P(N)=C_s^T\bar{P}C_s$ and $v(N)=C_s^T\bar{P}r(N)$. $K_{fb}(k)$ is the optimal feedback gain at time k, and $K_{ff}(k)$ is a feedforward gain that depends on the auxiliary sequence v determined from the reference signal. Note that (5) is a set of *backward* difference equations.

In practice, there typically is no fixed end time N; this is particularly true in LCS-based AFM imaging where the

extent of the sample being tracked and imaged is not known *a priori*. We thus implement a *receding horizon* LQT with a look ahead of N steps. As is common, we apply the first control in the planned sequence and then re-plan from the next time step. (One could also apply multiple control steps before replanning.) From (4), we see that $u(0)$ depends on the measured state at the current time, the gain matrices $K_{fb}(0)$ and $K_{ff}(0)$ (which in turn depend on the terminal values defined by the weighting matrices), and the auxiliary signal $v(1)$. These can all be solved for either online or offline, backwards from their terminal conditions. The choice of what to do online versus what to do offline has implications in terms of computational complexity and memory requirements. From these considerations we define three approaches to implementing the LQT.

Approach I uses a completely offline approach. This method has the lowest computational cost since it simply takes a measurement from the observer and then inserts that value into (4) to get the next control value. Note that regardless of time horizon, only the values of the gain matrices at $k = 0$ are needed at runtime. Similarly, only the value $v(1)$ of the auxiliary signal is needed. Unlike the gain matrices, however, this term depends on the reference signal at the end of the horizon. In the receding horizon approach, the value of $r_s(N)$ is changing as time moves forward and we must therefore compute and store $v(1)$ for *every* point in the reference signal. For a generic signal, this implies an infinite length sequence and thus a completely offline approach is not viable. In our setting, however, $r(\cdot)$ is a periodic signal. Thus, while the auxiliary signal is not periodic as a function of N, it is periodic in terms of the time during the run with a period equal to that of the reference signal. One need only precompute and store the values $v(1, s)$ for $s = 1, 2, \ldots, N_r$ where s is in index into (one period of) the reference signal and

$$N_r = \frac{f_s}{f_{ref}}. \tag{6}$$

Here f_s is the sampling rate, and f_{ref} is the frequency of the periodic reference signal. So long as N_r is not too large (which depends on the physical hardware used), this is approach is quite feasible. However, when using a high sampling rate (to control, e.g., a high Q system with high-speed resonances) for tracking a relatively slow reference signal, N_r could become quite large, requiring significant onboard resources to store the vector $v(1, \cdot)$. In addition, the values of $v(1, \cdot)$ depend on the reference signal so if this is changed, the entire sequence must be calculated again, implying such a change cannot be made at runtime. However, since the size of the gain matrices $K_{fb}(0)$ and $K_{ff}(0)$ and of the vector $v(1, \cdot)$ do not depend on the horizon N, that horizon can be chosen to be of arbitrary length without any impact at run time.

Approach II is very similar to Approach I but recognizes that, so long as (A_s, B_s) is reachable and $(A_s, C_s\sqrt{Q})$ is observable, the gain matrices K_{fb} and K_{ff} will reach steady state values $K_{fb\infty}$ and $K_{ff\infty}$ when $N \to \infty$. This second method, then, replaces the true optimal gains $K_{fb}(1)$ and $K_{ff}(1)$ with there steady state values. One could then use the steady state matrices in the computation of the auxiliary signal, simplifying the offline computations. However, we found that doing so led to very poor performance and thus Approach II still uses the same $v(1, \cdot)$ as calculated in the first approach. As a result, Approach II has the same online requirements and offline complexity as Approach I; the only difference is in the gains used. Since we are using a receding horizon approach, it is reasonable to conjecture that these infinite-horizon matrices may have better performance.

