IEEE Control Systems Letters paper presented at
2023 American Control Conference (ACC)
San Diego, CA, USA. May 31 - June 2, 2023

Data-Driven Control: Theory and Applications

Damoon Soudbakhsh[1], Anuradha M. Annaswamy[2], Yan Wang[3], Steven L. Brunton[4], Joseph Gaudio[5], Heather Hussain[5], Draguna Vrabie[6], Jan Drgona[6], and Dimitar Filev[3]

***Abstract*— The ushering in of the big-data era, ably supported by exponential advances in computation, has provided new impetus to data-driven control in several engineering sectors. The rapid and deep expansion of this topic has precipitated the need for a showcase of the highlights of data-driven approaches. There has been a rich history of contributions from the control systems community in the area of data-driven control. At the same time, there have been several new concepts and research directions that have also been introduced in recent years. Many of these contributions and concepts have started to transition from theory to practical applications. This paper will provide an overview of the historical contributions and highlight recent concepts and research directions.**

I. Introduction

Input-output data and control systems are intricately interwoven. The entire field of control systems is predicated on determining inputs into a system so as to have its outputs directed, altered, or managed in some manner. It is, therefore, not surprising that data-driven approaches based on input-output data are part and parcel of most, if not all of the control systems. Much of the control systems design, analysis, and implementation has revolved around the system response, i.e., the behavior of the system in response to various inputs. Data-driven control methods, therefore, are not just a recent notion but have been studied over the past seven to eight decades. There is no doubt, however, of increased focus on this topic due to explosive growth not in data but also in the area of machine learning. Several other factors are making this topic more attractive of late - digitization has become all-pervasive, supported by the ease of computation and communication, reduction in cost, and advances in actuation technology, facilitating all data-driven decision-making. These tools enable the study of large-scale systems with not only an increase in scale but also an increase in scope. The use of data-driven methods has extended beyond engineering systems into economics, and social science, to mention a few. It is, therefore, time to revisit this topic, which is the focus of this paper.

The topic of data-driven modeling has been studied under various headings, including system realization and system identification. Input-output models and state-space models have been examined since the 1960s, where the efforts therein have sought to capture the dominant characteristics of the system under consideration. The purpose of these models is not just their control, i.e., directing their performance in a desired manner, but also for prediction, monitoring, diagnosis, and optimization. These efforts encompass linear and nonlinear systems in continuous and discrete-time, have included single- and multiple inputs, and have provided both deterministic and stochastic treatments. Recent topics that have occupied more attention have been due to the scale and scope of the systems under consideration. Large-scale systems have necessitated the study of distributed methods and the impact of sparsity in data, in model-topology, and in control. There have been several types of uncertainties occurring in various ways and at different time scales. This in turn has led to an explosive need to understand learning, adaptation, optimization, and their interrelationships. A large number of papers have appeared on these topics and their relation to machine-learning-based methods in general and neural network architectures in particular. We note that all control methods, for the most part, are model-based. These models are either used implicitly or explicitly in the control design. Any effort to provide analytical guarantees necessitates structure. Entities such as system order, system lag, and system parameters enter the picture – these are often in the form of apriori information that needs to be known in order to provide answers of convergence, boundedness, robustness, and optimal behavior, with rigor. We have made an attempt to address these issues in this paper.

In what follows, we undertake the ambitious task of discussing data-driven control methods. Five sections follow. We provide problem statements in §II. We encapsulate developments in the data-driven modeling topic in §III. As the identification of systems and identification of parameters go hand in hand, we discuss the latter in §IV. Data-driven control is then addressed in §V. Several application examples are provided in §VI.

II. Problem Statement

A. Problem Statements

The problem that we address is the control of dynamic systems

* AA was supported in part by Boeing through the Strategic University Initiative and in part by Ford through the Ford-MIT Alliance. DV and JD (PNNL) were supported by DOE contract number DE-AC05-76RL0-1830.

