Tracking Control Using Recurrent-Neural-Network-Based Inversion Model: A Case Study on a Piezo Actuator

Shengwen Xie and Juan Ren, *Member, IEEE*

***Abstract*—This article is concerned with designing a broadband, high accuracy, and computationally efficient real-time controller for piezo actuators (PEAs). The essential component proposed is a recurrent-neural-network (RNN) based inversion model (RNNinv) used to compensate for the PEA nonlinearities. However, the obtained RNNinv may not efficiently model the low-frequency and/or time-varying dynamics of the system due to the limited length of the RNN training set. To address this issue, a linear model embedded with an error term is used to model the low-frequency dynamics in case it is not precisely modeled by RNNinv, and a predictive controller based on this linear model is then designed for precise output tracking. To validate the proposed control framework, the closed-loop stability condition is derived, and the RNNinv stability in unforced mode is investigated. To further improve the accuracy, a mechanism is proposed to separate the controlling dynamics to achieve higher accuracy for applications that cover broad and/or high-frequency ranges. The proposed approach was implemented on a commercial PEA and its performance was demonstrated through comparison with other controllers.**

***Index Terms*—Inversion model, Piezo actuator (PEA), predictive control, recurrent neural network (RNN).**

I. Introduction

PIEZO actuators (PEAs) have been widely used for nanopositioning and are applied in various applications, such as microforming [1], atomic force microscope [2], and vibration control [3]. However, accurate trajectory tracking of PEAs over large bandwidth is quite challenging due to the limited modeling bandwidth of existing techniques [4].

Instead of modeling the nonlinear system dynamics, using the inversion model to compensate for the system nonlinearity is more computationally efficient and also eases the complexities on controller design. Combined with iterative learning control (ILC), the inversion-based ILC can achieve precise output tracking for repetitive tasks [5]. With the nonlinearities compensated by the inversion model, repetitive controllers are developed for trajectory tracking for both single and dual-stage nanopositioning (PEA-based) [6]–[8]. However, both ILC and repetitive controllers are limited to repetitive tasks, i.e., they are not real-time controllers. For nonrepetitive tasks, real-time controllers are needed. Inversion models have been proposed in real-time control of PEAs. For example, the inversion model based on Prandtl–Ishlinskii operators was used to remove the hysteresis effect [9]–[11]. An inversion model based on the ferromagnetic material hysteresis was developed to compensate for the nonlinearities of the PEAs in [12]. However, the highest control bandwidth achieved by the existing inversion models is reported to be around 100 Hz [9]–[12]. Thus, existing inversion models are limited for high-precision and broad-bandwidth trajectory tracking of PEAs.

Neural networks have been used in system identification in order to enlarge the modeling bandwidth. In [13], a dynamic recurrent neural network (RNN) is proposed to compensate for the hysteresis of magnetostrictive actuators, but no experiments results are presented and the reported bandwidth in simulation is less than 30 Hz. Elman Neural Networks have been proven to be effective in modeling hysteresis, but effective controller based on it is not available [14], [15]. In [16], a radial basis function neural network has been proposed to model the input–output relationship; however, the control bandwidth of the sliding mode controller bandwidth is limited. In [17] and [18], two feedforward neural networks (FNN) are combined to model the PEA dynamics. However, FNN is not robust enough to model the dynamical systems in the circumstances where the measurement data are noisy and/or data size is not large enough [19], [20]. In such cases, either more parameters are needed, or the accuracy of the obtained model is not satisfactory. In contrast, it has been proven that the RNN, which considers the sequence of the input and output time series, is effective in system identification [19]. Recently, RNN has been proposed to model the dynamics of PEAs over a large bandwidth with high accuracy [21]. But extra computation power is required when directly using the RNN model in model-based controls, such as model predictive control [21].

Therefore, in order to reduce the computation burden on controller design, we proposed to use RNN to model the inverse

Manuscript received November 5, 2019; revised April 17, 2020, June 23, 2020, and October 12, 2020; accepted November 1, 2020. Date of publication November 19, 2020; date of current version July 19, 2021. This work was supported by the National Science Foundation (NSF)(CMMI-1751503 and CMMI-1634592) and Iowa State University. *(Corresponding author: Juan Ren.)*

The authors are with the Department of Mechanical Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: swxie@iastate.edu; juanren@iastate.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TIE.2020.3037989

Digital Object Identifier 10.1109/TIE.2020.3037989

dynamics and then compensate for the nonlinearities of PEAs. Thus, the RNN inversion model (RNNinv) is not involved in designing controllers for the real-time trajectory tracking task. It is worth noting that the RNNinv may not be able to capture the time-varying dynamics. Also, given the limited length of the RNN training set for the sake of shortening training time, the PEA low-frequency dynamics may not be adequately compensated by the RNNinv. To address this issue, a linear model embedded with an error term (LME) is proposed to deal with the unmodeled low-frequency and/or time-varying dynamics of PEAs, and then a predictive controller is designed based on the LME for output tracking. Furthermore, in case the dynamics of the LME will interfere with that of the RNNinv causing downgraded performance at high-frequency region, a separation mechanism is designed to ensure that the predictive controller based on the LME is only effective at low frequencies.

The advantages of the proposed RNNinv and LME integrated framework (RNNinv+LME) are threefold. First, compared to the previous neural network approaches [17], [18], the computation burden is greatly mitigated since the predictive controller is based on a linear model—LME. Second, the bandwidth of the RNNinv is much higher than existing inversion models. Another advantage is that the modeling accuracy can be further improved with more complex neural networks without worrying about the computation issue, thus better controlling performance can be achieved.

The contributions of this article include the following:

1) using RNN to model the inversion dynamics is proposed, and the issue of RNNinv—low modeling accuracy in low-frequency region is addressed;
2) stability of the RNNinv in unforced mode is investigated and the closed-loop stability condition of the predictive controller is derived;
3) a mechanism of separating the controlling dynamics to further improve the controller performance is designed.

For demonstration, the proposed RNNinv+LME framework was implemented to control the displacement of a PEA stage, and the control performance was compared with that of a PI controller and even one ILC approach—modeling-free inversion-based iterative feedforward control (MIIFC) to demonstrate that the real-time control accuracy achieved by RNNinv+LME is comparable or even better than that of the offline ILC approach.

II. System Dynamics Modeling

A. How the Inversion Model Works

For the following dynamical system (1), suppose $U = [u_k, u_{k+1}, \ldots]$ and $Y = [y_k, y_{k+1}, \ldots]$ are the input and output time series, respectively. The corresponding inversion model can also be of state-space form (2). The difference is that if Y is fed to the inversion model (2), the output of the inversion model becomes U

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= h(x_k, u_k) \end{aligned} \tag{1}$$

$$\begin{aligned} z_{k+1} &= f_{\text{inv}}(z_k, \bar{u}_k) \\ w_k &= h_{\text{inv}}(z_k, \bar{u}_k). \end{aligned} \tag{2}$$

If the inversion model is linear, for instance, $f_{\text{inv}} = Az_k + B\bar{u}_k$ and $h_{\text{inv}} = Cz_k + D\bar{u}_k$, the parameters A, B, C, and D can be determined with the linear system theories. However, when $f_{\text{inv}}(\cdot)$ and $h_{\text{inv}}(\cdot)$ are nonlinear as is the case in this work, one possible way to determine the parameters is to formulate and solve an optimization problem [e.g., (4)] with the output–input pair $\{Y, U\}$ of the original system, which will be explained in detail as follows.

B. RNN Inversion Model

It is clear that a system inversion model is supposed to map one time series (i.e., output of the original system) to another time series (i.e., input to the original system). In this sense, an RNN is more suitable to generate the inversion model since the training input is a time series, because an FNN is not trained with time series as different samples in the training set are not correlated.

In this article, the nonlinear inversion model is an RNN. RNNs have different structures, as an example, we choose the one that has been used previously for modeling the system dynamics [21]. The RNN can be represented using the following nonlinear model:

$$\begin{aligned} x_{k+1} &= \tanh(W_1 x_k + B_2 + B_1 u_{(r),k}) \\ y_{(r),k} &= W_2 x_k + B_3. \end{aligned} \tag{3}$$

The RNNinv takes the PEA trajectory to be tracked as input $u_{(r),k}$, and outputs the desired input $y_{(r),k}$ to the PEA system. Suppose the output (i.e., trajectory) of the PEA system is $Y_{(\text{ts})}$ subject to the drive input $U_{(\text{ts})}$, an ideal inversion model with input $Y_{(\text{ts})}$ should output $Y_{(\text{rts})}$ such as that $||Y_{(\text{rts})} - U_{(\text{ts})}|| < \epsilon$ for any $\epsilon > 0$. Therefore, the pair $(Y_{(\text{ts})}, U_{(\text{ts})})$ can be used to train the RNNinv [i.e., to obtain the parameters W_1, B_2, B_1, W_2, and B_3 in (3)].

