2019 American Control Conference (ACC)
Philadelphia, PA, USA, July 10-12, 2019

Recurrent-neural-network-based Predictive Control of Piezo Actuators for Precision Trajectory Tracking

Shengwen Xie[1] and Juan Ren[1,†]

Abstract— **Precise real-time trajectory tracking of piezo actuators (PEAs) is essential to high-precision systems and applications. However, most current real-time control techniques for PEAs are based on linear models and suffer significantly from modeling uncertainty. In this paper, we propose a network (RNN)-based predictive control technique for real-time PEA trajectory tracking. Specifically, a RNN is trained to model the nonlinear dynamics of the PEA system. Considering the length of the RNN training set is limited, a second order linear model embedded with an error term (LME) is proposed to model the PEA low frequency dynamics. Moreover, an unscented Kalman filter is designed to estimate the states of the nonlinear model. Then the nonlinear model consisting of the RNN and the LME are used for nonlinear predictive control based on gradient descent algorithm. To solve the optimization problem in the nonlinear predictive control, a method for analytically calculating the gradient of the cost function is developed as well. To verify the effectiveness of the proposed approach, experiments were conducted on a nano piezo actuator. The results demonstrated that the proposed method can achieve high precision output tracking of PEAs in real time.**

I. Introduction

Owing to its fast response and mechanical stability, piezo actuators (PEAs) have been broadly used in high-precision systems and applications, such as atomic force microscope [1], [2], microforming [3] and adaptive optics [4]. However, it is the nonlinearities such as creep effect, hysteresis and mechanical vibration that make it difficult to realize precision control of PEAs in real time, especially when operated at high speed. Various control efforts have been made to solve this issue. For repetitive tasks, iterative learning control (ILC) and repetitive control are very effective in precision output (i.e., trajectory) tracking [5]–[7]. Recently, ILC algorithms aiming for tracking varying trajectories are proposed [8]–[10]. Although the ILC approaches can realize high precision output tracking, they are not capable of achieving real-time trajectory tracking of PEAs as the convergence of these approaches can only be reached through iterations during the applications [8], [9], [11]. Real-time output tracking (i.e., trajectory tracking without iterations), of PEAs still remains challenging.

Real-time control techniques have been applied for output tracking of PEAs. In [12], the dynamics of a PEA was identified with linear model and controlled with model predictive control (MPC). However, as the frequency of the trajectories to be tracked increases, the effect of nonlinearities (creep effect and hysteresis) becomes more pronounced which directly results in significant increase of the tracking error. Also, sliding mode control (SMC) based on linear model of PEAs has been developed [13]. Compared to MPC, SMC is more robust to the modeling uncertainties and disturbances. However, the system nonlinearities affect the control bandwidth of SMC significantly [13]. Moreover, the SMC induces the problem of chattering [14].

To improve the tracking performance, the nonlinearities and disturbances (i.e., creep effect, hysteresis and mechanical vibration) have been taken into account. For instance, based on the physical behavior of PEAs, a lot of models are proposed to model the hysteresis and creep effects, such as Domain wall model, Duhem model, and Prandtl-Ishlinskii model [15]–[17]. These models can be used to obtain the inversion model of PEAs which may help to remove the nonlinearities and then control algorithms such as MPC, feedback control and SMC can be applied [18]–[20]. Again, the performances of these inversion model-based approaches are directly affected by the modeling accuracy. Considering the modeling uncertainty, robust control tools such as H_∞ and adaptive control are developed to control the PEAs [21], [22]. However, the control bandwidth is rather limited due to the stability problem [21], [22].

Recently, neural network has been proposed in output tracking applications of PEAs [23]. In [23], feedforward neural network (FNN) is proposed to model the dynamics of PEAs. However, the problem with FNN is that the input to FNN is not treated as time series in the training process although the sequence of input can affect the behavior of PEAs greatly [23]. In contrast to FNN, recurrent neural network (RNN) is designed to deal with time series and was proved to be a universal approximator in modeling dynamical systems [24]. Therefore, in this paper, we propose a RNN-based approach to achieve accurate output tracking of PEAs. Specifically, a RNN is trained to accurately capture the nonlinear dynamics of the PEA system. However, due to the limited length of the RNN training set in real applications, the PEA low frequency behaviors may not be fully captured. Thus, a second order linear model embedded with an error term (LME) is proposed for modeling the residual dynamics (i.e., low frequency dynamics) [25]. Then a nonlinear predictive controller and an unscented Kalman filter are designed to work with this PEA dynamic model to achieve precise output tracking. The proposed control technique was implemented on a nano piezo positioning stage for experimental validation.

[1]S. Xie and [1]J. Ren are with the Department of Mechanical Engineering, Iowa State University, Ames, IA 50011, USA swxie@iastate.edu,juanren@iastate.edu
† Corresponding author.

Authorized licensed use limited to: Iowa State University. Downloaded on June 26,2020 at 17:24:13 UTC from IEEE Xplore. Restrictions apply.