By contrast, Approach III computes the auxiliary signal online. Unlike the previous two approaches, the available computational power will limit the horizon N. The maximum value of N can be quite low, especially when high sample rates are needed as in the high-frequency, highly-resonant system of the DSA. We see from (5c) that the gains are needed at every step of the horizon, not just at $k = 0$. While these could be computed online, doing so will further limit the size of N. Since N is typically small in this setting, we pre-compute and store these matrices, replaying them at run time to compute the auxiliary signal. In addition, the entire reference signal sequence must be stored. However, this is a scalar at each point in time (over N_r time points) and thus the required space is not onerous. One major benefit of this approach is that the reference signal can be changed at runtime as needed, provided some often useful flexibility to an implementation.

Note that one could imagine several other variations on these three approaches. One other natural choice is a fully online computation of all terms. This would use the least amount of memory but would also require significant online resources; we found that in our setting such an approach was not practical on the physical hardware we had. Use of the infinite horizon gains to compute the auxiliary signal online is a reasonable idea but we found it to work quite poorly in practice as the auxiliary signal was then far from the true one.

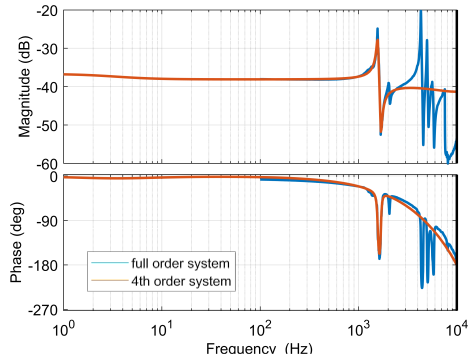
Table I summarizes the description of the the three approaches. In the table, p is the dimension of the state space system. Note that the storage requirements does not include the common elements such as the state matrices since those are needed by all three approaches.

TABLE I: Qualitative comparisons of the three approaches

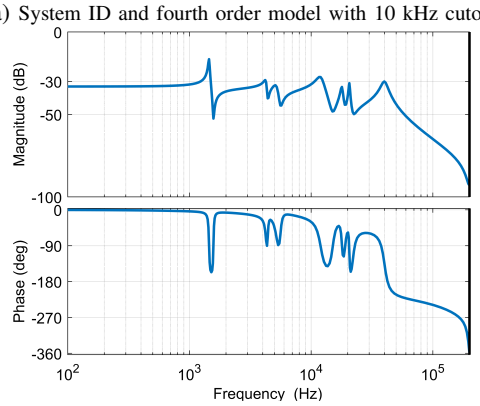
Method	Storage	Computation	Flexibility
Approach I	$pN_r + 2p$	Low	Strict
Approach II	$pN_r + 2p$	Low	Strict
Approach III	$N_r + 2pN$	High	Flexible

III. Simulation comparison of LQT approaches

In this section, we compare the performances of the three different approaches, focusing on tracking error as the metric of interest. We do this through simulations, using a model of a multi-axis, dual-stage system in our lab [18] (see Sec. IV for more details on the physical device.)



(a) System ID and fourth order model with 10 kHz cutoff.



(b) Bode plot of 15^{th} order model, provided by our collaborators.

Fig. 2: Bode plot of SRA system models. (top) (blue) Swept-sine system ID (to 10 kHz) using on-board capactive sensor with 20 kHz bandwidth. (red) corresponding fourth order model used for control design. (bottom) 15th-order model (fit to swept-sine data captured with a laser vibrometer) used to model the device in simulations.

A. Simulation setup

As our interest here is in using the SRA to realize the sinusoidal trajectory element of the LCS imaging method, we consider only the SRA. (Under the assumption of the controller structure shown in Fig. 1, the controllers for the SRA and the LRA of the DSA can be designed independently.) A swept-sine system identification was performed on the physical stage in our lab (see Sec. IV); the results are shown in the blue curve in Fig. 2a. In order to capture the primary dynamics while limiting the complexity of the controller, a fourth order model was fit to this data (red curve in the Bode plot). This model was given by

$$A = \begin{bmatrix} 2.8514 & -3.0203 & 1.2755 & -0.1068 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \; B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 0.0096 & -0.026 & 0.0258 & -0.0094 \end{bmatrix}.$$

To better capture the physical system, a more accurate 15^{th} order model (provided by our collaborators and shown in Fig 2b) was used to model the actuator in this simulation comparison. (Details on the high order model can be found in [18].) Note that the 15^{th} order model used in the simulation and the 4^{th} order system used to design the controller show significant differences, most strikingly in the low frequency gain. We decided to keep this difference in place in the simulation to explore performance in the face of modeling error.