[1]Damoon Soudbakhsh is with the Department of Mechanical Engineering, Temple University, Philadelphia, PA, USA damoon.soudbakhsh@temple.edu

[2]Anuradha M. Annaswamy is with the Department of Mechanical Engineering, Massachusetts Institute of Technology, MA, USA aanna@mit.edu

[3]Yan Wang and Dimitar Filev are with the Ford Motor Company, Dearborn, MI, USA ywang21@ford.com

[4]Steven Brunton is with the Department of Mechanical Engineering, University of Washington, Seatle, WA, USA sbrunton@uw.edu

[5] Joseph E. Gaudio and Heather Hussain are with The Boeing Company, USA {heather.s.hussain, joseph.e.gaudio}@boeing.com

[6]Draguna Vrabie and Jan Drguna are with Pacific Northwest National Lab (PNNL), Richland, WA, USA {draguna.vrabie,jan.drgona}@pnnl.gov

$$\begin{aligned} \dot{x} &= f(x, \theta, u, w, t) \\ y &= g(x, \theta, u, w, t) \end{aligned} \tag{1}$$

where $x(t) \in \mathbb{R}^n$ represents the system state, $y(t) \in \mathbb{R}^p$ represents all measurable system outputs, $u(t) \in \mathbb{R}^m$ represent control inputs that affect the system dynamics of a system, $\theta \in \mathbb{R}^\ell$ represents system parameters. $f(\cdot)$ and $g(\cdot)$ denote nonlinearities, $w(t) \in \mathbb{R}^n$ and $v(t) \in \mathbb{R}^p$ denotes stochastic and deterministic disturbances, respectively, with all quantities specified in continuous time. We also address the control of nonlinear discrete-time dynamics of the form

$$y[k] = f(u[k], k) + v[k], \tag{2}$$

where $u[k]$ is the vector of control inputs, $y[k]$ is the vector of observed outputs, $v[k]$ are due to either stochastic noise and/or deterministic disturbances in the system, and k is the discrete-time index. Whether in continuous- or discrete-time, the problem of interest is the determination of the control input u so that the dynamic system behaves satisfactorily. Most importantly, the focus is on a data-driven approach, i.e., the goal is to be accomplished using only input-output data. Noting that almost all control methods are model-based, data-driven control approaches proposed in the literature can be broadly divided into two categories: i) first identify the model and then design a controller based on the model; ii) directly identify the requisite controller. In what follows, we will discuss both categories i) and ii).

B. Performance Metrics

The goal of data-driven models is to create a model that produces the same response for a class of inputs. The goodness of this fit, i.e., the performance metric, depends on the purpose of the model. One example of the performance metric is given by a loss function defined as

$$\mathcal{J} := \int_{t_0}^{t_f} e_y(t)^T Q_e e_y(t) dt \tag{3}$$

where e_y is a performance error (ex. error in the estimation of the output), $[t_0, t_f]$ is the period of interest, and $Q_e \in \mathbb{R}^{p \times p}$ is a positive definite weighting matrix.

One may be interested in minimizing this loss function over a class of inputs, i.e., for $u(t) \in U$, where $U \subset \mathbb{R}^m$. The simplest example may of the form

$$\min_{u \in U} . \mathcal{J} := \int_{t_0}^{t_f} \left(e_y(t)^T Q_e e_y(t) + u(t)^T R u(t) \right) dt \tag{4}$$

Several other variations exist, where the norm of interest is the peak error over a period or ℓ_1-norm, ℓ_∞, etc. In the context of data-driven control, a similar performance metric as in (4) may be relevant, but with $e_y = y - y_d$, where y_d corresponds to a desired output.

C. Data collection and class of Inputs

An essential part of a data-driven modeling approach is the design of experiments that generate input-output data. This, in turn, raises several questions: i) Is the data collected offline or online? If online, and the system goal involves control, the fundamental conflict between system identification and control has to be addressed. The latter may require any additional inputs used for modeling should be in the form of small perturbations to keep the deviations from the operating conditions small. ii) do we have apriori information about the system? iii) What is the sampling time? What are the constraints on sampling? The number of samples depends on the algorithms used (e.g., impulse response versus deep learning) and the level of complexity. iv) Are there significant nonlinearities? Disturbances? Level of noise? v) What are the considerations for control goals? Specific system attributes such as stability, optimality, accuracy, safety, robustness, uncertainty, and complexity govern the complexity of the model as well as the controller.