II. System Identification

A. Recurrent Neural Network (RNN) Structure

The RNN used in this study, consisting of an input layer (solid circles), a hidden layer (circles), and an output layer (dashed circles), is shown in Fig. 1. $u_{(r),k}$ and $y_{(r),k}$ denote the input $u_{(r)}$ and output $y_{(r)}$ at the sampling instant k, respectively. $x_k = [x_{k,1}, x_{k,2}, \cdots, x_{k,N}]^T$ is the state vector of the RNN system. The activation functions of the hidden layer $g(x)$ and output layer $h(x)$ are designed as $g(x) = \tanh(x)$ and $h(x) = Wx + b$, respectively, where W is a 1-by-N matrix. Thus, the RNN in Fig. 1 can be represented by the following nonlinear state space equation as

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

where the dimensions of W_1, B_2, B_1, W_2 and B_3 are $N \times N$, $N \times 1$, $N \times 1$, $1 \times N$ and 1×1, respectively. Unlike FNN which is essentially a "nonlinear autoregressive-moving-average with exogenous inputs" model and takes the past inputs and outputs as input to the network, RNN only needs the current input $u_{(r),k}$ to generate output signal [23]. Suppose $U_{(ts)}$ is any given time series (i.e., drive voltage) and $Y_{(ts)}$ is the corresponding output time series (i.e., displacement) of a PEA system, the output of the RNN $Y_{(rts)}$ subject to input $U_{(ts)}$ should be equal to $Y_{(ts)}$ if the RNN can accurately model the PEA system dynamics, i.e., $||Y_{(ts)} - Y_{(rts)}|| < \varepsilon$ for any $\varepsilon > 0$. Therefore, the RNN can be trained (i.e., obtaining the parameters W_1, B_2, B_1, W_2 and B_3 in Eq. (1)) by solving the following optimization problem using a pre-designed time series input $U_{(ts)}$ and the measured corresponding PEA system output $Y_{(ts)}$.

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

where $U_{(ts)} = [U_{(ts),1}, U_{(ts),2}, \cdots, U_{(ts),L}]^T$, $Y_{(ts)} = [Y_{(ts),1}, Y_{(ts),2}, \cdots, Y_{(ts),L}]^T$ and $Y_{(rts)} = [Y_{(rts),1}, Y_{(rts),2}, \cdots, Y_{(rts),L}]^T$. Next, we present how to design $U_{(ts)}$.

B. Construction of Training Set for RNN

There are various ways to construct $U_{(ts)}$. Here, we choose sinusoidal signals as building blocks to form $U_{(ts)}$. Define $S(f, A) = A(\sin(2\pi f t + \frac{3\pi}{2}) + 1), t \in [0,\ 1/f]$, i.e., $S(f,\ A)$ is a sinusoidal signal with amplitude A and frequency f in a period. In practice, t will be sampled, thus $S(A,\ f)$ is a time series. Then $U_{(ts)}$ can be written as

$$U_{(ts)} = \bigcup_{(f_i,\ A_i) \in \Omega} S(f_i,\ A_i),$$

where Ω is a set consisting of $(f_i,\ A_i)$ pairs, and $\bigcup$ denotes concatenation. Therefore, each $(f_i,\ A_i)$ pair represents a point in the $f - A$ plane. Suppose $U_{(ts)}$ consists of N_1 sinusoidal signals with $f_i \in [0, \bar{f}]$ and $A_i \in [0, \bar{A}]$. Obviously, the optimal training set should consist of all the $(f_i,\ A_i)$ pairs in the given ranges, which is unpractical. Here we expect to find N_1 $(f_i,\ A_i)$ pairs to achieve the highest modeling accuracy (i.e., a suboptimal approach). Finding the suboptimal $U_{(ts)}$ is equivalent to locating N_1 points in the $f - A$ plane such that any point $(f_j,\ A_j)$ in the plane can be represented by the nearest point $(f_k,\ A_k)$ (one of the N_1 points) and the distance between $(f_j,\ A_j)$ and $(f_k,\ A_k)$ is minimized. The solution to selecting the N_1 $(f_i,\ A_i)$ points can be obtained using k-means algorithm [26]. In practice, we can randomly generate enough number of points in the $f - A$ plane to cover all the possibilities as the behavior of PEAs is frequency and displacement range dependent. Note that this method of choosing $U_{(ts)}$ is not trivial in the sense that the weight of f and A can vary. For example, if frequency is more important for the PEA output tracking tasks, we can scale the f axis by multiplying a factor 10 to f, thus the signal with frequency 10 and amplitude 1 corresponds to $(100,\ 1)$ instead of $(10,\ 1)$ in the $f - A$ plane. Moreover, some frequency components are better to be avoided (such as the resonance frequency of the piezo actuator) when designing $U_{(ts)}$ by removing the points with f_is close to them.