To determine the underlying states of the SRA from the output we implemented a standard Luenberger observer with eigenvalues selected to be significantly faster than the main actuator dynamics. The corresponding observer gains were

$$L = \begin{bmatrix} 1529 & 2192 & 2720 & 3008 \end{bmatrix}^T. \tag{7}$$

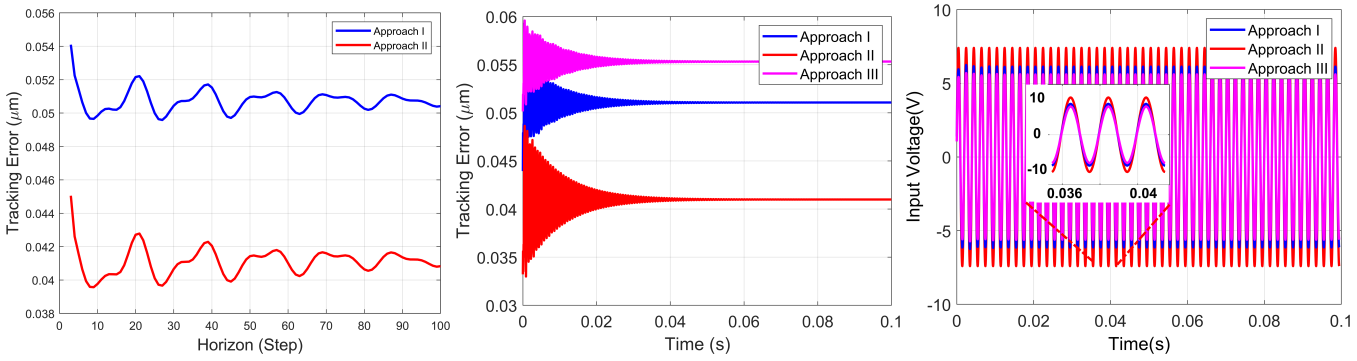
For all LQT implementations, the weights for the cost function were set to $R = 1$, $Q = 20,000$, and $\bar{P} = 0$. This choice reflected the fact that the main consideration was tracking error; the simple sinusoidal reference was not expected to require large control excursions so R was kept low relative to Q; if actuator bounds are a stronger consideration a different relative weighting should be selected. The choice of a zero weight on the terminal condition was in large part to simplify the implementation. In the simulation we also enforced an input voltage boundary of ± 10 volts to represent the limitations of the physical hardware. Note that the controller design assumed unconstrained control. The reference signal was set to