For the cases when system modeling can occur prior to determining the controller, the input data needs to be carefully selected. This depends on several factors, such as online or offline modeling, availability of high-fidelity simulators, the duration at which the input can be applied, and computational resources. Some inputs, such as impulse and step functions, require minimal resources but may not be sufficient to accomplish accurate modeling. Sinusoidal inputs, white/colored noise, and pseudo-random noise have low computational time and can be optimally designed. In many cases, we are interested in the algorithms that do not pose limits on the type of input as they would limit the model accuracy and, therefore, its scope of application.

A standard requirement that is often imposed on the input in order to carry out an estimation of the model parameters is known as persistent excitation [1], [2], defined below:

Definition II.1: A piecewise continuous and bounded input $u[k] \in \mathbb{R}^q$ is persistently exciting if $\alpha > 0$ exists s.t.

$$\sum_{k}^{k+n-1} u[k] u[k]^T \geq \alpha I \tag{5}$$

This condition is necessary and sufficient for the uniformly asymptotic stability of the equilibrium state of the system [2]. For stochastic inputs, the PE condition (5) implies that the covariance of the input is positive. For deterministic inputs, this condition is related to the autocovariance of the signal [3].

III. Data-Driven Modeling

In this section, we present several classes of data-driven models that have been derived in the literature. We start with the simplest class, ARMA models, in section III-A, and SIM models, in section III-B, both of which were proposed several years ago. These are followed by more recent methods, which have focused on nonlinear models, which include SINDy, Koopman Operators, and Neural Networks.

A. ARMA models

The simplest form of data-driven modeling of a dynamic system with an input-output pair $\{u[k], y[k]\}$ is an Auto-Regressive Moving-Average (ARMA) one, where the output $y[k+1]$ is a linear combination of past outputs and past inputs. A compact description of an ARMA model takes the

form

$$y[k] = \sum_{i=1}^{n} a_i^* y[k-1] + \sum_{j=1}^{m} b_j^* u[k-j-d] + \sum_{j=1}^{q} c_j^* v[k-j] \tag{6}$$

where a_i^*, b_j^* and c_j^* are ARMA parameters and d is a time-delay. n denotes the order of the system, d is the input-output delay, and m denotes determines the averaging properties of the ARMA model. While a_i^* and b_j^* determine the transient characteristics of the ARMA model, the parameters c_j determine the nature of the noise present in the system (when v is stochastic) and the effect of external disturbances on the output (when v is deterministic). Quite often, n and m are chosen apriori based on prior information about the physics of the problem or based on simple experiments such as evaluating an impulse response of the system or doing a frequency sweep, with the input $u[k]$ chosen as a sinusoid with varying frequencies. Several system identification methods have been developed using the ARMA model description in (6), as the overall system model can be reduced to a simple linear regression model

$$y[k] = \phi[k]^\top \theta^* + \overline{v}[k] \tag{7}$$

where $\phi[k] = [z_y[k-1]^\top, z_u[k-d-1]^\top]^\top$ is a regressor with $z_y[k-1] = [y[k-1], \ldots, y[k-n]]^\top$ and $z_u[k-d-1] = [u[k-1-d], \ldots, u[k-m-d]]^\top$. and $\theta^* = [a_1^*, \ldots, a_n^*, b_1^*, \ldots, b_m^*]^\top$, and $\overline{v}[k]$ represents a filtered noise term. A typical prediction algorithm takes the form

$$\widehat{y}[k] = \phi[k]^\top \widehat{\theta} \tag{8}$$

where $\widehat{\theta}$ is to be determined so as to minimize the prediction error $\widehat{y} - y$ [4]. For the case when u and y are vectors, one obtains the relation

$$Y[k] = \Theta^* \Phi[k] + \mathbf{v} \tag{9}$$

where $\Theta^* \in \mathbb{R}^{n \times m}$ is a matrix of model parameters; a vector of inputs $U[k]$ and $Y[k]$ that determine the regressor $\Phi[k]$ in the presence of noise $\mathbf{v}$. Similar estimation procedures can be used to predict the output and the parameter Θ^* [4], [5]. Yet another attractive property of the ARMA model is its predictor-form [5], which allows the model in (7) to be rewritten in the form

$$y[k] = z^{-d} \left[\overline{\phi}[k]^\top \overline{\theta}^* \right], \tag{10}$$

where z denotes the delay operator with $z^{-1}[x] = x[k-1]$, $\overline{\theta}^*$ depends on θ^*, and $\overline{\phi}[k] = [z_y[k-1]^\top, z_u[k-1]^\top]^\top$. The predictor form is highly useful in designing a control input u and overall controller structure.