To construct the training set $(Y_{(\text{ts})}, U_{(\text{ts})})$, we need to design the time series $Y_{(\text{ts})}$ and consider both the frequency- and amplitude-dependent behavior of PEAs. In this work, the method developed in our previous work [21] is used. However, since we do not have a real inversion of the original system, one can use ILC to find $U_{(\text{ts})}$. Specifically, we can apply ILC such that the designed $Y_{(\text{ts})}$ is accurately tracked by the PEA output, therefore, the corresponding converged ILC input to the PEA is $U_{(\text{ts})}$ [22], [23].

Once the $(Y_{(\text{ts})}, U_{(\text{ts})})$ pair is determined, the RNNinv parameters can be obtained through the training process by solving the following optimization problem:

$$\begin{aligned} \min_{W_1, B_2, B_1, W_2, B_3} \quad & J = ||U_{(\text{ts})} - Y_{(\text{rts})}|| \\ \text{subject to} : \quad & x_{k+1} = \tanh(W_1 x_k + B_2 + B_1 y_{(\text{ts}),k}) \\ & y_{(\text{rts}),k} = W_2 x_k + B_3 \\ & x_0 = [0, 0, \ldots, 0]^T, \ k = 1, 2, 3, \ldots, L \end{aligned} \tag{4}$$

where $U_{(\text{ts})} = [u_{(\text{ts}),1}, u_{(\text{ts}),2}, \ldots, u_{(\text{ts}),L}]^T$, $Y_{(\text{ts})} = [y_{(\text{ts}),1}, y_{(\text{ts}),2}, \ldots, y_{(\text{ts}),L}]^T$, and $Y_{(\text{rts})} = [y_{(\text{rts}),1}, y_{(\text{rts}),2}, \ldots, y_{(\text{rts}),L}]^T$. L is the length of the time series.

Remark 1. Note that $Y_{(\text{ts})}$ may contain sinusoidal signals with different frequencies [21]. If the sampling frequency is $f_s =$

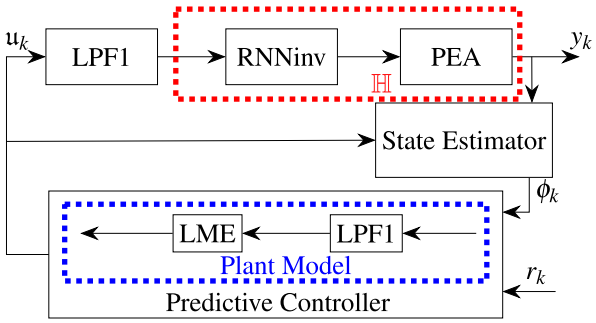


Fig. 1. Schematic block diagram of integrating RNNinv and LME (called "RNNinv+LME").

10 kHz, the number of points to represent a 1 Hz sinusoidal signal in one period is 10 000. However, for a 20 Hz sinusoidal signal, there are only 500 points. Therefore, if a lot of low-frequency sinusoidal signals are included in $Y_{(\text{ts})}$, the length of $Y_{(\text{ts})}$ will be very large, which will dramatically increase the training time. On the other hand, with the large training set it is difficult for the training algorithm to converge to a satisfactory local minimum using the current RNN structure (3). Thus, the current RNN cannot model the low-frequency dynamics accurately, this issue will be addressed next through introducing a linear model.

C. Linear Model Embedded With an Error Term

The obtained RNNinv will be cascaded with the original plant forming a new system $\mathbb{H}$ as shown in Fig. 1. Since it is possible that the RNNinv may not accurately capture the PEA low-frequency dynamics when the length of the training set is limited for the concern of computation efficiency, the LME represented by (5) in the following is proposed to address this issue

$$\begin{aligned} \eta_{k+1} &= A_e\eta_k + B_e\hat{u}_k + G_e\hat{e}_k \\ \hat{y}_k &= C_e\eta_k \end{aligned} \tag{5}$$

where $\hat{u}_k$ is the input to the LME, $\hat{e}_k = \bar{y}_k - \hat{y}_k$ is the model output error with $\bar{y}_k$ as the actual PEA output. The sizes of A_e, B_e, G_e, and C_e are 2×2, 2×1, 2×1, and 1×2, respectively. Note that LME is to model the dynamics of $\mathbb{H}$. The error term can be regarded as a feedback term. Suppose the output of $\mathbb{H}$ is $\bar{y}_{(\text{ts}),k}$ subject to the designed $y_{(\text{ts}),k}$, the output of the LME $\hat{y}_{(\text{ts}),k}$ should be nearly the same as $\bar{y}_{(\text{ts}),k}$. Similar to the RNNinv, the parameters of LME can be obtained through solving the following optimization problem

$$\begin{aligned} \min_{A_e,B_e,G_e,C_e} \quad & J_1 = ||\hat{Y}_{(\text{ts})} - \bar{Y}_{(\text{ts})}|| \\ \text{subject to}: \quad & \eta_{k+1} = A_e\eta_k + B_e y_{(\text{ts}),k} + G_e\hat{e}_k \\ & \hat{Y}_{(\text{ts}),k+1} = C_e\eta_{k+1} \\ & \hat{e}_k = \bar{y}_{(\text{ts}),k} - \hat{y}_{(\text{ts}),k} \\ & \eta_0 = [0,0]^T \\ & \hat{e}_0 = 0, k = 0,1,\ldots \end{aligned} \tag{6}$$

where $\bar{Y}_{(\text{ts})} = [\bar{y}_{(\text{ts}),1}, \bar{y}_{(\text{ts}),2}, \ldots]$, $\hat{Y}_{(\text{ts})} = [\hat{y}_{(\text{ts}),1}, \hat{y}_{(\text{ts}),2}, \ldots]$, and $Y_{(\text{ts})} = [y_{(\text{ts}),1}, y_{(\text{ts}),2}, \ldots]$.

Furthermore, to avoid high-frequency disturbance to be fed into the feedback loop, a low-pass filter (LPF1) is cascaded to the LME to remove the ultrahigh-frequency dynamics (see Fig. 1). Suppose LPF1 can be represented as

$$\begin{aligned} \beta_{k+1} &= \bar{A}\beta_k + \bar{B}\mathfrak{u}_k \\ \bar{z}_k &= \bar{C}\beta_k. \end{aligned} \tag{7}$$

Since system (5) is connected to (7), we have $\bar{z}_k = \hat{u}_k$. Thus, the two models can be combined as the "plant model" shown as follows:

$$\begin{aligned} \phi_{k+1} = \begin{bmatrix} \beta_{k+1} \\ \eta_{k+1} \end{bmatrix} &= \begin{bmatrix} \bar{A} & 0 \\ B_e\bar{C} & A_e \end{bmatrix}\phi_k + \begin{bmatrix} \bar{B} \\ 0 \end{bmatrix}\mathfrak{u}_k + \begin{bmatrix} 0 \\ G_e \end{bmatrix}\hat{e}_k \\ &= A\phi_k + B\mathfrak{u}_k + G\hat{e}_k \\ \hat{y}_k &= \begin{bmatrix} 0 & C_e \end{bmatrix}\phi_k = C\phi_k \end{aligned} \tag{8}$$

where $\mathfrak{u}_k$ is the input. The block diagram of the LME+RNNinv framework is schematically shown in Fig. 1.

Remark 2. Since the RNNinv can already model the PEA high-frequency dynamics with high accuracy, it is expected that if a high-frequency signal X is fed to the LME, the output should be X, too, i.e., the gain of LME in high-frequency region is 1 without phase delay. However, the LME obtained earlier may not satisfy this condition. Therefore, if the predictive controller based on LME is applied directly on the system, the tracking error for high-frequency signal may be pronounced. This issue is addressed in Section V.

III. Controller Design

Since not all the states in (8) can be accurately measured, a Kalman estimator can be used to estimate the states. Interested readers can find details in [24]. Then, a predictive controller is designed based on the linear model (8). The control diagram is shown in Fig. 1.