$$r_{x,SRA} = 0.1 \sin\left(2\pi 500 \Delta t\right), \tag{8}$$

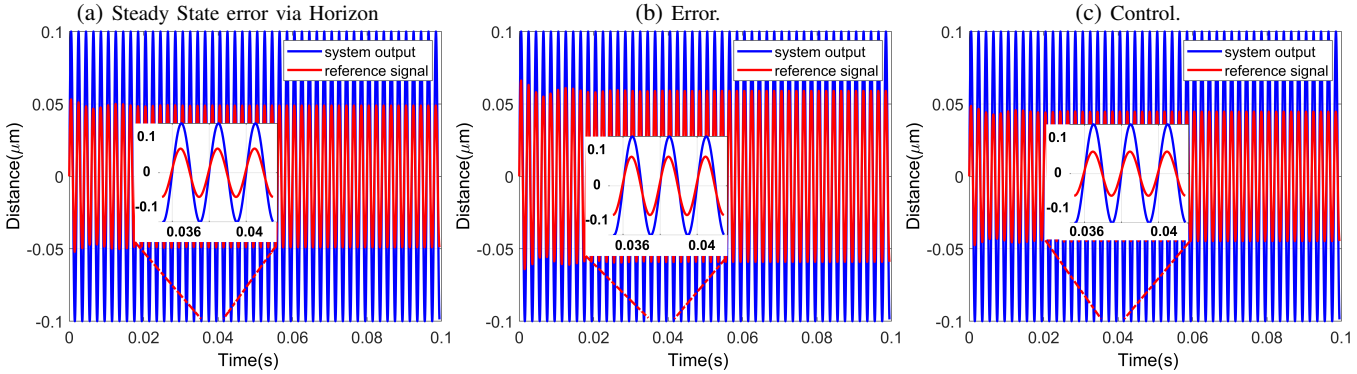
where Δt is the sampling rate (50 μs). With an eye towards the capabilities of our physical hardware (see Sec. IV), the look-ahead horizon for Approach III was set to $N = 3$. For Approaches I and II, recall that the horizon can be set arbitrarily long as its choice has no effect on the runtime complexity of the control implementation. Since there is no guarantee that longer is better, we ran a series of simulations where we increased the horizon N. The results, shown in Fig. 3a, indicate a rapid reduction in tracking error with increasing horizon up to approximately $N = 10$. After that the error essentially saturates, though with some oscillatory behavior. As a result of these studies, we selected $N = 10$ for approaches I and II.

In Fig. 3 (b-f) we show the results of simulation runs for all three approaches. A large tracking error (defined as the amplitude error only since the phase error is irrelevant to the LCS application) is evident. This likely arises in larg part efrom the model mismatch (see Fig. 2). Fig. 3e illustrates that the infinite horizon gain matrices improve the tracking performance in Approach II. From Fig. 3b, we see that Approach II has a slightly longer convergence rate than the other two. Note that the amplitude error for all systems could be reduced by using a more accurate model, increasing Q, or by including integral action [19]. However, in practical systems there will always be modeling error and other issues that must be considered, such as actuator constraints, when attempting to overcome it.

From this exploration, it is clear that there is not a clear "best" choice of approach for LQT implementation. Approach II, which uses the steady-state gain matrices, performed well in this simulations, while requiring the same amount of computation time and storage space as Approach



(a) Steady State error via Horizon (b) Error. (c) Control.



(d) Approach I (e) Approach II (f) Approach III

Fig. 3: Simulation results with reference signal (8). (a) Steady state error of (blue) Approach I and (red) Approach 2 as a function of the look-ahead horizon. (b) Control signal for each approach. All methods stay within the $\pm$ 10 V boundary. (c) Convergence of error signal for each approach, showing different convergence rates. (d-e) Output of the SRA under each approach.

I, though with a slight tradeoff in convergence speed. Approach III had essentially the same dynamics as Approach I but with somewhat worse error due to the limited horizon. However, it has the benefit of requiring less storage (at least for reference signals with periods that are much slower than the sample rate) and flexibility of implementation. Guided by these factors, we selected this approach for implementation as described in the next section.

IV. Experimental Results

To validate our results, we used LQT to control the SRA on one axis of the experimental multi-axis dual-stage nanopositioner shown in Fig. 4 (see [18] for device details). This system consists of two planar piezoelectric DSAs combined with a mechanical flexure mechanism to guide and amplify their motion. Samples can be placed on the central cube and the faces of that cube are used as the measurement surfaces for capacitive sensors (Series 8810, Microsense Technologies, Lowell, MA); these sensors have a 20 kHz bandwidth. They also have a small bias in their measurement, reading a consistent non-zero value when the stage positions are zeroed. The high voltage amplifiers accept input signals from -10 V to 10 V; the SRA has a range of 700 nm and the LRA a range of 13 μm.

