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

Combining Learning and Model Based Control: Case Study for Single-Input Lotka-Volterra System

W. Steven Gray[†] G. S. Venkatesh[†] Luis A. Duffaut Espinosa[‡]

Abstract— **A hybrid control architecture for nonlinear dynamical systems is described which combines the advantages of model based control with those of real-time learning. The basic idea is to generate input-output data from an error system involving the plant and a proposed model. A discretized Chen-Fliess functional series is then identified from this data and used in conjunction with the model for predictive control. This method builds on the authors' previous work on model-free control of a single-input, single-output Lotka-Volterra system. The problem is revisited here, but now with the introduction of a model for the dynamics. The single-input, multiple-output version of the problem is also investigated as a way to enhance closed-loop performance.**

I. Introduction

Artificial neural networks (ANN) have traditionally been the backbone of machine learning. A typical feedforward ANN learning unit (neuron) involves weighted sums of inputs and a nonlinear thresholding/activation function. If this function is real analytic, say a sigmoidal function, then the input-output map of a neuron is mathematically equivalent to a multivariable Taylor series. Learning in this case amounts to tuning the coefficients (weights) to fit a static memoryless map to the input-output data in order to discriminate between different regions of the decision space. On the other hand, if the intent is to build an approximation of a dynamical system for the purpose of control, then so called recurrent networks are a more natural choice as they are known to provide universal functional approximations for specific classes of systems [1], [12]–[14]. While these biologically inspired approximation schemes certainly have their strengths and a certain intuitive appeal, they also have limitations. For example, physics based models often provide key physical insight into the system to be controlled. So completely discarding such models in favor of a pure learning based control system can be counterproductive. This raises the question of whether there exists other types of learning systems that are better suited for control as they accommodate modeling information in some form.

The main goal of this paper is to take an incremental step in this direction by proposing a hybrid control architecture for nonlinear dynamical systems which combines the advantages of model based control with those of real-time learning and data-driven control. The basic idea is to generate input-output data from an error system involving the plant and a proposed model, which can then be learned and used in conjunction with the model for predictive control. The approach is distinct from model based adaptive control in that the plant is not made to track the output of the reference model, and there is no adaptation of the model. Instead, the desired output is given and the error dynamics are learned using a generic input-output representation known as a *Chen-Fliess functional series* or *Fliess operator*. This weighted sum of iterated integrals of the control input employs a set of coefficients that completely characterizes the error system. The coefficients are estimated in real-time using a minimum mean-square error estimator after truncating the series to some suitable length based on the desired accuracy. In which case, the neuron in a conventional ANN is replaced here by a new type of learning unit to produce an approximate model of the error system. It has been known since the 1980s that a nonlinear dynamical system under certain conditions always has a Fliess operator representation [2], [3], [10]. This is exactly analogous to representing an analytic function in terms of its Taylor series, as done implicitly with feedforward ANNs, only in this case the power series representing the input-output functional is *noncommutative*. The main advantages to this approach are three-fold:

1) Fliess operator representations are known to be universal and unique;
2) the structural assumptions concerning the error system are minimal;
3) fewer parameters need to be estimated than, for example, with high dimensional state space models.

Up until recently, a major impediment to taking such an approach has been that Fliess operators are exclusively used to represent continuous-time systems, but data is generally collected in discrete-time. However, it was shown in [5] that Fliess operators can be accurately discretized with a priori error bounds to produce discrete-time Fliess operators. This method was employed by the authors in [6] to give a purely data-driven solution to a tracking problem involving single-input, single-output (SISO) Lotka-Volterra system. The problem is revisited here, but now with the introduction of a model for the dynamics. The single-input, multiple-output (SIMO) version of the problem is also investigated as a way to enhance closed-loop performance. The main goal is to quantify what advantages the introduction of a reference model provides and to identify directions for future improvements.

The paper is organized as follows. In the next section, to keep the paper self-contained, a brief summary of the key concepts used from [5] and [6] is given. The main results for the SISO case are given in Section III. The SIMO case is presented in Section IV. Finally, the conclusions and directions for future work are given in the last section.