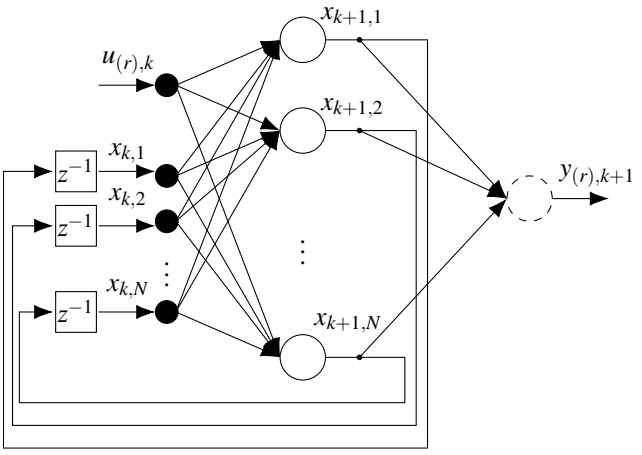

Fig. 1. Recurrent neural network

C. Linear Model Embedded with an Error Term (LME)

It is worth to note that for $S(f,\ A)$, the smaller f is, the longer the time series $S(f,\ A)$ will be. For example, if $f = 1$Hz and the sampling frequency is 10kHz, the length of time series is 10,000 to cover the entire period of this sinusoidal signal. From Eq. (2), it can be seen that increase of the length of time series leads to increase of the number of constrains, which implies that either more complex RNN is needed or the modeling accuracy will decrease. Therefore, the long time series should be avoided considering the modeling accuracy and the computation efficiency of Eq. (2). On the other hand, even if a lot of high frequency sinusoidal signals are to be included in $U_{(ts)}$, the length of $U_{(ts)}$ will not be affected too much. Therefore, the above method of constructing $U_{(ts)}$ may lead to that the low frequency dynamics of PEAs cannot be entirely captured. In addition, the drift effect of PEAs (usually in very low frequency range)

Authorized licensed use limited to: Iowa State University. Downloaded on June 26,2020 at 17:24:13 UTC from IEEE Xplore. Restrictions apply.

is hard to be modeled with RNN alone. Therefore, to address the aforementioned issues, we proposed to use the following linear model embedded with an error term (LME) to capture the residual dynamics (i.e., the dynamics not captured by RNN) including the low frequency dynamics and drift.

$$\begin{aligned} \eta_{k+1} &= A_e\eta_k + B_e u_{(l),k} + G\hat{e}_k \\ \hat{y}_k &= C_e\eta_k \end{aligned}, \tag{3}$$

where $u_{(l),k}$ is the input to the LME, $\hat{e}_k = y_k - \hat{y}_k$ is the model output error with y_k the PEA system output [25]. The dimensions of A_e, B_e, G and C_e are 2×2, 2×1, 2×1 and 1×2, respectively. Suppose the output of RNN (with parameters from the solution of Eq. (2)) is $Y_{(rts)}$ subject to the input $U_{(ts)}$ as shown in Fig. 2, then it is expected that the output of LME $\hat{Y}_{(ts)}$ will be very close to $Y_{(ts)}$ with the input $Y_{(rts)}$. Similar to the training of RNN, parameters in this LME will be determined through solving the following optimization problem with $Y_{(rts)}$ and $Y_{(ts)}$.

$$\begin{aligned} \min_{A_e,B_e,G,C_e} \quad & J_1 = ||Y_{(ts)} - \hat{Y}_{(ts)}|| \\ \text{subject to}: \quad & \eta_{k+1} = A_e\eta_k + B_e Y_{(rts),k} + G\hat{e}_k \\ & \hat{Y}_{(ts),k+1} = C_e\eta_{k+1} \\ & \hat{e}_{k+1} = Y_{(ts),k+1} - \hat{Y}_{(ts),k+1} \\ & \eta(0) = [0,0]^T \\ & \hat{e}_0 = 0, k = 0,1,\cdots \end{aligned}, \tag{4}$$

where $\hat{Y}_{(ts)} = [\hat{Y}_{(ts),1}, \hat{Y}_{(ts),2}, \cdots, \hat{Y}_{(ts),L}]^T$. Before training, $Y_{(rts)}$ and $Y_{(ts)}$ are known and $Y_{(ts)}$ is the same as that in Eq. (2).

D. Combine RNN and LME

Since the modeled dynamics in RNN and LME is limited by the frequency range of the training set, a low pass filter (LPF) is cascaded to the PEA model (i.e., RNN+LME) to avoid instability induced by ultra-high frequency dynamics. Suppose the LPF is formulated as

$$\begin{aligned} \beta_{k+1} &= \bar{A}\beta_k + \bar{B}u_k \\ z_k &= \bar{C}\beta_k \end{aligned}. \tag{5}$$

With $u_{(r),k} = z_k$ and $u_{(l),k} = y_{(r),k}$, Eqs. (1), (3), and (5) can be combined as the following nonlinear model,

$$\begin{aligned} \phi_{k+1} &= \begin{bmatrix} \beta_{k+1} \\ x_{k+1} \\ \eta_{k+1} \end{bmatrix} = F(\phi_k, u_k) = F\left(\begin{bmatrix} \beta_k \\ x_k \\ \eta_k \end{bmatrix}, u_k\right) \\ &= \begin{bmatrix} \bar{A}\beta_k + \bar{B}u_k \\ \tanh(W_1 x_k + B_2 + B_1\bar{C}\beta_k) \\ A_e\eta_k + B_e W_2 x_k + B_e B_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ G \end{bmatrix}\hat{e}_k, \\ \hat{y}_k &= H(\phi_k) = \begin{bmatrix} 0 & 0 & C_e \end{bmatrix}\begin{bmatrix} \beta_k \\ x_k \\ \eta_k \end{bmatrix} \end{aligned} \tag{6}$$

where u_k is the input to the nonlinear model. The block diagram of the model represented by Eq. (6) is shown in Fig. 3, i.e., the "Plant Model" in the dash box.