A. Predictive Control

With the PEA low-frequency dynamics modeled by LME, a predictive controller based on the system model represented by (8) is designed. To facilitate analyzing the closed-loop stability, a modeling uncertainty term $\delta_k = G\hat{e}_k$ is introduced to replace $G\hat{e}_k$ in (8). The system dynamics becomes

$$\begin{aligned} \phi_{k+1} &= A\phi_k + B\mathfrak{u}_k + \delta_k \\ \hat{y}_k &= C\phi_k. \end{aligned} \tag{9}$$

Let $\mathfrak{U} = [\mathfrak{u}_{k+1}, \mathfrak{u}_{k+2}, \ldots, \mathfrak{u}_{k+N_c}]$, $\Delta\mathfrak{U} = [\mathfrak{u}_{k+1} - \mathfrak{u}_k, \mathfrak{u}_{k+2} - \mathfrak{u}_{k+1}, \ldots, \mathfrak{u}_{k+N_c} - \mathfrak{u}_{k+N_c-1}]$, $\Delta_k = [\delta_k^T, \delta_{k+1}^T, \ldots, \delta_{k+N_p-1}^T]^T$, and $\mathbf{1}_n = [1,1,\ldots,1]^T$. Given the current state ϕ_k and input $\mathfrak{u}_k$, the outputs of the system $\mathbb{H}$ in the future N_p steps can be predicted as

$$\begin{aligned} \bar{Y}^p &= G_p\phi_k + H\mathfrak{U} + F\mathfrak{u}_k \\ &= G_p\phi_k + H(S\Delta\mathfrak{U} + \mathbf{1}\mathfrak{u}_k) + H_1\Delta_k + F\mathfrak{u}_k \\ &= G_p\phi_k + HS\Delta\mathfrak{U} + H_1\Delta_k + (F + H\mathbf{1})\mathfrak{u}_k \end{aligned} \tag{10}$$

with

$$\bar{Y}^p = \begin{bmatrix} \hat{y}_{k+1} \\ \hat{y}_{k+2} \\ \vdots \\ \hat{y}_{k+N_p} \end{bmatrix}_{N_p \times 1}, \; G_p = \begin{bmatrix} CA \\ CA^2 \\ \dots \\ CA^{N_p} \end{bmatrix}_{N_p \times 1},$$

$$\mathfrak{U}_p = \begin{bmatrix} \mathfrak{u}_{k+1} \\ \mathfrak{u}_{k+2} \\ \vdots \\ \mathfrak{u}_{k+N_c} \end{bmatrix}_{N_c \times 1}$$

$$H = \begin{bmatrix} 0 & 0 & \dots & 0 \\ CB & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-2}B & CA^{N_p-3}B & \dots & 0 \end{bmatrix},$$

$$F = \begin{bmatrix} CB \\ CAB \\ \vdots \\ CA^{N_p-1}B \end{bmatrix}_{N_p \times 1}$$

$$H_1 = \begin{bmatrix} C & 0 & \dots & 0 \\ CA\mathbf{1} & C & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}\mathbf{1} & CA^{N_p-2}\mathbf{1} & \dots & C \end{bmatrix}$$

$$S = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

where N_p and N_c are the prediction horizon and control horizon, respectively, $I_{N_c \times N_c}$ is an identity matrix. The future inputs $\mathfrak{U}_p$ can be computed in each sample time through minimizing the cost function as follows:

$$\begin{aligned} J &= (\bar{Y}^p - R_p)^T(\bar{Y}^p - R_p) + \rho \Delta\mathfrak{U}^T \Delta\mathfrak{U} \\ &= \Delta\mathfrak{U}^T(\rho\mathbf{I} + S^T H^T H S)\Delta\mathfrak{U} + 2\Delta\mathfrak{U}^T S^T H^T E + E^T E \end{aligned} \tag{11}$$

where $R_p = [r_{k+1}, r_{k+2}, \dots, r_{k+N_c}, \dots, r_{k+N_c}]^T$ is the reference signal, ρ is the weighting coefficient, and $E = G_p\phi_k + H_2\Delta_k + (F + H\mathbf{1})\mathfrak{u}_k - R_p$. Note that convex constraints on the input can also be added, the resulting optimization problem will remain convex thus efficient algorithm can be deployed. The closed-loop stability without adding constraints on the input will be analyzed in the following section.

B. Closed-Loop Stability

To minimize J (11), $\Delta\mathfrak{U}$ should be

$$\Delta\mathfrak{U}^* = \arg\min_{\Delta\mathfrak{U}} J = -\Psi^{-1} S^T H^T E \tag{12}$$

where $\Psi = \rho\mathbf{I} + S^T H^T H S$. Let $E_1 = [1, 0, \dots, 0]$. Thus, the control law will be

$$\begin{aligned} \mathfrak{u}_{k+1} &= \mathfrak{u}_k + E_1 \Delta\mathfrak{U}^* = \mathfrak{u}_k - E_1 \Psi^{-1} S^T H^T E \\ &= M_1\phi_k + M_2 u_k + M_3 R_p + M_4 \Delta_k \end{aligned} \tag{13}$$

where $M_1 = -E_1\Psi^{-1}S^T H^T G_p$, $M_2 = 1 - E_1\Psi^{-1}S^T H^T(F + H\mathbf{1})$, $M_3 = E_1\Psi^{-1}S^T H^T$, and $M_4 = -E_1\Psi^{-1}S^T H^T H_1$, they are constant matrices and only depend on N_p and N_c. The closed-loop system can then be rewritten as

$$\begin{bmatrix} \phi_{k+1} \\ u_{k+1} \end{bmatrix} = \begin{bmatrix} A & B \\ M_1 & M_2 \end{bmatrix} \begin{bmatrix} \phi_k \\ u_k \end{bmatrix} + \begin{bmatrix} 0 \\ M_3 \end{bmatrix} R_p + \begin{bmatrix} 0 \\ M_4 \end{bmatrix} \Delta_k \quad . \tag{14}$$

Therefore, the closed-loop stability condition is

$$\left| \lambda \left(\begin{bmatrix} A & B \\ M_1 & M_2 \end{bmatrix} \right) \right|_{\max} < 1 \tag{15}$$

where $\lambda(\cdot)$ denotes the eigenvalues of the matrix. Furthermore, as can be seen from (14), the modeling uncertainty and/or disturbances Δ_k will not affect the closed-loop stability if it is bounded. It is worth pointing out that the control law is very computationally efficient with time complexity of $\Theta(N_h N)$ (N: order of the model).

IV. Stability of the RNNinv in Unforced Mode

As the RNNinv is proposed to model the inversion dynamics of the PEA system, here we present a numerical method to prove the stability of the RNNinv [i.e., (3)] in unforced mode (i.e., $u_{(r),k} = 0$). The stability of (3) in unforced mode means that given any initial state x_0, the following autonomous system converges to one point, i.e., there exists a unique equilibrium point for the following equation:

$$\begin{aligned} x_{k+1} &= \tanh(W_1 x_k + B_2) \\ y_{(r),k} &= W_2 x_k + B_3. \end{aligned} \tag{16}$$

To prove the stability, first, we narrow the dissipativity domain until it becomes sufficiently small [25]. As a result, proving the global asymptotically stability of (16) is transformed to the problem of proving local stability. Then, the proof can be completed by searching a convex Lyapunov function using linear matrix inequality (LMI) [26]. The details are presented as follows.

For the discrete system $x_{k+1} = f(x_k)$, suppose the equilibrium point is located at the origin. The basic idea of contracting the dissipativity domain is to construct a series of sets D_k such that $D_{k+1} \subset D_k$ and $f(D_k) \subset D_{k+1}$ [25]. Thus, if $x_0 \in D_0$, then $x_k \in D_k$. Therefore, $D_k \to \{0\}$ will lead to $x_k \to 0$ as $k \to \infty$.

D_0 can be constructed with the set $\{x = [x_1, x_2, \dots, x_N] : |x_i| \le \max|f_i(x)|, x \in \mathbb{R}\}$. For the system

$$x_{k+1} = W \tanh(x_k) \tag{17}$$

where W is an $N \times N$ matrix, D_0 is chosen as the polytope $\prod = \{x : |x_i| \leq \sum_{i=1}^{N} |W_{ji}|\}$ as $|\tanh(x)| \leq 1$. The algorithm provided in [25] shows the detailed steps of reduction of the dissipativity domain, and proves that the system has the equilibrium point at the origin. To use the algorithm, we transform (16) to the equivalent form of $x_{k+1} = f(x_k)$ with the equilibrium point at the origin. Assume x^* is the equilibrium point of (16), i.e.,

$$x^* = \tanh(W_1 x^* + B_2). \tag{18}$$

Take $c = W_1 x^* + B_2$, and use change of variable $v = W_1(x - x^*)$, (16) can be rewritten as

$$v_{k+1} = W_1(x_{k+1} - x^*) = W_1(\tanh(v_k + c) - \tanh(c)). \tag{19}$$

It is clear that the equilibrium of (19) is at the origin. After the domain is contracted to a small region, we use a simplified version (see Theorem 1) of the theorem in [26] to complete the stability proof.