[†]Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, Virginia 23529 USA

[‡]Department of Electrical and Biomedical Engineering, University of Vermont, Burlington, Vermont 05405 USA

II. PRELIMINARIES

An *alphabet* $X = \{x_0, x_1, \ldots, x_m\}$ is any nonempty finite set of noncommuting symbols called *letters*. A *word* $\eta = x_{i_1} \cdots x_{i_k}$ is a finite sequence of letters from X. The length of η is defined to be $|\eta| = k$. The word of length zero is the empty word $\emptyset$. Let X^k be the set of words having length k, and define $X^* = \bigcup_{k\geq 0} X^k$. X^* is a monoid under the catenation product. Any mapping $c : X^* \to \mathbb{R}^\ell : \eta \mapsto (c, \eta)$ is called a *formal power series*. It is often convenient to write c as the formal sum $c = \sum_{\eta\in X^*}(c, \eta)\eta$. The collection of noncommuting formal power series over X forms an associative $\mathbb{R}$-algebra under catenation denoted here by $\mathbb{R}^\ell\langle\langle X\rangle\rangle$.

A. Discrete-time Fliess Operators as Universal Approximators

The goal of this section is to define the class of discrete-time approximators for analytic input-output systems to be employed by the learning unit described in the next section. See [5] for additional details. Inputs in this case are real-valued sequences from the normed linear space

$$l_\infty^{m+1}[N_0] := \{\hat{u} = (\hat{u}(N_0), \hat{u}(N_0 + 1), \ldots) : \|\hat{u}\|_\infty < \infty\},$$

where $\hat{u} := [\hat{u}_0\, \hat{u}_1 \cdots \hat{u}_m]^T$, $|\hat{u}(N)| := \max_{i\in\{0,1,\ldots,m\}} |\hat{u}_i(N)|$, and $\|\hat{u}\|_\infty := \sup_{N\geq N_0} |\hat{u}(N)|$.

Definition 1: Given a generating series $c \in \mathbb{R}^\ell\langle\langle X\rangle\rangle$, the corresponding **discrete-time Fliess operator** is defined as

$$\hat{F}_c[\hat{u}](N) = \sum_{\eta\in X^*} (c, \eta) S_\eta[\hat{u}](N) \tag{1}$$

for any $N \geq 1$, where

$$S_{x_i\eta}[\hat{u}](N) = \sum_{k=1}^{N} \hat{u}_i(k) S_\eta[\hat{u}](k) \tag{2}$$

with $x_i \in X$, $\eta \in X^*$, and $\hat{u} \in l_\infty^{m+1}[1]$. By assumption, $S_\emptyset[\hat{u}](N) := 1$.

Following [9], select some fixed $u \in L_1^m[0, T]$ with $T > 0$ finite. Choose an integer $L \geq 1$, let $\Delta := T/L$ and define the sequence of real numbers

$$\hat{u}_i(N) = \int_{(N-1)\Delta}^{N\Delta} u_i(t)\, dt, \quad i = 0, 1, \ldots, m, \tag{3}$$

where $N \in [1, L]$. Assume $u_0 = 1$ so that $\hat{u}_0(N) = \Delta$. The operator (1) is next truncated to words of length J, that is,

$$\hat{y}(N) = \hat{F}_c^J[\hat{u}](N) := \sum_{j=0}^{J} \sum_{\eta\in X^j} (c, \eta) S_\eta[\hat{u}](N), \tag{4}$$

since numerically only finite sums can be computed. The main assertion proved in [5] is that the class of truncated, discrete-time Fliess operators acts as a set of universal approximators with computable error bounds for their continuous-time counterparts described in [2], [3], [10].

B. Learning Unit Based on Discrete-Time Fliess Operator

In this section a type of learning unit suitable for dynamical systems is briefly described. See [6] for a more complete treatment. The focus is on the SISO case. First observe that (4) can be written in the form

$$\hat{y}(N) = \phi^T(N)\theta_0, \quad N \geq 1, \tag{5}$$

where

$$\phi(N) = [S_{\eta_1}[\hat{u}](N)\; S_{\eta_2}[\hat{u}](N) \cdots S_{\eta_l}[\hat{u}](N)]^T \tag{6a}$$

$$\theta_0 = [(c, \eta_1)\; (c, \eta_2) \cdots (c, \eta_l)]^T \tag{6b}$$

with $l = 2^{J+1} - 1$ and assuming a lexicographical ordering on the words where $x_0 < x_1$. If some estimate of θ_0 is available at time $N - 1$, say $\hat{\theta}(N - 1)$, then (5) gives a corresponding estimate of $\hat{y}(N)$:

$$\hat{y}_p(N) := \phi^T(N)\hat{\theta}(N - 1). \tag{7}$$

The coefficients can be updated in real-time from input-output measurements using a standard least-squares algorithm [4, p. 65]. In practice there is no guarantee that the estimates $\hat{\theta}(N)$, $N \geq 1$ will converge to c in any sense, but this is not an issue as the only objective is to ensure that the underlying input-output map is well approximated by $\hat{F}^J_{\hat{\theta}(N)}$. The theory presented in the previous subsection guarantees that if the underlying system has a Fliess operator representation then the class of discrete-time approximators provides at least one possible limit point.