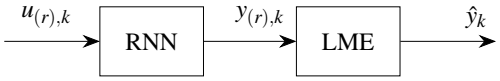

Fig. 2. Use RNN and LME to model the system dynamics

III. Nonlinear Predictive Control

The control scheme of the predictive controller based on the above system model (Eq. (6)) is shown in Fig. 3. Since the system model is nonlinear and the states are unavailable through measurement, a state estimator is designed to estimate the states during the tracking process. In this work, we choose unscented Kalman filter (UKF) over extended Kalman filter (EKF).

A. Nonlinear Estimator

In EKF, the nonlinear dynamics function is linearized at the current point (first order linearization) in each estimation and the conventional Kalman filter algorithms can be implemented. However, the approximation error through linearization may increase as the nonlinearity at the linearization point is severe. Instead, such an issue can be avoided in UKF as UKF generates a series of sample points which will be propagated through the nonlinear dynamics function, then the covariance matrix can be computed from the sampling points [27]. For the system model Eq. (6), the error term $\hat{e}_k$ is calculated before each estimation and updated at each sample instant. The estimation steps for the UKF used in this study are as follows:

Step 1: Initialize the parameters at $k = 0$

$$\begin{aligned} \hat{\phi}_0 &= E[\phi_0] \\ P_0 &= E[(\phi_0 - \hat{\phi}_0)(\phi_0 - \hat{\phi}_0)]^T \\ \hat{e}_0^- &= 0 \end{aligned} \tag{7}$$

Step 2: Generate sampling points (sigma points):

$$\begin{aligned} \chi_{k-1} &= [\hat{\phi}_{k-1} \quad \hat{\phi}_{k-1} + \gamma\sqrt{P_{k-1}} \quad \hat{\phi}_{k-1} - \gamma\sqrt{P_{k-1}}] \\ k &= 1,2,3,\cdots \end{aligned} \tag{8}$$

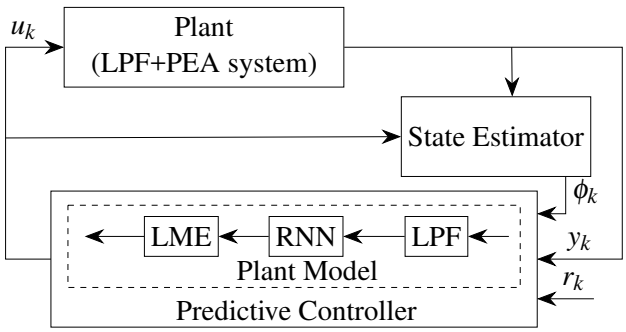

Fig. 3. Block diagram of the controller

Authorized licensed use limited to: Iowa State University. Downloaded on June 26,2020 at 17:24:13 UTC from IEEE Xplore. Restrictions apply.

Step 3: Time update:

$$\begin{aligned}
\chi_{k|k-1} =& [F(\chi_{1,k-1}, u_{k-1}), \cdots, F(\chi_{2N+1,k-1}, u_{k-1})] \\
\hat{\phi}_k^- =& \sum_{i=1}^{2N+1} W_m(i)\chi_{i,k|k-1} \\
P_k^- =& \sum_{i=1}^{2N+1} W_c(i)[\chi_{i,k|k-1} - \hat{\phi}_k^-][\chi_{i,k|k-1} - \hat{\phi}_k^-]^T + R_v \\
\mathscr{Y}_{k|k-1} =& [H(\chi_{1,k|k-1}), \cdots, H(\chi_{2N+1,k|k-1})] \\
\hat{y}_k^- =& \sum_{i=1}^{2N+1} W_m(i)\mathscr{Y}_{i,k|k-1}
\end{aligned} \tag{9}$$

Step 4: Measurement update and calculation of the error term:

$$\begin{aligned}
P_{\delta_y\delta_y} =& \sum_{i=1}^{2N+1} W_c(i)[\mathscr{Y}_{i,k|k-1} - \hat{y}_k^-][\mathscr{Y}_{i,k|k-1} - \hat{y}_k^-]^T + R_n \\
P_{\delta_\phi\delta_y} =& \sum_{i=1}^{2N+1} W_c(i)[\chi_{i,k|k-1} - \hat{\phi}_k^-][\mathscr{Y}_{i,k|k-1} - \hat{y}_k^-] \\
\mathscr{K}_k =& P_{\delta_\phi\delta_y} P_{\delta_y\delta_y}^{-1} \\
\hat{\phi}_k =& \hat{\phi}_k^- + \mathscr{K}_k(y_k - \hat{y}_k^-) \\
P_k =& P_k^- - \mathscr{K}_k P_{\delta_y\delta_y} \mathscr{K}_k^T \\
\hat{e}_k^- =& y_k - \hat{y}_k^-
\end{aligned} \tag{10}$$

Note that $\sqrt{P_{k-1}}$ is the square root of a matrix and can be calculated with Cholesky factorization, i.e., $P_{k-1} = \sqrt{P_{k-1}}\sqrt{P_{k-1}}^T$. R_v and R_n are the process and measurement noise covariance matrices, respectively. γ is a constant scalar, W_m and W_c are $(2N+1)\times 1$ vectors determined by the order of the system, N, $W_m(i)$ is the i^{th} element of W_m. $\chi_{i,k|k-1}$ is the i^{th} column of matrix $\chi_{k|k-1}$. The readers are referred to [27] for how to choose γ, W_m and W_c.