Yet another class of ARMA models that have received a lot of attention in data-driven modeling is nonlinear ARMA models [6], which are a direct extension of the ARMA model in (6), and is of the form

$$y[k] = \sum_{i=1}^{n} a_i^* y[k-1] + \sum_{j=1}^{m} b_j^* u[k-j-d] + \sum_{j=1}^{q} c_j^* v[k-j] + \sum_{\ell=1}^{p} d_\ell^* f_\ell \left(z_y[k-1], z_u[k-d-1] \right) \tag{11}$$

where f_ℓ is a nonlinear function of its arguments. It is easy to see that (11) still leads to a linear regression model similar to (10), with the only distinction that the regressor ϕ is replaced by a vector that includes not only past values of the input and output but also nonlinear functions of u and y. This simple linear regression is then leveraged in order to carry out parameter estimation.

1) Data-driven state-space models

Rather than use an input-output representation, a classical approach based on state-space representations can also be adopted [1], [7], [8]

$$x[k+1] = Ax[k] + Bu[k] + w[k] \tag{12}$$

$$y[k] = Cx[k] + Du[k] + v[k] \tag{13}$$

where $x[k] \in \mathbb{R}^n$ represents the system state in discrete-time, $y[k] \in \mathbb{R}^p$ represents discrete-time outputs, and $u(t) \in \mathbb{R}^m$ represent the input variables. $w[k] \in \mathbb{R}^n$ and $v[k] \in \mathbb{R}^p$ are the unmeasurable signals. In deterministic methods (such as [9]), they are ignored. The stochastic inputs$v$ and w are often assumed to be Gaussian, with zero means, and covariance given by

$$\mathbf{E}\left[\begin{bmatrix} w \\ v \end{bmatrix} \begin{bmatrix} w^T & v^T \end{bmatrix} \right] = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \tag{14}$$

and w_k is a noise process made up of Gaussian i.i.d. random variables $N(0, \Sigma)$. Such a state-space description is often utilized to design an optimal control objective of choosing $u[k]$ in (12) so that the cost function

$$J(A,B) \stackrel{\text{def}}{=} \limsup_{T \to \infty} \frac{1}{T} \sum_{i=1}^{T} \left[x[i]^T Q x[i] + u[i]^T R u[i] \right], \tag{15}$$

where $Q = Q^T > 0, R = R^T > 0$, is minimized. Here too, prior information about the system to be modeled is assumed to be present so as to select n and m. With the model structure determined as in (12) and the performance metric as in (15), the next step is to determine the parameters of interest and the control input u.

B. Subspace Identification Methods

Subspace Identification Methods (SIM) can be traced to the introduction of Kalman's minimum realization to identify linear empirical models via analysis of time-series response data [9]–[11]. SIM get their name from the space spanned by rows and columns of matrices constructed from input-output data, which are in turn used to compute linear models. Use of system theory, linear algebra, and statistics concepts are the foundation for SIM, some examples of which include QR decomposition [12], [13], singular value decomposition (SVD) [12], [13], and Least-squares (LS) [11], [12]. With

SIM, the system matrices can be derived once the Kalman states have been calculated from the input-output data. In contrast, in the classical methods, we find the system matrices first in a model as in (12), then estimate the states [11]. Also, SIMs allow for finding the reduced-order models directly from the data as opposed to the classical method of finding a higher-order model and then reducing the order.

Other milestones in SIMs after the deterministic realization of [9] include Stochastic Realization [14], Canonical Variate Analysis (CVA) [15], Multivariable Output-Error State Space (MOESP) [13], and Numerical algorithms for Subspace State Space System Identification (N4SID) [11]. Efficient algorithms to compute the SVD matrices [16] inspired methods such as the Eigensystem Realization Algorithm (ERA) [10] and the Observer Kalman Identification (OKID) technique [17]. Following OKID, Balanced POD (BPOD) was introduced as a tractable model reduction technique that combined the balancing principle of BT with the computational efficiency of POD [18]. In what follows, we provide a brief overview of SIM:

The starting point is determining the order of unknown system (12) subject to (14) as well as the system matrices (A, B, C, D) such that the second-order statistics of the stochastic subsystem is equal to the outputs' stochastic part. As only the observed modes can be recognized, we require the pair $\{A, C\}$ to be observable. Additionally, we need the pair $\{A, [B,\ Q^{1/2}]\}$ to be controllable. Also, we assume that the modes of $\{A, Q^{1/2}]\}$ are stable. The SIM method consists of determining these system parameters and the parameters of the noise inputs in (14) using input-output data.