To implement a real-time learning algorithm, a difference equation is needed to update the data vector $\phi(N)$ in (6a). For convenience define $\hat{u}_\xi(N) = \hat{u}_{i_k}(N) \cdots \hat{u}_{i_1}(N)$ for any $\xi = x_{i_k} \cdots x_{i_1} \in X^*$ and $N \geq 1$. Let $\hat{u}_\emptyset(N) := 1$. The next lemma shows that the update equation for the iterated sums defined in (2) is computed in terms of the Cauchy product of the formal power series $c_u(N + 1) := \sum_{\eta\in X^*} \hat{u}_\eta(N + 1)\eta$ and $S[\hat{u}](N) := \sum_{\eta\in X^*} S_\eta[\hat{u}](N)\eta$.

Lemma 1: [6] For any $\eta \in X^*$ and $N \geq 1$

$$\begin{aligned} S_\eta[\hat{u}](N + 1) &= (c_u(N + 1)S[\hat{u}](N), \eta) \\ &= \sum_{\eta=\xi\nu} \hat{u}_\xi(N + 1) S_\nu[\hat{u}](N). \end{aligned}$$

Therefore,

$$\phi(N + 1) = \begin{bmatrix} \hat{u}_\emptyset(N + 1)S_\emptyset[\hat{u}](N) \\ \hat{u}_\emptyset(N + 1)S_{x_0}[\hat{u}](N) + \hat{u}_{x_0}(N + 1)S_\emptyset[\hat{u}](N) \\ \hat{u}_\emptyset(N + 1)S_{x_1}[\hat{u}](N) + \hat{u}_{x_1}(N + 1)S_\emptyset[\hat{u}](N) \\ \sum_{x_0^2=\xi\nu} \hat{u}_\xi(N + 1)S_\nu[\hat{u}](N) \\ \sum_{x_0x_1=\xi\nu} \hat{u}_\xi(N + 1)S_\nu[\hat{u}](N) \\ \sum_{x_1x_0=\xi\nu} \hat{u}_\xi(N + 1)S_\nu[\hat{u}](N) \\ \sum_{x_1^2=\xi\nu} \hat{u}_\xi(N + 1)S_\nu[\hat{u}](N) \\ \vdots \\ \sum_{\eta_\ell=\xi\nu} \hat{u}_\xi(N + 1)S_\nu[\hat{u}](N) \end{bmatrix}. \tag{8}$$

In summary, equations (7) and (8) along with the mean-square estimation algorithm provide a fully inductive algorithm to predict $\hat{y}(N + 1)$ given the next input $\hat{u}(N + 1)$:

$$\hat{y}_p(N + 1) = \phi^T(N + 1)\hat{\theta}(N)$$

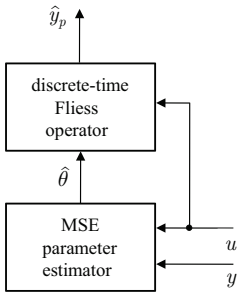

Fig. 1: Learning unit

$$
\begin{aligned}
&= \sum_{i=1}^{\ell} \left[\sum_{\eta_i = \xi\nu} \hat{u}_\xi(N+1) S_\nu[\hat{u}](N) \right] \hat{\theta}_i(N) \\
&= \sum_{i=1}^{\ell} \sum_{\eta_i = \xi\nu} \Delta^{|\xi|_{x_0}} \hat{u}^{|\xi|_{x_1}}(N+1) S_\nu[\hat{u}](N) \hat{\theta}_i(N) \\
&= \mathcal{U}^T(N+1) \mathcal{S}(N) \hat{\theta}(N) \\
&=: \mathcal{Q}(\hat{u}(N+1)),
\end{aligned}
$$

where

$$\mathcal{U}^T(N+1) := [1 \;\; \hat{u}(N+1) \;\; \hat{u}^2(N+1) \cdots \hat{u}^J(N+1)]$$