B. Predictive Controller

For the nonlinear predictive control, the following optimization problem is to be solved and the input u_k for the next sample time can be obtained from the solution.

$$\begin{aligned}
\min_U \mathscr{J} =& (\hat{Y}^f - R^f)^T(\hat{Y}^f - R^f) + \rho {U^f}^T D^T D U^f \\
D =& \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{11}$$

where $U^f = [u_k, u_{k+1}, \cdots, u_{k+N_c-1}]^T$, $\hat{Y}^f = [\hat{y}_{k+1}, \hat{y}_{k+2}, \cdots, \hat{y}_{k+N_h}]^T$ and $R^f = [r_{k+1}, r_{k+2}, \cdots, r_{k+N_c}, \cdots, r_{k+N_c}]^T_{N_h\times 1}$ are the future inputs, predicted outputs, and the reference (i.e., desired trajectory), respectively, with N_h the prediction horizon and N_c the control horizon. Notice that the current state ϕ_k and the previous input u_{k-1} are known when solving Eq. (11). Here we do not consider the constrains to the input for simplicity, but they can be incorporated later. Since the system dynamics is nonlinear, the objective function $\mathscr{J}$ cannot be written in the quadratic form as linear model predictive control. As an example, we choose gradient descent method to solve the unconstrained optimization problem (i.e., Eq. (11)).

Next, we show how to analytically compute the gradient $\frac{\partial \mathscr{J}}{\partial U}$. By using the analytical method, it is not only more precise without considering the increment Δ, but also more computationally efficient.

Let $E = (\hat{Y}^f - R^f)^T(\hat{Y}^f - R^f)$, then the key to compute $\frac{\partial \mathscr{J}}{\partial U}$ is to compute $\frac{\partial E}{\partial U}$ since calculation of the derivative of the other term is trivial. E can be expressed as

$$\begin{aligned}
E =& \sum_{i=1}^{N_h} E_i^2 = (\hat{y}_{k+1} - r_{k+1})^2 + (\hat{y}_{k+2} - r_{k+2})^2 + \cdots \\
&+ (\hat{y}_{k+N_h} - r_{k+N_c})^2 = (C_e\eta_{k+1} - r_{k+1})^2 \\
&+ (C_e\eta_{k+2} - r_{k+2})^2 + \cdots + (C_e\eta_{k+N_h} - r_{k+N_c})^2
\end{aligned} . \tag{12}$$

Thus, $\frac{\partial E}{\partial U^f}$ can be written as

$$\frac{\partial E}{\partial U^f} = \begin{bmatrix} \frac{\partial E}{\partial u_k} \\ \frac{\partial E}{\partial u_{k+1}} \\ \vdots \\ \frac{\partial E}{\partial u_{k+N_c-1}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E_1^2}{\partial u_k} + \frac{\partial E_2^2}{\partial u_k} + \frac{\partial E_3^2}{\partial u_k} + \cdots + \frac{\partial E_{N_h}^2}{\partial u_k} \\ \frac{\partial E_2^2}{\partial u_{k+1}} + \frac{\partial E_3^2}{\partial u_{k+1}} + \cdots + \frac{\partial E_{N_h}^2}{\partial u_{k+1}} \\ \vdots \\ \frac{\partial E_{N_c}^2}{\partial u_{k+N_c-1}} + \cdots + \frac{\partial E_{N_h}^2}{\partial u_{k+N_c-1}} \end{bmatrix}. \tag{13}$$

Next, we show how to compute $\frac{\partial E}{\partial u_k}$ and the rest follows. Note that $\phi_k = [\beta_k, x_k, \eta_k]^T$ and u_{k-1} are known. With Eq. (6), we have

$$\frac{\partial E_1^2}{\partial u_k} = 2E_1C_e\frac{\partial \eta_{k+1}}{\partial u_k} = 2E_1C_e\frac{\partial(A_e\eta_k + B_eW_2x_k + B_eB_3)}{\partial u_k} = 0. \tag{14}$$

Thus $\frac{\partial \eta_{k+1}}{\partial u_k} = 0$. Since

$$\frac{\partial x_{k+1}}{\partial u_k} = \frac{\partial \tanh(W_1x_k + B_2 + B_1\bar{C}\beta_k)}{\partial u_k} = 0, \tag{15}$$

then

$$\begin{aligned}
\frac{\partial E_2^2}{\partial u_k} =& 2E_2C_e\frac{\partial \eta_{k+2}}{\partial u_k} = 2E_2C_e\frac{\partial(A_e\eta_{k+1} + B_eW_2x_{k+1} + B_eB_3)}{\partial u_k} \\
=& 0.
\end{aligned} \tag{16}$$

Then $\frac{\partial \eta_{k+2}}{\partial u_k} = 0$. Similarly,

$$\begin{aligned}
\frac{\partial E_3^2}{\partial u_k} =& 2E_3C_e\frac{\partial \eta_{k+3}}{\partial u_k} = 2E_3C_e\frac{\partial(A_e\eta_{k+2} + B_eW_2x_{k+2} + B_eB_3)}{\partial u_k} \\
=& 2E_3C_eB_eW_2\frac{\partial x_{k+2}}{\partial u_k}.
\end{aligned} \tag{17}$$