Theorem 1. For the following system:

$$\begin{aligned} x_{k+1} &= B_p \phi(\xi_k) \\ \xi_k &= C_q x_k \end{aligned} \tag{20}$$

which has one equilibrium point at the origin, it is globally asymptotically stable if there exist symmetric positive-definite matrices P and diagonal semipositive-definite matrices Λ and M such that the following LMI holds:

$$\begin{bmatrix} -P - 2C_q^T M Q U C_q & C_q^T \Lambda + C_q^T (Q+U) M \\ \Lambda C_q + M(Q+U) C_q & B_p P B_p - 2M \end{bmatrix} < 0 \tag{21}$$

where $Q = \text{diag}\{q_1, q_2, \ldots\}$, $U = \text{diag}\{\mu_1, \mu_2, \ldots\}$, and $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \ldots\}$. Q and U can be determined from $\phi(\cdot)$, i.e., the sector conditions (22) with $q_i, \mu_i > 0$

$$\phi_i(\xi_i(\cdot))/\xi_i(\cdot) \in [q_i, \mu_i]. \tag{22}$$

Proof: Choose the following Lyapunov functional:

$$V(x_k) = x_k^T P x_k + 2 \sum_{i=1}^{L} \lambda_i \sum_{j=0}^{k-1} \phi_i(\xi_i(j)) \xi_i(j) \tag{23}$$

where $\lambda_i \geq 0$. Since A, D_p in [26, eq. (52)] are zeros, the LMI [i.e., (21)] can be obtained based on the sector conditions in (22). The readers are referred to [26] for the detailed proof.■

For system (19), which is equivalent to (16), let $B_p = W_1$, $Cq = I$ (identity matrix), $\phi(s) = \tanh(s + c) - \tanh(c)$. Then, Theorem 1 can be applied as q_i and μ_i will be very close to 0 because of the bounds added on the variables by the contraction algorithm [25]. In sum, the algorithm in [25] contracts the domain to a small area around the equilibrium point, i.e., $q_i, \mu_i \to 0$, then by the use of Theorem 1, the globally asymptotically stability of (16) is proved. This completes the stability proof of the proposed RNNinv in unforced mode.

Note that the stability in unforced mode can be directly verified by simulations. In practice, stability of an RNNinv in unforced mode is useful as once proven stable, the RNNinv always evolves to a certain state instead of an unpredictable random one. In addition, it may provide insights about the input–output stability of RNNinv for future investigations.

V. Separating the Controlling Dynamics

In practice, the LME may affect the control performance at high frequencies, as the LME high-frequency dynamics cannot be exactly gain one with zero-phase delay. Therefore, we present a mechanism to ensure that the PEA high-frequency dynamics and control is solely taken care of by the RNNinv by using LPFs, as shown in Fig. 2 (this augmented approach will be called "RNNinv+LME," compared to $\overline{\text{RNNinv+LME}}$, the bar is removed to indicate that the effect the LME on high-frequency dynamics has been removed).

First, suppose the dynamics of RNNinv+PEA can be represented by the following state space equation:

$$\begin{aligned} x_{k+1} &= h(x_k, u_k) \\ y_k &= W x_k + B. \end{aligned} \tag{24}$$

The mechanism of isolating the LME for high-frequency control rests on the assumption that h in (24) satisfies

$$h(x_{Lk} + x_{Hk}, u_{Lk} + u_{Hk}) = h_1(x_{Lk}, u_{Lk}) + h_2(x_{Hk}, u_{Hk}) \tag{25}$$

where $x_k = x_{Lk} + x_{Hk}$, $u_k = u_{Lk} + u_{Hk}$, h_1, and h_2 can be any reasonable functions. Equation (25) is also called additivity. Whether this condition will hold or not depends on how accurate the RNNinv is. If RNNinv is not effective in compensating for the PEA nonlinearity, this assumption will be violated, and tracking error will occur.

Next, we show how the separation mechanism works. Previously, we showed that the system LPF1+RNNinv+PEA (i.e., G_1 in Fig. 2) can be described by (8). Introduce another LPF—LPF2 represented by the following model:

$$\begin{aligned} \alpha_{k+1} &= \underline{A}\alpha_k + \underline{B}\mathrm{u}_k \\ \underline{z}_k &= \underline{C}\alpha_k. \end{aligned} \tag{26}$$

It follows that G_1+LPF2 (called G_2 in Fig. 2) is

$$\begin{aligned} \begin{bmatrix} \phi_{k+1} \\ \alpha_{k+1} \end{bmatrix} &= \begin{bmatrix} A & 0 \\ \underline{B}C & \underline{A} \end{bmatrix} \begin{bmatrix} \phi_k \\ \alpha_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \mathrm{u}_k + \begin{bmatrix} G \\ 0 \end{bmatrix} \hat{e}_k \\ \underline{z}_k &= \begin{bmatrix} 0 & \underline{C} \end{bmatrix} \begin{bmatrix} \phi_k \\ \alpha_k \end{bmatrix}. \end{aligned} \tag{27}$$

As shown in Fig. 2, the desired trajectory known *a priori* is divided into low-frequency part r_L and high-frequency part r_H offline using a zero-phase LPF (LPF0 in Fig. 2). Since it is assumed that (25) holds, r_L and r_H will incur output y_L and y_H, respectively. Note that y_H will be blocked by LPF2, it is equivalent to the case where r_L is the only reference applied to the predictive controller. Therefore, the Kalman estimator in Fig. 2 is designed based on G_2 with the estimated state $[\phi_k^T \alpha_k^T]^T$. By extracting ϕ_k from $[\phi_k^T \alpha_k^T]^T$ and feeding it to the predictive controller, which is based on the LME+LPF1, the

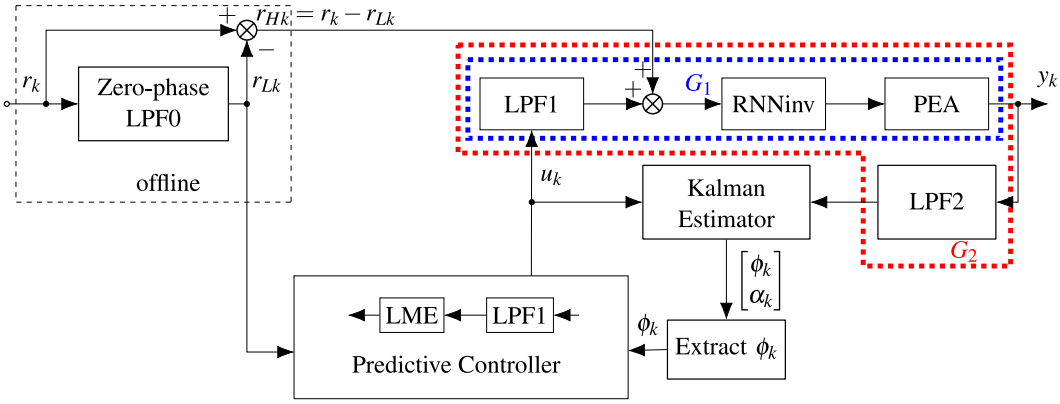


Fig. 2. Block diagram of integrating RNNinv and LME with the separation mechanism (called "RNNinv+LME").

predictive controller will only act on low-frequency dynamics. The necessity of adding such a dynamics separation mechanism in the proposed control framework was verified in experiments and shown in the subsequent section.

Suppose the cutoff frequencies of LPF0 and LPF2 are f_{c0} and f_{c2}, respectively. To ensure that the output y_H corresponding to r_H will be completely removed after passing through LPF2, f_{c2} should be chosen slightly higher than f_{c0}.

Proposition 1. Every frequency component in the reference trajectory is accounted for either by RNNinv, or, predictive controller, or both.

Proof: As shown in Fig. 2, the reference trajectory is r_k, after passing through LPF0, the low-frequency reference is r_{Lk} and the high-frequency part is $r_{Hk} = r_k - r_{Lk}$. Suppose the cutoff frequency of LPF0 is f_{c0}, due to the feature of LPF, some frequencies close to f_{c0} can appear both in r_{Hk} and r_{Lk}. Suppose the bandwidth of r_{Lk} and r_{Hk} are $[0, f_{c0} + \epsilon_1]$ and $[f_{c0} - \epsilon_2, f_s/2]$, respectively, where f_s is the sampling frequency, $\epsilon_1 > 0$ and $\epsilon_2 > 0$. To separate the dynamics, it only needs to ensure that any frequency higher than $f_{c0} - \epsilon_2$ cannot pass through LPF2 as explained as follows.