and $\mathcal{S}(N) \in \mathbb{R}^{(J+1)\times \ell}$ with entries depending solely on the sums $\{S_\eta[\hat{u}](N) : |\eta| \leq J\}$. So $\mathcal{Q}(\hat{u}(N+1))$ is a polynomial in $\hat{u}(N+1)$ of at most degree J. The corresponding learning unit is shown in Figure 1. Continuous-time input-output data u and y enters the parameter estimator and is discretized to produce samples $\hat{u}$ and $\hat{y}$ as in (3)-(4). The parameter estimate and input are then passed to the discrete-time Fliess operator block. The iterated sums are updated, and $\hat{y}_p$ is now available in terms of polynomial $\mathcal{Q}$. A more explicit implementation of the algorithm in terms of only matrix and vector type operations can be found in [8]. A straightforward analysis of this implementation shows that the complexity of the learning unit is $\mathcal{O}((m+1)^{2J})$, which is exponential in J. If certain vectors in the implementation called *order vectors* are stored as a dictionary, then the computation time can be reduced, but the complexity still remains exponential in J.

C. Lotka-Volterra Systems

Population dynamics of n species in competition can be modeled by the Lotka-Volterra system

$$\dot{z}_i = \beta_i z_i + \sum_{j=1}^{n} \alpha_{ij} z_i z_j, \;\; i = 1, \ldots, n, \tag{9}$$

where z_i is the biomass of the i-th species, β_i denotes the growth rate of the i-th species, and α_{ij} describes the influence of the j-th species on the i-th species. More recently in [11] it was shown that a power network, where each node voltage z_i is regulated by a quadratic droop controller, has dynamics governed by a Lotka-Volterra model.

Consider the case where a subset of system parameters in (9) can be actuated and thus viewed as inputs u_i, $i = 1, \ldots, m$. Assume some set of output functions is given

$$y_j = h_j(z), \;\; j = 1, \ldots, \ell. \tag{10}$$

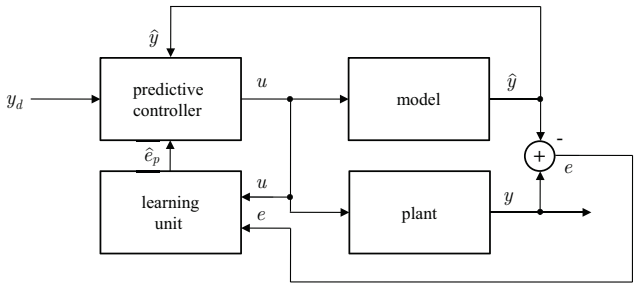

Fig. 2: Closed-loop system with SISO predictive controller and one learning unit

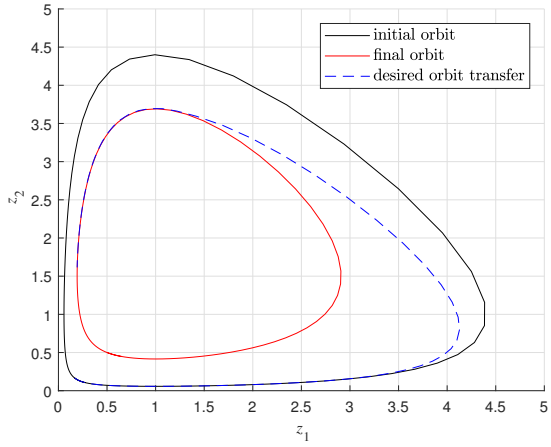

Fig. 3: Desired orbit transfer given full knowledge of the plant

Since the inputs enter the dynamics linearly, it is clear that (9)-(10) constitutes an analytic control affine state space system. In which case, the input-output system $u \mapsto y$ must have an underlying Fliess operator representation F_c with generating series c computable from the Lotka-Volterra dynamics and a given initial condition [2], [3], [10]. Of particular interest here will be the special case of a predator-prey system

$$
\begin{aligned}
\dot{z}_1 &= \beta_1 z_1 - \alpha_{12} z_1 z_2 \\
\dot{z}_2 &= -\beta_2 z_2 + \alpha_{21} z_1 z_2,
\end{aligned}
$$

where z_1 is assumed to be the prey species, and (9) has been re-parameterized so that $\beta_i, \alpha_{ij} > 0$. This positive system has precisely two equilibria when all the parameters are fixed, namely, a stable equilibrium at the origin and a saddle point at $z_e = (\beta_2/\alpha_{21}, \beta_1/\alpha_{12})$. The vector fields in the first quadrant are complete and give concentric periodic trajectories about z_e.