With $\beta_{k+1} = \bar{A}\beta_k + \bar{B}u_k$, we have

$$\begin{aligned}
\frac{\partial x_{k+2}}{\partial u_k} =& \frac{\partial \tanh(W_1x_{k+1} + B_2 + B_1\bar{C}\beta_{k+1})}{\partial u_k} \\
=& \frac{\partial \tanh(\mathscr{X}_{k+1,k})}{\partial \mathscr{X}_{k+1,k}}\frac{\partial \mathscr{X}_{k+1,k}}{\partial u_k} = \frac{\partial \tanh(\mathscr{X}_{k+1,k})}{\partial \mathscr{X}_{k+1,k}}B_1\bar{C}\bar{B},
\end{aligned} \tag{18}$$

where $\mathscr{X}_{k+1,k} = W_1x_{k+1} + B_2 + B_1\bar{C}\beta_{k+1}$. Therefore, $\frac{\partial E_3^2}{\partial u_k}$ is computed with Eqs.(17) and (18). At the same time, the

Authorized licensed use limited to: Iowa State University. Downloaded on June 26,2020 at 17:24:13 UTC from IEEE Xplore. Restrictions apply.

resulting $\frac{\partial \eta_{k+3}}{\partial u_k}$, $\frac{\partial x_{k+2}}{\partial u_k}$ and $\frac{\partial \beta_{k+1}}{\partial u_k}$ can be used to compute $\frac{\partial E_4^2}{\partial u_k}$ which is

$$\frac{\partial E_4^2}{\partial u_k} = 2E_4 C_e (A_e \frac{\partial \eta_{k+3}}{\partial u_k} + B_e W_2 \frac{\partial x_{k+3}}{\partial u_k}). \tag{19}$$

Note that $\frac{\partial x_{k+3}}{\partial u_k}$ is related to $\frac{\partial x_{k+2}}{\partial u_k}$ and $\frac{\partial \beta_{k+1}}{\partial u_k}$ with

$$\begin{aligned} \frac{\partial x_{k+3}}{\partial u_k} &= \frac{\partial \tanh(W_1 x_{k+2} + B_2 + B_1 \bar{C} \beta_{k+2})}{\partial u_k} \\ &= \frac{\partial \tanh(\mathscr{X}_{k+2,k})}{\partial \mathscr{X}_{k+2,k}} \cdot W_1 \frac{\partial x_{k+2}}{\partial u_k} \cdot B_1 \bar{C} \bar{A} \frac{\partial \beta_{k+1}}{\partial u_k}. \end{aligned} \tag{20}$$

Therefore, $\frac{\partial x_{k+i+1}}{\partial u_k}$ can be calculated from the previous calculation of $\frac{\partial x_{k+i}}{\partial u_k}$.

Once the derivative can be computed, the gradient descent method can be implemented with

$$U^{f(m+1)} \leftarrow U^{f(m)} + \delta^{(m)} \frac{\partial \mathscr{J}}{\partial U^{f(m)}}, \tag{21}$$

where m is the iteration number and δ^m is the step length at m^{th} step. In practice, we keep $m < 15$ or smaller to improve the computation efficiency.

IV. Experiment Results and Discussion

For experimental validation, a PEA (Nano-OP30, Mad City Labs) was controlled with the proposed method to track given trajectories and the performance was compared with that of a PID controller. The experiment setup is shown in Fig. 4. All the signals were collected and generated through a data acquisition system (NI PCIe-6353, National Instruments) which was installed in the workstation (Intel Xeon W-2125, RAM 32GB). The controller was designed using MATLAB Simulink (MathWorks, Inc.).

The sampling frequency was chosen as 10kHz. A second order LPF with cutoff frequency of 1kHz was used. A 20th order RNN was chosen to model the system. To generate the training set for RNN, we set the frequency range to 0-400Hz and the amplitude range to 0-3.5μm, thus 100 (f_i, A_i) pairs were used to construct $U_{(ts)}$. Note that the chosen amplitude range was about 45% of the total displacement range of this PEA, at which the nonlinear hysteresis effect was quite significant—to demonstrate the efficacy of the proposed approach in dealing with PEA nonlinear dynamics. The generated training set is shown in Fig. 5.

The proposed RNN-based predictive control (RNNPC) was then applied on the PEA to track given trajectories which included sinusoidal trajectories with frequencies of 30Hz, 100Hz, 200Hz, and a Γ signal as below.

$$\begin{aligned} \Gamma(t) = [&0.8\sin(2\pi 5t + 1.5\pi) + 0.43\sin(2\pi 50t) + \\ &0.12\sin(2\pi 120t + 1.2\pi) + 0.3\sin(2\pi 180t + \pi)]/1.3 + 1.2. \end{aligned} \tag{22}$$

The parameters for the predictive controller were $N_h = 8$ and $N_c = 6$. The tracking errors E_{max} and E_{rms} were computed as did in [7].