Suppose that for LPF2 (with cutoff frequency of f_{c2}), frequency higher than $f_{c2} + \epsilon_3$ ($\epsilon_3 > 0$) cannot pass through LPF2. Set $f_{c2} + \epsilon_3 = f_{c0} - \epsilon_2$, thus $f_{c2} = f_{c0} - \epsilon_2 - \epsilon_3$. ϵ_2 and ϵ_3 can be obtained with MATLAB. Therefore, the output incurred by r_{Hk} cannot pass through LPF2 by choosing $f_{c2} = f_{c0} - \epsilon_2 - \epsilon_3$, thus, it is equivalent that r_{Hk} cannot affect the system consisting of LPF1+RNNinv+PEA+LPF2 assuming that (25) holds.

Recall that r_{Hk} is fed into RNNinv directly and r_{Lk} is the reference trajectory to be tracked by the predictive controller as seen in Fig. 2. Therefore, r_{Hk} is accounted for by RNNinv only, and r_{Lk} is accounted for by the predictive controller. Considering $r_{Hk} + r_{Lk} = r_k$, we can now claim that every frequency is accounted for either by RNNinv, or the predictive controller, or both. ■

Remark 3. Note that $\hat{e}_k$ in (8) is the same with that in (27). Therefore, for the Kalman estimator, the computation of $\hat{e}_k$ should be $\hat{e}_k = \bar{y}_k - r_{H,k} - C\hat{\phi}_k^- = \bar{y}_{L,k} + \bar{y}_{H,k} - r_{H,k} - C\hat{\phi}_k^-$. If the RNNinv for high-frequency dynamics is accurate enough, we will have $\bar{y}_{H,k} - r_{H,k} \approx 0$.

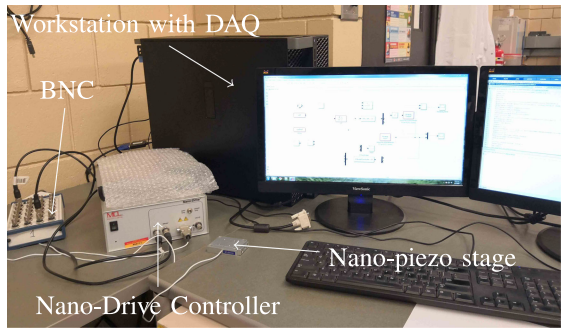


Fig. 3. Experimental setup.

VI. Experiment Results and Discussion

The proposed RNNinv+LME framework was implemented on a PEA (Nano-OP30, Mad City Labs) with the maximum displacement of 30 μm (see the supplementary material for detailed manufacturer specifications), and the results were compared with that of a PI feedback controller. Also, to further evaluate the accuracy of the proposed method, the tracking results were compared with one ILC approach—MIIFC [23]—to demonstrate that the proposed real-time control approach was able to reach the accuracy as high as that of the offline control technique.

All the signals were acquired by the data acquisition system (NI PCIe-6353, National Instruments), which was installed in the workstation (Intel Xeon W-2125, RAM 32 GB). The controller was designed in MATLAB Simulink (MathWorks, Inc.) environment. The experiment setup is shown in Fig. 3. The sampling frequency was 10 kHz. Frequency response of the PEA is shown in Fig. 4.

A. Constructing the RNNinv Training Set

To generate $Y_{(\text{ts})}$, 70 (frequency, amplitude) pairs were computed using k-means algorithms [27]. Before implementing the algorithm, 5000 points were randomly generated to cover the space in the given range (frequency: 0–250 Hz, amplitude: 0–4.5 V). Then, the generated $Y_{(\text{ts})}$ shown in Fig. 5 was set as the desired trajectory to be tracked by the PEA through MIIFC,

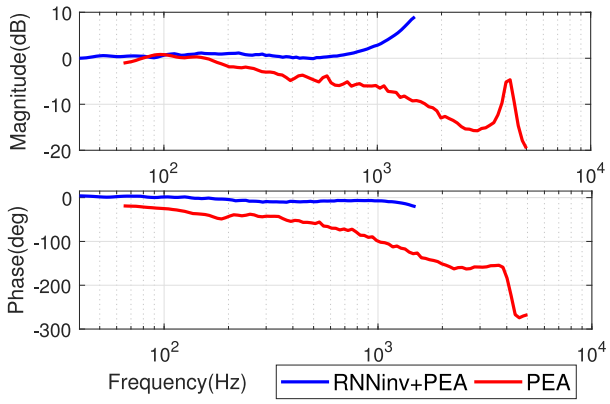


Fig. 4. Bode plot of the PEA and RNNinv+PEA.

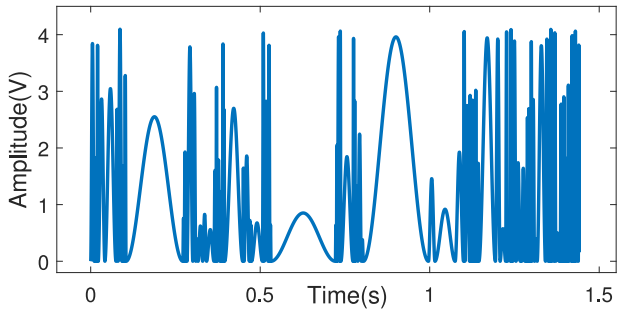


Fig. 5. Generated time series $Y_{(\mathrm{ts})}$ for RNNinv training.

and the corresponding $U_{(\mathrm{ts})}$ (i.e., the converged drive input to the PEA calculated by MIIFC) was obtained. A 9th order RNNinv was trained with the obtained $(Y_{(\mathrm{ts})}, U_{(\mathrm{ts})})$.

B. Evaluating the Modeling Accuracy of RNNinv

The effectiveness of the RNNinv in eliminating system nonlinearities such as hysteresis was validated first. The PEA hysteresis curves (see Fig. 6) were measured using different sinusoidal drive voltages with the frequencies of 30, 120, and 200 Hz, respectively, for the original PEA system and system $\mathbb{H}$ (RNNinv cascaded with PEA). Note that these three frequencies were randomly selected and did not overlap with the training frequency components. The displacement range generated was about 45% of maximum PEA displacement range. It can be seen that the PEA hysteresis is both rate- and amplitude-dependent. By cascading the proposed RNNinv, hysteresis at all measured frequencies and amplitudes were removed. Therefore, the PEA hysteresis nonlinearity was effectively accounted for by the RNNinv.

In Section II, we stated that FNN is not suitable for the inversion model. We have conducted experiments to show that RNN is more accurate than FNN for the PEA system in this work (see the supplementary material).

C. RNNinv Stability in Unforced Mode

If the given system (16) is globally asymptotically stable, it should have only one equilibrium point and converges to the same point for any initial state. Given any RNNinv, it is not guaranteed to be stable. As running the algorithm in Section IV is more complicated than directly simulating the system, we first simulate RNNinv numerically with randomly selected initial states, if divergence occurs, there is no need to run the algorithm in Section IV. Once convergence is verified numerically in simulation, the aforementioned algorithm can be used to rigorously prove that RNNinv is globally asymptotically stable.

Three hundred random initial states were chosen for the RNNinv, and for each of them, 100 steps were simulated. The simulation results (see Fig. 7) show that the RNNinv converges to the same point for any initial state in less than 0.01 s (100 steps), which implies that the RNNinv is probably globally asymptotically stable. Note that the convergence of the obtained RNNinv, based on its function, is directly determined by the PEA inversion dynamics. Therefore, the convergence rate in simulation cannot be used to evaluate the performance of the proposed approach.

Also, one equilibrium point $x^* = [0.688, -0.378, 0.104, 0.529, 0.969, 0.229, -0.913, 0.884, 0.904]^T$ was obtained. Next the global asymptotic stability was verified using the theories presented in Section II-B. By using the algorithm in [25], the dissipativity domain is contracted to the area denoted with $\bar{q}$ and $\bar{\mu}$, where

$$\begin{aligned}\bar{q} &= [0.1332, 0.1095, 0.0857, 0.2120, 0.2536 \\ &\quad 0.1119, 0.3493, 0.1416, 0.9065]^T \\ \bar{\mu} &= [-0.0952, -0.0979, -0.1255, -0.1198, -0.2044 \\ &\quad -0.3694, -0.1010, -0.2514, -0.1128]^T. \end{aligned} \tag{28}$$

In the simulation, it took 261 steps with 569 linear constraints to reach this result. With the bounds in (28), the LMI in (21) was solved using MATLAB LMI toolbox. Since the solution of LMI did exist, it can be concluded that the obtained RNNinv is indeed globally asymptotically stable in unforced mode.