III. Predictive Controller Based on Learning and a SISO Model

Suppose y_d is a desired output known to be in the range of a given plant with an underlying but unknown Fliess operator representation $F_c : u \mapsto y$ (see [7]). In addition, assume that some control affine state space model is available for the plant. In applications it is most likely that y_d is selected using this model with perhaps the aid of some expert knowledge. Already this implies that the model should not

TABLE I: Discretization parameters in (3) and (4)

L	J	T	Δ
100	3	6	0.06

TABLE II: Control system parameters in (7), (11), and (12)

controller type	$\hat{\theta}(0)$	$\bar{u}$
model free	0	2
with model	0	1.4

be too different from the plant, otherwise y_d may not be in its range. When both the plant and model are given the same input as shown in Figure 2, a modeling error $e = y - \hat{y}$ is generated. This signal and the applied input are then fed to a learning unit of the type presented in the previous section in order to learn the error dynamics, which in this case must also have a Fliess operator representation. At any time instant then the output of the plant is approximated by $\hat{y}+\hat{e}_p = \hat{y}+\mathcal{Q}(\hat{u})$. A suitable input u for tracking y_d can be approximated by a piecewise constant function taking values for $N \in [1, L]$ equivalent to

$$\hat{u}(N) := \underset{|\hat{u}(N)|\leq\bar{u}}{\arg\min} \|y_d(N\Delta) - [\hat{y}(N\Delta) + \mathcal{Q}(\hat{u}(N))]\|^2 \quad (11)$$

for some fixed $\bar{u} > 0$. The MatLab command `fmincon` is used here to compute these local minima over a given finite interval in $\mathbb{R}$.

Consider as an example a plant modeled by a predator-prey system with $u(t) = \beta_1(t)$ so that

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} -\alpha_{12} z_1 z_2 \\ -\beta_2 z_2 + \alpha_{21} z_1 z_2 \end{bmatrix} + \begin{bmatrix} z_1 \\ 0 \end{bmatrix} u$$

with initial conditions $z_1(0) = z_{1,0}$ and $z_2(0) = z_{2,0}$. The output y can be taken to be either z_1 or z_2. Assume the true plant has all the constant system parameters equal to unity, and the positivity constraint on u will not be enforced. Initializing the plant at $[z_{1,0}, z_{2,0}]^T = [1/6, 1/6]^T$ and applying the constant input $u(t) = \beta_1(t) = 1$ yields the orbit shown in black in Figure 3. Suppose in the biological setting the goal is to reduce the large population fluctuations which lead to very low predator and prey populations appearing during certain portions of this orbit. A new desired orbit can be identified by setting $\beta_1 = 1.5$ and $[z_{1,0}, z_{2,0}]^T = [1/2, 1/2]^T$ to produce the red orbit in Figure 3. A desired orbit transfer is the dashed blue line shown in Figure 3. It was designed using the true dynamics of the plant and represents the ideal situation. In the simulations to follow, however, the desired trajectory will be designed instead using the assumed model, which can be different from the actual model. This will provide the desired signal y_d for the chosen output channel. The orbit transfer maneuver is terminated once the error between the current state and the desired trajectory is less than 0.05.

The starting point is the model free case. So in effect, $\hat{y} = 0$, and therefore the learning unit must learn the plant dynamics not the error dynamics. This case was studied in [6] when $y = z_1$, but here for comparison a more complete analysis is provided. The parameters used in the simulations

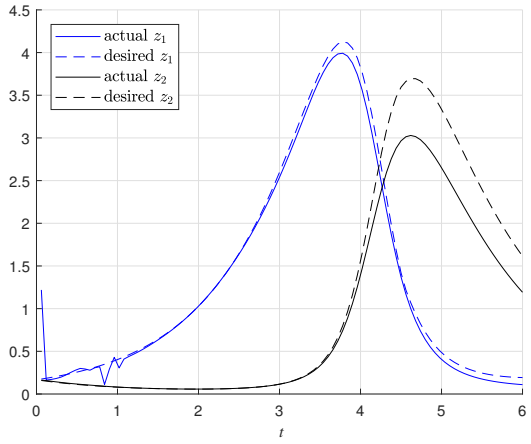

Fig. 4: Trajectories using model free control of z_1

are summarized in Tables I and II. Figure 4 shows the output tracking performance when z_1 is controlled. The corresponding RMS of the relative errors per time sample are given in Tables III and IV. The tracking errors for both z_1 and z_2 are quite acceptable for many applications. In contrast, the system is *entirely* unable to track the desired trajectory when z_2 is used as the output. The source of the problem is that the input u controls z_2 *indirectly* through z_1 in the state space model. During the initial part of the orbit, z_2 is quite small and therefore $\dot{z}_2$ is also small. Hence, the input-output map $u \mapsto z_2$ is not sufficiently rich to train the learning unit. Note the large RMS tracking errors in Table IV. At least for this specific system, this illustrates the limits of this model free approach and motivates the introduction of a model into this learning system.