The tracking performances of RNNPC and PID are compared in Table I. As shown in Table I, the proposed method

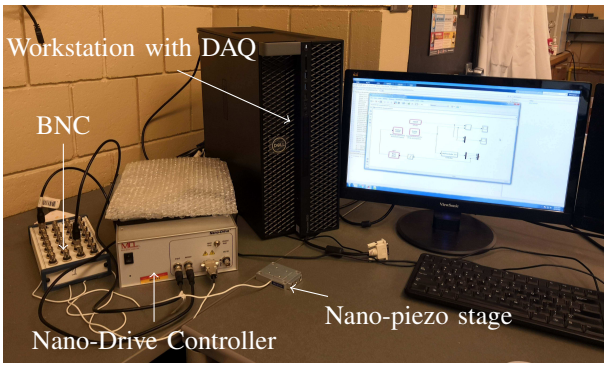

Fig. 4. Experimental setup

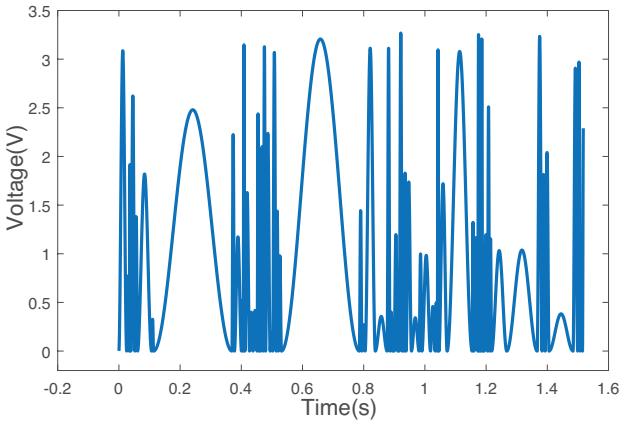

Fig. 5. Designed input $U_{(ts)}$ for training RNN.

reduced the tracking error including both E_{rms} and E_{max} by at least 50% for all trajectories tracked. Specifically, when tracking low frequency trajectories (30Hz, 100Hz and Γ), the tracking errors of the proposed approach are mostly less than 2%, which implies that the proposed system identification method—RNN+LME can precisely model the nonlinear PEA system. The tracking results in time domain for 100Hz sinusoidal signal and Γ are shown in Fig. 6, which again verified the effectiveness of the proposed method. The error increased when tracking high frequency trajectory: E_{rms} and E_{max} were both more than 6% when tracking the 200Hz desired trajectory, but were still 8% though (see Table I) much less than that of PID results. This downgrade of performance when tracking higher frequency trajectories is caused by the limited prediction and control horizon ($N_h = 8$ and $N_c = 6$) chosen in the predictive controller. The tracking accuracy can be greatly improved by using bigger N_h and N_c, however, that will increase the computation burden greatly. Therefore, to improve the tracking accuracy at higher frequency range, faster hardware such as field-programmable gate array (FPGA), should be used. Overall, the tracking accuracy of RNNPC, as a real-time control approach is satisfying.

Authorized licensed use limited to: Iowa State University. Downloaded on June 26,2020 at 17:24:13 UTC from IEEE Xplore. Restrictions apply.

TABLE I

TRACKING PERFORMANCE COMPARISON OF PID AND RNNPC WHEN TRACKING DIFFERENT TRAJECTORIES.

Refs.	30Hz		100Hz		200Hz		Γ	
$Error(\%)$	E_{rms}	E_{max}	E_{rms}	E_{max}	E_{rms}	E_{max}	E_{rms}	E_{max}
RNNPC	0.39	1.27	1.74	2.32	6.31	7.48	0.69	1.82
PID	2.29	2.69	7.93	9.19	16.61	19.29	3.30	4.81

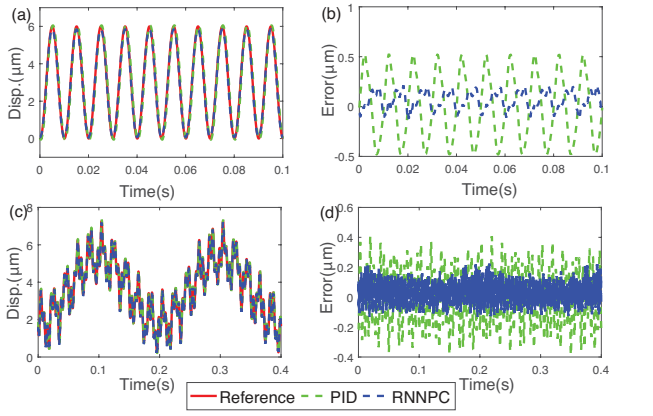

Fig. 6. (a) comparison of the tracking results for 100Hz sinusoidal signal using PID and RNNPC, (b)the tracking error, (c) comparison of the tracking results for Γ signal using PID and RNNPC, (d) the tracking error,

V. CONCLUSION

In this paper, we proposed a RNN-based predictive control (RNNPC) approach to achieve accurate real-time trajectory tracking of PEAs. Implementation of RNNPC to a PEA showed that the proposed method can achieve high tracking accuracy when the desired trajectory spanned over a broad frequency range. In addition, anything system which can be modeled by the RNN can be controlled with the proposed method.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation (NSF) (CMMI-1634592 and CMMI-1751503) and Iowa State University.