Note that as the order of RNNinv increases, the computation power needed to contract the dissipativity domain will increase exponentially, which limits this method for the proof of globally asymptotically stability for high-order RNN system.

D. Tracking Performance Comparison

The tracking performance is demonstrated as follows. The prediction horizon was 20 and the closed-loop stability condition (15) was satisfied. The modeling bandwidth of RNNinv was 250 Hz, but the control bandwidth could reach 350 Hz due to the generalization ability of RNN. Therefore, the bandwidth of the trajectories to be tracked should be within 350 Hz. The desired PEA trajectories used were sinusoidal signals (with the frequencies of 30, 100, and 200 Hz, and amplitude of 6 μm), triangle signal (50 Hz), chirp signal (frequency range 0–350 Hz) and Γ which is written as $\Gamma(t) = [0.8\sin(2\pi 5t + 1.5\pi) + 0.43\sin(2\pi 50t) + 0.12\sin(2\pi 120t + 1.2\pi) + 0.3\sin(2\pi 180t + \pi)]/1.3$.

First, the proposed method (RNNinv+LME) is compared with the case in which only RNNinv is used to show the necessity

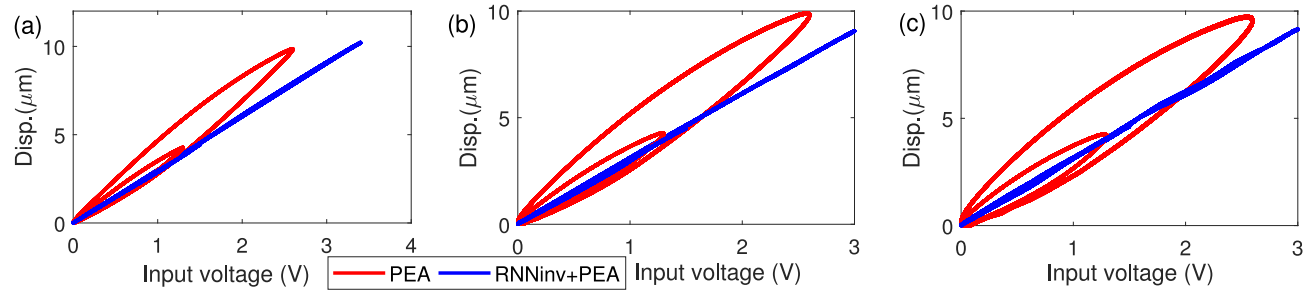


Fig. 6. Measured displacement versus input voltage curves at the frequencies of (a) 30 Hz, (b) 120 Hz, (c) 200 Hz from the PEA and the RNNinv+PEA system, respectively.

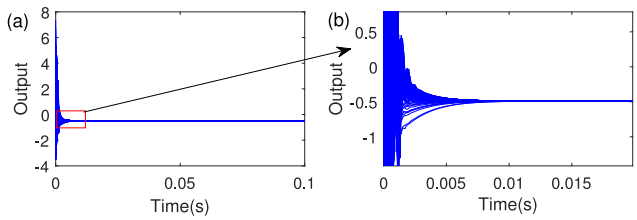


Fig. 7. (a) Simulation of the obtained RNNinv for 100 steps with three hundreds randomly selected initial states. (b) Zoomed-in view of (a).

of the LME and the predictive controller. Then, it is compared with a PI feedback controller. The parameters of PI controller were $P = 0.01$, and $I = 300$. Since MIIFC has been proven effective in achieving high-precision PEA control, it is chosen as the benchmark to evaluate the accuracy of the proposed framework. The control law of MIIFC [23] is shown in the following equation. The gain α was set to $\alpha = 1.2$

$$u_0(j\omega) = \alpha y_d(j\omega)$$
$$u_i(j\omega) = \begin{cases} \frac{u_{i-1}(j\omega)}{y_{i-1}(j\omega)} y_d(j\omega) & \text{if } y_{i-1}(j\omega) \neq 0 \text{ and} \\ & y_d(j\omega) \neq 0,\ i \geq 1 \\ 0 \quad \text{otherwise.} \end{cases} \tag{29}$$

Furthermore, we also demonstrate that it is necessary to implement the control separation mechanism introduced in Section IV by comparing the performances of RNNinv+LME and that without the control separation mechanism (called "RNNinv+LME"). f_{c0}, f_{c1}, and f_{c2} were chosen as 45, 1200, and 25 Hz, respectively. Note that f_{c1} should be as large as possible when tracking broad-band trajectories, however, since not all the dynamics are modeled, its value should be limited. The frequency response of RNNinv+PEA shown in Fig. 4 is helpful for choosing f_{c1} as the modeling bandwidth is close to 1000 Hz, choosing 1200 Hz can make the controller more aggressive. However, in the experiment, it is observed that when f_{c1} changed between 800 and 1300 Hz, the performance for tracking band-limited trajectories did not show much difference. How to choose f_{c0} and f_{c2} has been elaborated in the proof for *Proposition* 1.

For the predictive controller, N_p and N_c were chosen to be 40 and 20, respectively. The tracking errors were computed as follows [23]:

$$E_{\max} \triangleq \frac{||r(\cdot) - y(\cdot)||_\infty}{||r(\cdot)||_\infty},\ E_{\text{rms}} \triangleq \frac{||r(\cdot) - y(\cdot)||_2}{||r(\cdot)||_2} \tag{30}$$

where $r(\cdot)$ and $y(\cdot)$ are complex vectors obtained through discrete Fourier transform of the corresponding signals, respectively. Table I shows the tracking errors for all the approaches. Fig. 8 shows the tracking performance in time domain for the desired trajectories of 100 Hz sinusoidal signal, stair-like trajectory, and Γ, respectively. More tracking results can be found in the supplementary material.

RNNinv+LME, versus RNNinv When tracking trajectories contained low-frequency dynamics, RNNinv+LME outperformed RNNinv. For example, in Table I, the tracking errors of RNNinv+LME for 30 Hz signal and Γ signal are about 40% less than those of RNNinv. This is because LME captured the PEA low-frequency dynamics that RNNinv missed, and the error term in LME could also eliminate the tracking error contributed by the low-frequency dynamics, which is clearly shown in Fig. 8(b), (c), (e), and (f) part of the low-frequency error was removed by RNNinv+LME. However, since LME was not designed to model the PEA high-frequency dynamics, it did not improve the tracking performance when tracking 100 Hz and 200 Hz signals as can be seen from Table I and Fig. 8. Furthermore, LME could negatively influence the tracking performance at high frequencies. In Table I, the tracking error of RNNinv for 200 Hz signal is very low, which indicates that the RNNinv could model the high-frequency inversion dynamics accurately. Thus, the large tracking errors of RNNinv+LME in this case might be induced by the LME: tracking errors of RNNinv+LME are about three times larger than that of RNNinv for 200 Hz signal. However, by incorporating the separating mechanism based on RNNinv+LME, the overall control performance is further improved in this case as seen in Table I.

RNNinv+LME versus PI: PI controller, as one of the most popular real-time controllers, works for low-frequency tasks in general. But its performance downgrades significantly as the frequency increases. As seen in Table I, RNNinv+LME decreased the tracking errors by at least 50% for all the cases compared to PI. The performance difference is more obvious as the frequency increases. Moreover, the tracking errors in time domain in Fig. 8 can clearly demonstrate the superiority of the proposed method over PI.