Consider next the other extreme case where a model is given that exactly matches the plant. In this situation the error signal is always zero so that the learning unit always predicts zero error. Hence, the learning is completely removed

TABLE III: Summary of tracking errors when z_1 is controlled, i.e., $y = z_1$

model type	tracking error z_1	tracking error z_2
model free	0.6015	0.3298
exact model	1.5466×10^{-5}	1.1654×10^{-4}
-5% β_2 error model	0.0100	0.1574
-5% α_{12} error model	0.0710	0.0030
-5% α_{21} error model	0.0235	0.1184

TABLE IV: Summary of tracking errors when z_2 is controlled, i.e., $y = z_2$

model type	tracking error z_1	tracking error z_2
model free	1.9695	1.6802
exact model	0.0067	1.932×10^{-7}
-5% β_2 error model	0.5435	0.0103
-5% α_{12} error model	0.2077	0.0052
-5% α_{21} error model	0.1623	0.0164

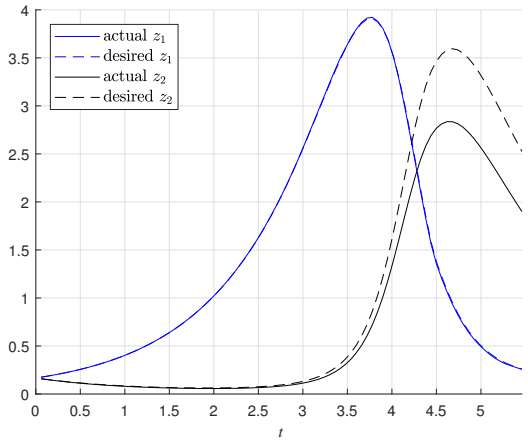

Fig. 5: Trajectories using $\beta_2 = 0.95$ model for control of z_1

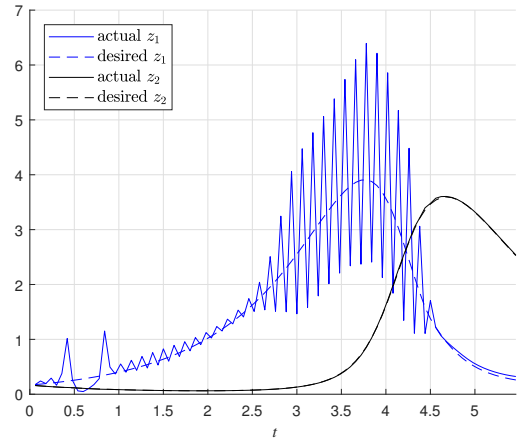

Fig. 6: Trajectories using $\beta_2 = 0.95$ model for control of z_2

from the picture, and the performance of the closed-loop is determined only by the optimization method used to solve (11). Interestingly, no matter which output is controlled, the other output matches its desired trajectory very accurately. The RMS tracking errors are given in Tables III and IV, but are not discernable on the trajectory plots.

The more representative case is where a model of the plant is given, but it has some parametric errors, for example. Consider the case where $\beta_2 = 0.95$, i.e., a -5% error in one parameter. The corresponding trajectories are shown in Figures 5 and 6 when z_1 and z_2 are controlled, respectively. The RMS tracking errors are given in Tables III and IV for this case and when $\alpha_{12} = 0.95$ and $\alpha_{21} = 0.95$. When z_1 is controlled, there is an order of magnitude improvement in tracking performance over the model free case. But the more interesting case is when z_2 is controlled. For example, with the introduction of the $\beta_2 = 0.95$ model, z_2 can now be accurately tracked. However, large oscillations appear in the uncontrolled z_1 state. The applied input is shown in Figure 7. This asymmetry between z_1 control and z_2 control is again due to the way the input enters the system. This motivates the need for a SIMO approach as described in the next section.