REFERENCES

[1] Y. F. Dufrêne, T. Ando, R. Garcia, D. Alsteens, D. Martinez-Martin, A. Engel, C. Gerber, and D. J. Müller, "Imaging modes of atomic force microscopy for application in molecular and cell biology," *Nature nanotechnology*, vol. 12, no. 4, p. 295, 2017.

[2] K. Mollaeian, Y. Liu, S. Bi, and J. Ren, "Atomic force microscopy study revealed velocity-dependence and nonlinearity of nanoscale poroelasticity of eukaryotic cells," *Journal of the mechanical behavior of biomedical materials*, vol. 78, pp. 65–73, 2018.

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

[4] F. Qin, D. Zhang, D. Xing, D. Xu, and J. Li, "Laser beam pointing control with piezoelectric actuator model learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[5] S. Tien, Q. Zou, and S. Devasia, "Iterative control of dynamics-coupling-caused errors in piezoscanners during high-speed afm operation," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 6, pp. 921–931, 2005.

[6] A. A. Eielsen, J. T. Gravdahl, and K. K. Leang, "Low-order continuous-time robust repetitive control: Application in nanopositioning," *Mechatronics*, vol. 30, pp. 231–243, 2015.

[7] 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 Transactions on Mechatronics*, vol. 18, no. 6, pp. 1767–1777, 2013.

[8] B. Altın, J. Willems, T. Oomen, and K. Barton, "Iterative learning control of iteration-varying systems via robust update laws with experimental implementation," *Control Engineering Practice*, vol. 62, pp. 36–45, 2017.

[9] S. Xie and J. Ren, "Note: Precision control of nano-positioning stage: An iterative learning-based model predictive control approach," *Review of Scientific Instruments*, vol. 89, no. 7, p. 076103, 2018.

[10] S. Xie and J. Ren, "Iterative learning-based model predictive control for precise trajectory tracking of piezo nanopositioning stage," in *2018 Annual American Control Conference (ACC)*, pp. 2922–2927, IEEE, 2018.

[11] S. Xie and J. Ren, "High-speed afm imaging via iterative learning-based model predictive control," *Mechatronics*, vol. 57, pp. 86–94, 2019.

[12] M. S. Rana, H. R. Pota, and I. R. Petersen, "The design of model predictive control for an afm and its impact on piezo nonlinearities," *European Journal of Control*, vol. 20, no. 4, pp. 188–198, 2014.

[13] Q. Xu, "Digital sliding-mode control of piezoelectric micropositioning system based on input-output model.," *IEEE Trans. Industrial Electronics*, vol. 61, no. 10, pp. 5517–5526, 2014.

[14] Y. Pan, C. Yang, L. Pan, and H. Yu, "Integral sliding mode control: performance, modification and improvement," *IEEE Transactions on Industrial Informatics*, 2017.

[15] 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. Automation Science and Engineering*, vol. 13, no. 1, pp. 313–332, 2016.

[16] R. C. Smith and Z. Ounaies, "A domain wall model for hysteresis in piezoelectric materials," *Journal of intelligent material systems and structures*, vol. 11, no. 1, pp. 62–79, 2000.

[17] J. W. Macki, P. Nistri, and P. Zecca, "Mathematical models for hysteresis," *SIAM review*, vol. 35, no. 1, pp. 94–123, 1993.

[18] 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 Transactions on Mechatronics*, vol. 18, no. 3, pp. 895–904, 2013.

[19] G. Song, J. Zhao, X. Zhou, and J. A. De Abreu-García, "Tracking control of a piezoceramic actuator with hysteresis compensation using inverse preisach model," *IEEE/ASME transactions on mechatronics*, vol. 10, no. 2, pp. 198–209, 2005.

[20] M. Al Janaideh, S. Rakheja, and C.-Y. Su, "An analytical generalized prandtl–ishlinskii model inversion for hysteresis compensation in micropositioning control," *IEEE/ASME Transactions on mechatronics*, vol. 16, no. 4, pp. 734–744, 2011.

[21] M.-S. Tsai and J.-S. Chen, "Robust tracking control of a piezoactuator using a new approximate hysteresis model," *Journal of dynamic systems, measurement, and control*, vol. 125, no. 1, pp. 96–102, 2003.

[22] M. Quant, H. Elizalde, A. Flores, R. Ramírez, P. Orta, and G. Song, "A comprehensive model for piezoceramic actuators: modelling, validation and application," *Smart Materials and Structures*, vol. 18, no. 12, p. 125011, 2009.

[23] L. Cheng, W. Liu, Z.-G. Hou, J. Yu, and M. Tan, "Neural-network-based nonlinear model predictive control for piezoelectric actuators," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 12, pp. 7717–7727, 2015.

[24] A. M. Schäfer and H.-G. Zimmermann, "Recurrent neural networks are universal approximators," *International journal of neural systems*, vol. 17, no. 04, pp. 253–263, 2007.

[25] C. Wang, A. Ohsumi, and I. Djurovic, "Model predictive control of noisy plants using kalman predictor and filter," in *TENCON'02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 3, pp. 1404–1407, IEEE, 2002.

[26] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[27] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter," *Kalman filtering and neural networks*, pp. 221–280, 2001.

Authorized licensed use limited to: Iowa State University. Downloaded on June 26,2020 at 17:24:13 UTC from IEEE Xplore. Restrictions apply.