We used the same 4^{th} order model, observer gains, and controller parameters as described in Sec. III. Algorithms were implemented on a Field Programmable Gate Array (FPGA) using a multifunction reconfigurable input/output

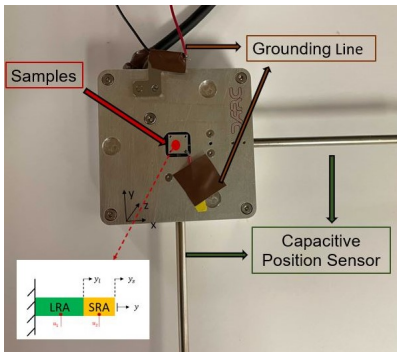
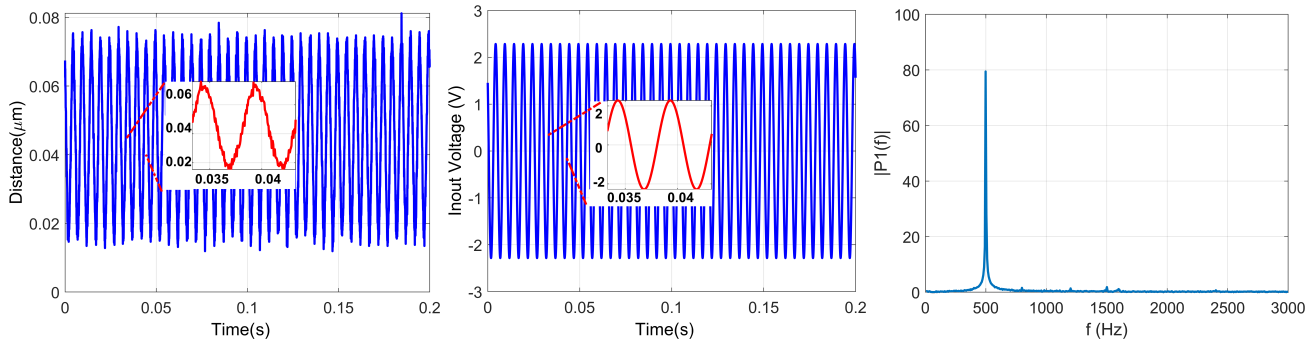


Fig. 4: Experimental multi-axis dual-stage nanopositioner.

device (National Instruments PCIe-7852R), coded through the LabVIEW® environment with a sampling rate of 20 kHz. As discussed at the end of Sec. III, Approach III with a horizon of $N = 3$ was chosen for implementation.

The reference signal (8) and the gain matrices K_{fb} and K_{ff} (5) were computed offline and transferred to the FPGA prior to an experiment. The auxiliary signal $v(\cdot)$ was computed online. It was observed that the SRA dynamics depended on the position of the LRA. To mitigate this effect, a simple proportional-integral controller was implemented and tuned by hand to hold the LRA fixed at zero.

The measured output from a typical run and the corresponding control signal are shown in Fig. 5(a,b) and an FFT of the output in Fig. 5c. The system output shows reasonable



(a) Measured output. (b) Control signal. (c) FFT of output

Fig. 5: Experimental results on a single axis of a DSA using an LQT (Approach III) with a horizon of $N = 3$ on the SRA to track a 500 Hz sinusoidal reference, combined with a PI controller on the LRA to hold it at zero.

tracking, a small amount of noise, and likely some residual oscillations from unmodeled dynamics. The effect of the sensor bias is also clearly evident in the offset in the tracking signal. The amplitude of the output signal is approximately 0.06, significantly below the 0.1 of the reference signal. This is likely caused by model mismatch and indicates the need to do good system identification close to runtime.

V. Conclusions

In this paper, three practical versions of a receding horizon LQT controller were compared in terms of performance as well as implementation limitations. The first used offline computation, allowing for a full optimal implementation with arbitrary horizon but requiring storage of the auxiliary signal; the second used the infinite horizon gains in lieu of the optimal values; the third used online computation of the auxiliary signal, leading to a possible reduction in required memory, depending on the relative values of the sample rate and the period of the reference signal, but an increase in computation required. Perhaps most important, it provided flexibility in changing the reference signal during runtime. All approaches performed reasonably well in simulation, leading to the conclusion that the appropriate choice depends upon the particular application, driven by the required sample rates, available memory, and computing power. The version that computed the auxiliary signal online but used pre-computed, optimal gain matrices was implemented and used to control a physical DSA system to validate the approach, showing good results for tracking a desired sinusoidal reference.