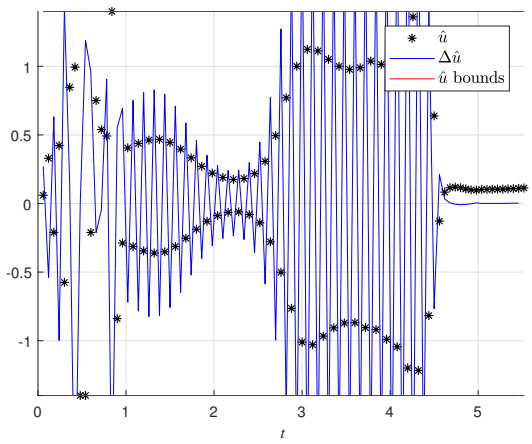

Fig. 7: Applied input to $\beta_2 = 0.95$ model for control of z_2

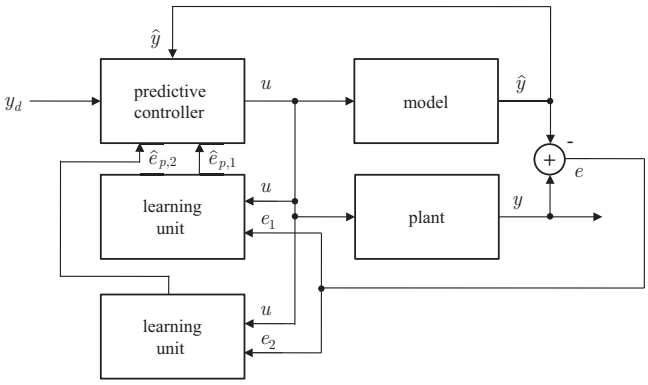

Fig. 8: Closed-loop system with SIMO predictive controller and two learning units

IV. Predictive Controller Based on Learning and a SIMO Model

The proposed closed-loop system for a SIMO system involving two outputs is shown in Figure 8. In this architecture, there is a SISO learning unit for each input-output pair (u, y_i), $i = 1, 2$. The predicted error vector $e_p = [\hat{e}_{p,1} \;\; \hat{e}_{p,2}]^T$ is then passed to the predictive controller which computes the next control input by solving the optimization problem

$$\hat{u}(N) := \underset{|\hat{u}(N)| \leq \bar{u}}{\arg\min}\, y_e^T(N) W y_e(N), \tag{12}$$

where

$$y_{e,i}(N) := y_{d,i}(N\Delta) - [\hat{y}_i(N\Delta) + \mathcal{Q}_i(\hat{u}(N))], \;\; i = 1, 2,$$

and $W = [w_{11} \; w_{12}; w_{21} \;\; w_{22}]$ is a symmetric semi-positive definite weighting matrix. In this context, $W = [1 \; 0; 0 \; 0]$

TABLE V: Summary of tracking errors for the SIMO controller

model type	tracking error z_1	tracking error z_2
-5% β_2 error model	0.0705	0.0886
-5% α_{12} error model	0.0092	0.0016
-5% α_{21} error model	0.0941	0.1182

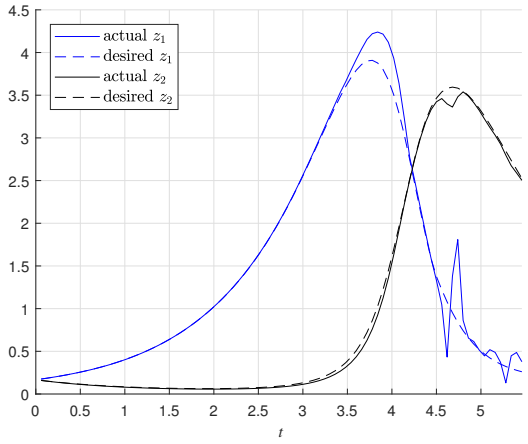

Fig. 9: Trajectories using $\beta_2 = 0.95$ SIMO model in the case where $W = [0.01\ 0; 0\ 1]$

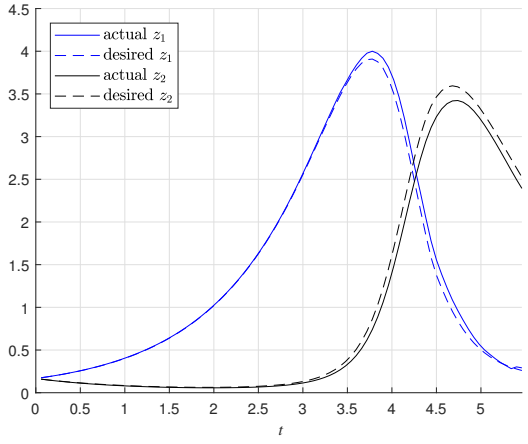

Fig. 10: Trajectories using $\beta_2 = 0.95$ SIMO model in the case where $W = [0.5\ 0.25; 0.25\ 2]$

corresponds to SISO control of z_1, and $W = [0\ 0; 0\ 1]$ is SISO control of z_2.