TABLE I
PEA TRACKING PERFORMANCE COMPARISON

Refs.	30Hz		100Hz		200Hz		Γ		Chirp 0-350Hz		Triangle 50Hz		stair	
Error(%)	E_{rms}	E_{max}	E_{rms}	E_{max}	E_{rms}	E_{max}	E_{rms}	E_{max}	E_{rms}	E_{max}	E_{rms}	E_{max}	E_{rms}	E_{max}
RNNinv+LME	0.65	0.33	1.35	0.94	2.31	1.61	2.26	1.11	2.98	0.26	3.17	2.27	1.89	0.70
$\overline{\text{RNNinv+LME}}$	0.93	0.43	1.71	1.29	4.12	3.27	2.14	1.06	6.89	0.57	2.86	2.41	2.82	1.43
RNNinv	2.41	1.68	1.64	1.19	1.37	0.54	3.92	1.28	1.74	0.22	2.83	2.07	9.15	6.33
PI	2.69	2.29	9.19	7.93	19.29	16.61	4.81	3.30	51.87	4.71	6.43	3.88	4.07	0.35
MIIFC	2.01	1.21	1.81	1.18	1.69	1.11	2.23	1.16	4.03	0.47	0.94	0.57	1.50	1.19
RNNPC	1.27	0.39	2.32	1.74	6.31	7.48	1.82	0.69	6.64	0.51	2.19	1.18	2.38	1.00

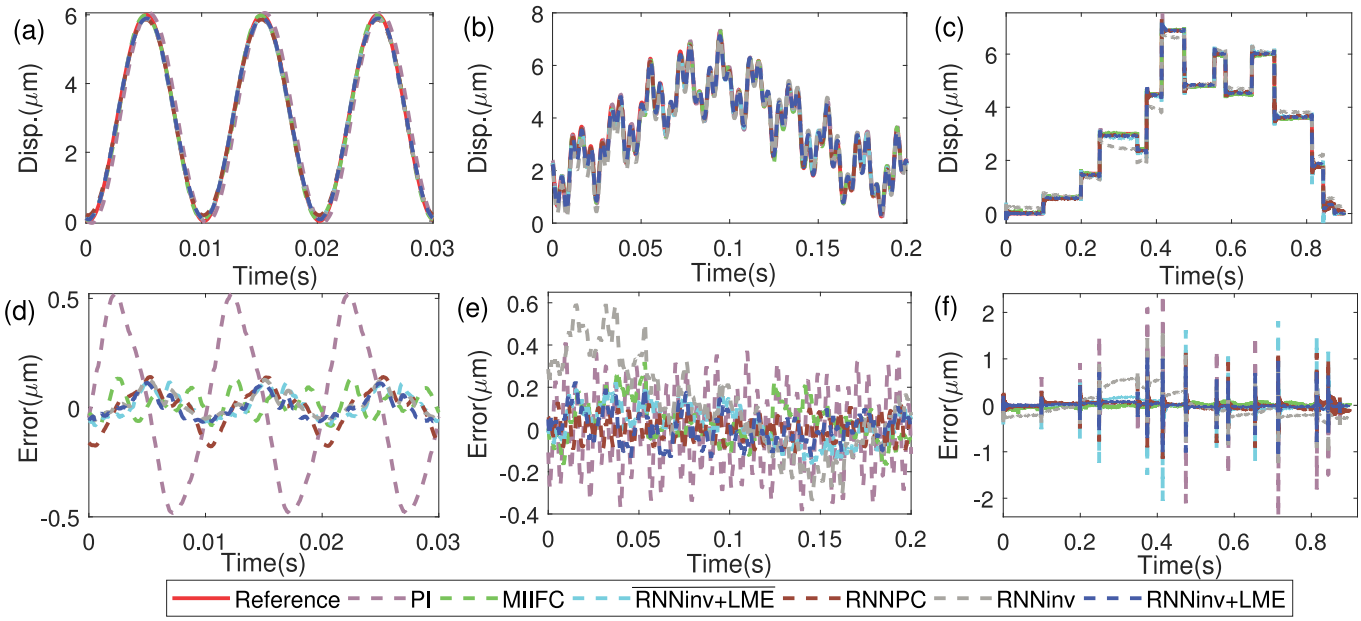


Fig. 8. Comparison of the tracking results of MIIFC, $\overline{\text{RNNinv+LME}}$, RNNPC, RNNinv, PI, and RNNinv+LME for (a) 100 Hz sinusoidal trajectory, (b) Γ, and (c) stair-like trajectory. (d)–(f) Corresponding tracking errors, respectively.

$\overline{\text{RNNinv+LME}}$ versus MIIFC: For MIIFC, the converged tracking results were chosen after eight iterations. From Table I, it can be seen that $\overline{\text{RNNinv+LME}}$ outperformed MIIFC when tracking low-frequency trajectories—the tracking error is about half of that of MIIFC for 30 Hz signal, and has comparable accuracy with MIIFC at "medium" frequency range, i.e., tracking 100 Hz signal and Γ signal, which is shown in details in Fig. 8. Although there is a surge in the tracking error for high-frequency trajectory tracking (i.e., 200 Hz), the separating mechanism can mitigate the problem in this case. Overall, even compared to the MIIFC—an offline control approach, the proposed method can still achieve similar or even better control accuracy in real time.

RNNinv+LME: Since RNNinv+LME is based on $\overline{\text{RNNinv+LME}}$ and is designed to ensure that LME and the predictive controller only work on the system control at low frequencies. Therefore, it is expected to achieve higher accuracy at low-frequency region compared to RNNinv and at high-frequency region compared to $\overline{\text{RNNinv+LME}}$. In Table I, the performance of RNNinv+LME is close to that of $\overline{\text{RNNinv+LME}}$ when tracking low-frequency trajectories (e.g., 30 Hz signal and Γ). There are two reasons that may induce tracking errors for RNNinv+LME. One is the violation of the assumption (25) and the other is the modeling error of RNNinv (i.e., $\bar{y}_{H,k} - r_{H,k} \neq 0$ as explained in *Remark 3*). Therefore, when tracking high-frequency signal (e.g., 200 Hz signal), RNNinv+LME cannot reach the accuracy of RNNinv. Nevertheless, RNNinv+LME absorbs the merits of both LME and RNNinv achieving the best overall performance.

Moreover, compared to RNNPC, the overall performance of RNNinv+LME is better especially for high-frequency trajectories tracking as seen in Table I. More importantly, RNNinv+LME is much more computationally efficient than RNNPC with controller time complexity of $\Theta(N_h N)$ compared to $\Theta(k2(N_h - \frac{N_c}{2})N_c N_m^2)$ of RNNPC [21].

Note that although the RNNinv can model the nonlinear dynamics of the PEA system, it does not assume any specific forms of nonlinearity of the system in advance, thus, it is expected to model any nonlinearities with enough parameters in theory.

TABLE II
COMPARISON OF DIFFERENT CONTROLLERS

Controllers	Time Complexity	Θ	Tractable?	Real-time?	Bandwidth	Accuracy
PI	$\Theta(N_m)$	$\Theta(5)$	Yes	Yes	Medium	Low
MIIFC	$\Theta(2N_d \log(N_d))$	$\Theta(5.7e5)$	Yes	No	Very High	Very High
$\overline{\text{RNNinv+LME}}$	$\Theta(N_m^2+N_h)$	$\Theta(45)$	Yes	Yes	High	High
RNNPC	$\Theta(k(N_h-\frac{N_c}{2})N_cN_m^2)$	$\Theta(9.6e4)$	No	Yes	High	High
RNNinv	$\Theta(N_m^2)$	$\Theta(25)$	Yes	Yes	High	Medium
RNNinv+LME	$\Theta(N_m^2+N_h)$	$\Theta(45)$	Yes	Yes	High	Very High

On the other hand, the LME overcomes the issue caused by the limited length of the RNN training set. Therefore, the RNNinv+LME framework is expected to have broader applications for tracking control of other systems.

All the controllers are compared in Table II in terms of computation features. In evaluating the time complexity Θ qualitatively (as shown in third column in Table II), the parameters are set as follows: modeling order $N_m = 5$, prediction horizon $N_h = 20$, number of sampled points in the trajectory to be tracked $N_d = 20000$, for RNNPC, $k = 10$, $N_m = 20$, $N_c = 6$, $N_h = 12$ (refer to [21] for the details). "Tractability" is determined by whether the optimization problem in the controller can be solved in limited time with limited computation resources. For instance, RNNPC involves solving a general optimization (usually nonconvex) problem, thus, is intractable. Note that the control bandwidth is defined as the first frequency where the gain drops below 70% of its dc value. As an example, when tracking 320 Hz sinusoidal trajectory, the gains of PI and RNNPC are 65% and 72%, respectively. When tracking 350 Hz sinusoidal trajectory, the gains of MIIFC, $\overline{\text{RNNinv+LME}}$, RNNinv, and RNNinv+LME are 100%, 87%, 96%, and 96%, respectively. Therefore, the proposed approach can achieve excellent overall performance among all the real-time controllers compared.