Acknowledgements

The authors thank Kam Leang and William Nagel of the University of Utah for providing the dual stage actuator. This work was supported in part by NSF through CMMI-1661586.

References

[1] D. J. Muller, "AFM: A Nanotool in Membrane Biology," *Biochemistry*, vol. 47, no. 31, pp. 7986–7998, Aug. 2008.

[2] S. Liu and Y. Wang, "Application of AFM in microbiology: a review," *Scanning*, vol. 32, no. 2, pp. 61–73, Mar. 2010.

[3] W. Melitz, J. Shen, A. C. Kummel, and S. Lee, "Kelvin probe force microscopy and its application," *Surface Science Reports*, vol. 66, no. 1, pp. 1–27, Jan. 2011.

[4] K. Haase and A. E. Pelling, "Investigating cell mechanics with atomic force microscopy," *Journal of The Royal Society Interface*, vol. 12, no. 104, pp. 20 140 970–20 140 970, Mar. 2015.

[5] T. Ando, "High-speed atomic force microscopy coming of age," *Nanotechnology*, vol. 23, no. 6, p. 062001, 2012.

[6] B. Hartman and S. B. Andersson, "Feature Tracking for High Speed AFM Imaging of Biopolymers," *International Journal of Molecular Sciences*, vol. 19, no. 4, p. 1044, Mar. 2018.

[7] Y. Chang, W. S. Nagel, K. K. Leang, and S. B. Andersson, "Comparison of two optimization-based controllers for feature tracking spm scanning in dual-stage nanopositioners," in *2021 American Control Conference (ACC)*, 2021, pp. 4315–4320.

[8] Y. Li and R. Horowitz, "Mechatronics of electrostatic microactuators for computer disk drive dual-stage servo systems," *IEEE/ASME Transactions on Mechatronics*, vol. 6, no. 2, pp. 111–121, 2001.

[9] Y. K. Yong, S. P. Wadikhaye, and A. J. Fleming, "High speed single- and dual-stage vertical positioners," *Review of Scientific Instruments*, vol. 87, no. 8, p. 085104, 2016.

[10] T. Tuma, W. Haeberle, H. Rothuizen, J. Lygeros, A. Pantazi, and A. Sebastian, "Dual-stage nanopositioning for high-speed scanning probe microscopy," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, pp. 1035–1045, 2014.

[11] A. Mitrovic, W. S. Nagel, K. K. Leang, and G. M. Clayton, "Closed-loop range-based control of dual-stage nanopositioning systems," *IEEE/ASME Transactions on Mechatronics*, 2020.

[12] Y. Chang and S. B. Andersson, "Observer-based control of a dual-stage piezoelectric scanner," in *Dynamic Systems and Control Conference*, vol. 59162. American Society of Mechanical Engineers, 2019, p. V003T19A008.

[13] A. Elfizy, G. Bone, and M. Elbestawi, "Design and control of a dual-stage feed drive," *International Journal of Machine Tools and Manufacture*, vol. 45, no. 2, pp. 153–165, 2005.

[14] M. A. Rahman, A. A. Mamun, K. Yao, and Y. Daud, "Discrete-time model predictive control for head-positioning servomechanism in a dual-stage hard disk drive," in *IEEE International Conference on Mechatronics and Automation*, 2014, pp. 8–13.

[15] B. D. O. Anderson, J. B. Moore, and B. P. Molinari, "Linear optimal control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-2, no. 4, pp. 559–559, 1972.

[16] H. Modares and F. L. Lewis, "Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 59, no. 11, pp. 3051–3056, 2014.

[17] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[18] W. S. Nagel and K. K. Leang, "Design of a dual-stage, three-axis hybrid parallel-serial-kinematic nanopositioner with mechanically mitigated cross-coupling," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2017, pp. 706–711.

[19] O. Park, H. Shin, and A. Tsourdos, "Linear Quadratic Tracker with Integrator using Integral Reinforcement Learning," in *International Workshop on Research, Education and Development on Unmanned Aerial Systems*, 2019, pp. 31–36.