Reconsider the last case in the previous section. The system response is shown in Figure 9 when $W = [0.01\ 0; 0\ 1]$. Interestingly, this very small inclusion of z_1 in the optimization problem *completely* suppresses the oscillations in z_1 shown in Figure 6 with only a modest increase in the z_2 tracking error. If $W = [0.5\ 0.25; 0.25\ 2]$ then these tracking results can be further improved as shown in Figure 10. The tracking error for this case and other parametric errors is shown in Table V.

V. Conclusions and Future Work

A hybrid control architecture for nonlinear dynamical systems was described which combines knowledge of the dynamical system to be controlled and real-time learning units. Using input-output data generated from an error system involving the plant and the proposed model, a truncated discrete-time Fliess operator for each input-output pair is identified. From this representation of the error system and the given model, a nonlinear predictive controller is implemented. The method was tested on a SISO predator-prey system and a SIMO version of this system. For this particular system, SISO control works quite well for one channel but to the detriment of the other. The introduction of a second learning unit for the SIMO case significantly enhanced closed-loop performance for both channels.

Future work will include a full multivariable version of this new control architecture. Other types of plants should be employed to see how well the conclusions from this study generalize. The case where noise is present in the system should also be investigated.

Acknowledgments

This research was supported by the National Science Foundation under grants CMMI-1839378 and CMMI-1839387.

References

[1] P. Baldi and K. Hornik, Universal approximation and learning of trajectories using oscillators, in *Advances in Neural Information Processing Systems*, D. Touretzky, M. Mozer, and M. Hasselmo, Eds., vol. 8, MIT Press, Cambridge, MA, 1996, pp. 451–457.

[2] M. Fliess, Fonctionnelles causales non linéaires et indéterminées non commutatives, *Bull. Soc. Math. France*, 109 (1981) 3–40.

[3] M. Fliess, Réalisation locale des systèmes non linéaires, algèbres de Lie filtrées transitives et séries génératrices non commutatives, *Invent. Math.*, 71 (1983) 521–537.

[4] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediciton and Control*, Dover Publications, Inc., Mineola, NY, 2009.

[5] W. S. Gray, L. A. Duffaut Espinosa, and K. Ebrahimi-Fard, Discrete-time approximations of Fliess operators, *Numer. Math.*, 137 (2017) 35–62.

[6] W. S. Gray, L. A. Duffaut Espinosa, and L. T. Kell, Data-driven SISO predictive control using adaptive discrete-time Fliess operator approximations, *Proc. 21st Inter. Conf. on System Theory, Control and Computing*, Sinaia, Romania, 2017, pp. 383–388.

[7] W. S. Gray, L. A. Duffaut Espinosa, and M. Thitsa, Left inversion of analytic nonlinear SISO systems via formal power series methods, *Automatica*, 50 (2014) 2381–2388.

[8] W. S. Gray, G. S. Venkatesh, and L. A. Duffaut Espinosa, Discrete-time Chen series for time discretization and machine learning, *Proc. 53rd Conf. on Information Sciences and Systems*, Baltimore, Maryland, 2019.

[9] L. Grüne and P. E. Kloeden, Higher order numerical schemes for affinely controlled nonlinear systems, *Numer. Math.*, 89 (2001) 669–690.

[10] A. Isidori, *Nonlinear Control Systems*, 3rd Ed., Springer–Verlag, London, 1995.

[11] M. Jafarian, H. Sandberg, and K. H. Johansson, The interconnection of quadratic droop voltage controllers is a Lotka-Volterra system: Implications for stability analysis, *IEEE Control Sysems Lett.*, 2 (2018) pp. 218–223.

[12] L. Jin, P. Nikiforuk, and M. Gupta, Approximation of discrete-time state-space trajectories using dynamic recurrent neural networks, *IEEE Trans. Automat. Control*, 40 (1995) 1266–1270.

[13] C. Kambhampati, F. Garces, and K. Warwick, Approximation of non-autonomous dynamic systems by continuous time recurrent neural networks, *Proc. International Joint Conference on Neural Networks*, vol. 1, 2000, pp. 64–69.

[14] A. Schäfer and H. Zimmermann, Recurrent neural networks are universal approximators, in *Artificial Neural Networks – ICANN 2006*, S. Kollias, A. Stafylopatis, W. Duch, and E. Oja, Eds., Springer, Berlin, 2006, pp. 632–640.