VII. CONCLUSION

In this article, we proposed to use RNNinv+LME to realize a broadband, high-precision, and computationally efficient real-time controller for PEA trajectory tracking. The key novelty of the proposed control framework lies in the following conditions: 1) the idea of using RNN to model system inversion dynamics (RNNinv) for nonlinearity compensation, and 2) the integration of RNNinv and an LME-based model predictive controller to ensure the control accuracy and computation efficiency. Besides, the proposed separating mechanism further improves the controller performance.

In the future, stability analysis of RNNinv and its combination with Kalman filter in forced mode will be thoroughly studied. Also, the RNN structure used in this article is one of many RNNs, we will explore RNNs with other structures to further increase the modeling bandwidth and accuracy.

REFERENCES

[1] Y. Tian, D. Zhang, and B. Shirinzadeh, "Dynamic modelling of a flexure-based mechanism for ultra-precision grinding operation," *Precis. Eng.*, vol. 35, no. 4, pp. 554–565, 2011.

[2] K. Mollaeian, Y. Liu, S. Bi, Y. Wang, J. Ren, and M. Lu, "Nonlinear cellular mechanical behavior adaptation to substrate mechanics identified by atomic force microscope," *Int. J. Mol. Sci.*, vol. 19, no. 11, 2018, Art. no. 3461.

[3] E. D. Gedikli, D. Chelidze, and J. Dahl, "Active control of flexible cylinders undergoing vortex-induced vibrations using piezo stripe actuators," in *Structural Health Monitoring, Photogrammetry & DIC, Volume 6*. Berlin, Germany: Springer, 2019, pp. 63–65.

[4] G.-Y. Gu, L.-M. Zhu, C.-Y. Su, H. Ding, and S. Fatikow, "Modeling and control of Piezo-actuated nanopositioning stages: A survey," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 1, pp. 313–332, Jan. 2016.

[5] S. Tien, Q. Zou, and S. Devasia, "Iterative control of dynamics-coupling-caused errors in piezoscanners during high-speed AFM operation," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 6, pp. 921–931, Nov. 2005.

[6] C.-Y. Lin and Y.-C. Liu, "Precision tracking control and constraint handling of mechatronic servo systems using model predictive control," *IEEE/ASME Trans. Mechatronics*, vol. 17, no. 4, pp. 593–605, Aug. 2012.

[7] Y. Shan and K. K. Leang, "Dual-stage repetitive control with Prandtl–Ishlinskii hysteresis inversion for piezo-based nanopositioning," *Mechatronics*, vol. 22, no. 3, pp. 271–281, 2012.

[8] Y. Shan and K. K. Leang, "Accounting for hysteresis in repetitive control design: Nanopositioning example," *Automatica*, vol. 48, no. 8, pp. 1751–1758, 2012.

[9] M. Edardar, X. Tan, and H. K. Khalil, "Sliding-mode tracking control of piezo-actuated nanopositioners," in *Proc. Amer. Control Conf.*, 2012, pp. 3825–3830.

[10] M. Al Janaideh, S. Rakheja, and C.-Y. Su, "An analytical generalized Prandtl–Ishlinskii model inversion for hysteresis compensation in micropositioning control," *IEEE/ASME Trans. Mechatronics.*, vol. 16, no. 4, pp. 734–744, Aug. 2011.

[11] Z. Li, J. Shan, and U. Gabbert, "Dynamics modeling and synchronized model predictive control for a Fabry-Perot spectrometer," *IEEE/ASME Trans. Mechatron.*, vol. 24, no. 4, pp. 1818–1828, Aug. 2019.

[12] Y. Cao, L. Cheng, X. Chen, and J. Peng, "An inversion-based model predictive control with an integral-of-error state variable for piezoelectric actuators," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 3, pp. 895–904, Jun. 2013.

[13] S. Cao, B. Wang, J. Zheng, W. Huang, L. Weng, and W. Yan, "Hysteresis compensation for giant magnetostrictive actuators using dynamic recurrent neural network," *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 1143–1146, Apr. 2006.

[14] M. Al Janaideh, M. Al Saaideh, and M. Rakotondrabe, "Temperature dependent hysteresis modeling of a piezotube actuator using Elman neural network," in *Proc. Dyn. Syst. Control Conf.*, vol. 59148, 2019, Art. no. V001T09A004.

[15] X. Zhao, S. Shen, L. Su, and X. Yin, "Elman neural network–based identification of rate-dependent hysteresis in piezoelectric actuators," *J. Intell. Mater. Syst. Struct.*, vol. 31, no. 7, pp. 980–989, 2020.

[16] J. Y. Lau, W. Liang, and K. K. Tan, "Motion control for piezoelectric-actuator-based surgical device using neural network and extended state observer," *IEEE Trans. Ind. Electron.*, vol. 67, no. 1, pp. 402–412, Jan. 2020.

[17] L. Cheng, W. Liu, Z.-G. Hou, J. Yu, and M. Tan, "Neural-network-based nonlinear model predictive control for piezoelectric actuators," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7717–7727, Dec. 2015.

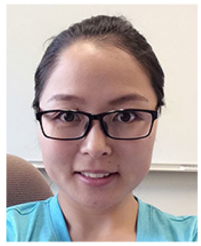
[18] W. Liu, L. Cheng, Z.-G. Hou, J. Yu, and M. Tan, "An inversion-free predictive controller for piezoelectric actuators based on a dynamic linearized neural network model," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 1, pp. 214–226, Feb. 2016.

[19] A. M. Schäfer and H.-G. Zimmermann, "Recurrent neural networks are universal approximators," *Int. J. Neural Syst.*, vol. 17, no. 4, pp. 253–263, 2007.

[20] H. Zimmermann, R. Grothmann, A. Schaefer, and C. Tietz, "Identification and forecasting of large dynamical systems by dynamical consistent neural networks," *New Directions in Statistical Signal Processing: From Systems to Brain*. Cambridge, MA, USA: MIT Press, 2006, pp. 203–242.
[21] S. Xie and J. Ren, "Recurrent-neural-network-based predictive control of piezo actuators for precision trajectory tracking," in *Proc. IEEE Annu. Amer. Control Conf.*, 2019, pp. 3795–3800.
[22] S. Xie and J. Ren, "Note: Precision control of nano-positioning stage: An iterative learning-based model predictive control approach," *Rev. Sci. Instrum.*, vol. 89, no. 7, 2018, Art. no. 076103.
[23] K.-S. Kim and Q. Zou, "A modeling-free inversion-based iterative feedforward control for precision output tracking of linear time-invariant systems," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 6, pp. 1767–1777, Dec. 2013.
[24] C. Wang, A. Ohsumi, and I. Djurovic, "Model predictive control of noisy plants using Kalman predictor and filter," in *Proc. IEEE Region 10 Conf. Comput., Commun., Control Power Eng.*, 2002, vol. 3, pp. 1404–1407.
[25] N. E. Barabanov and D. V. Prokhorov, "A new method for stability analysis of nonlinear discrete-time systems," *IEEE Trans. Autom. Control*, vol. 48, no. 12, pp. 2250–2255, Dec. 2003.
[26] M. Liu, "Delayed standard neural network models for control systems," *IEEE Trans. Neural Netw.*, vol. 18, no. 5, pp. 1376–1391, Sep. 2007.
[27] S. Xie and J. Ren, "High-speed AFM imaging via iterative learning-based model predictive control," *Mechatronics*, vol. 57, pp. 86–94, 2019.

Shengwen Xie received the B.S. degree in mechanical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2013, and the M.E. degree in electrical engineering from Tianjin University, Tianjin, China, in 2016. He is currently working toward the Ph.D. degree in mechanical engineering with the Iowa State University, Ames, IA, USA.

His research interests include dynamics, control, and robotics.



Juan Ren (Member, IEEE) received the B.S. degree from Xi'an Jiaotong University, Xi'an, China, in 2009, and the Ph.D. degree from Rutgers, the State University of New Jersey, New Brunswick, NJ, USA, in June 2015, both in mechanical engineering.

She is currently an Assistant Professor with the Department of Mechanical Engineering, Iowa State University, Ames, IA, USA, where she has been on the faculty since August 2015. Her research interests include learning-based output tracking and control, control tools for high-speed scanning probe microscope imaging, mechanotransduction modeling, and nanomechanical measurement and mapping of soft and live biological materials.

Dr. Ren was the recipient of the NSF CAREER Award in 2018, and currently holds the William and Virginia Binger Professorship in the Department of Mechanical Engineering of ISU. She is the representative of the IEEE Control Systems Society in the IEEE Nanotechnology Council, and an Associate Editor for Elsevier *Mechatronics*.