Let us assume we are using s number of time series measurements. If we use i measurements from time k to $k+i-1$, we can define the following for a sequence of outputs, inputs, and innovations as

$$Y[k,i] \stackrel{\text{def}}{=} \begin{bmatrix} y[k]^T & \cdots & y[k+i-1]^T \end{bmatrix}^T; \quad (16)$$

$$U[k,i] \stackrel{\text{def}}{=} \begin{bmatrix} u[k]^T & \cdots & u[k+i-1]^T \end{bmatrix}^T; \quad (17)$$

$$E[k,i] \stackrel{\text{def}}{=} \begin{bmatrix} \begin{bmatrix} w[k] \\ v[k] \end{bmatrix} & \cdots & \begin{bmatrix} w[k+i-1] \\ v[k+i-1] \end{bmatrix} \end{bmatrix}; \quad (18)$$

The outputs of (12) for these i samples can be written as the function of the current state variables $x[k]$, previous inputs, and innovations. Therefore, we can define an extended state-space model of (12) with a general form

$$Y[k,i] = \mathcal{C}_i x[k] + G_i U[k,i] + F_i E[k,i], \quad (19)$$

with G_i being a Toeplitz matrix of the form

$$G_i \stackrel{\text{def}}{=} \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{-1}B & \cdots & CAB & CB & D \end{bmatrix}, \quad (20)$$

and $\mathcal{C}^i$ is the extended controllability matrix. We define the extended controllability ($\mathcal{C}^i$) and observability ($\mathcal{O}^i$) matrices of system (12) as

$$\mathcal{C}_i = \begin{bmatrix} B & AB & A^2B & \cdots & A^{i-1}B \end{bmatrix} \quad (21)$$

$$\mathcal{O}_i = \begin{bmatrix} C^T & (CA)^T & (CA^2)^T & \cdots & (CA^{i-1})^T \end{bmatrix}^T \quad (22)$$

The subspace identification problem builds on the realization (19) and finding the state variables $x[k]$ as well as the system matrices. SIM algorithms start with forming a generalized Hankel matrix. Assuming that we use all the samples, the matrix will have i number of (block) rows and $i \leq (s+1-2i)$ (block) columns as

$$Y_{(0|i),j} \stackrel{\text{def}}{=} \begin{bmatrix} Y[0,i] & \cdots & Y[j,i+j] \end{bmatrix} \quad (23)$$

where the number of rows, i, relates to the number of modes of the system, and it should be large enough to capture the order of the system. The number of columns, j, depends on the number of samples used to construct the matrix. Let us divide the samples into two parts and call them past (denoted by Y_p) and future (Y_f) and define them as

$$Y_p \stackrel{\text{def}}{=} Y_{(0|i-1),j}, \quad (24)$$

$$Y_f \stackrel{\text{def}}{=} Y_{(i|2i-1),j} \quad (25)$$

For a general SIM algorithm, we need the following definitions of Y_p^+ and Y_f^-, which are defined by moving a row from Y_f to Y_p as follows

$$Y_p^+ \stackrel{\text{def}}{=} Y_{(0|i),j}, \quad (26)$$

$$Y_f^- \stackrel{\text{def}}{=} Y_{(i+1|2i-1),j} \quad (27)$$

Analogous to the outputs, we can define matrices for the inputs $u[i]$. The next step is to define the Hankel matrices to include inputs and outputs as

$$W_p \stackrel{\text{def}}{=} \begin{bmatrix} U_p^T & Y_p^T \end{bmatrix}^T, \quad (28)$$

$$W_p^+ \stackrel{\text{def}}{=} \begin{bmatrix} U_p^{+T} & Y_p^{+T} \end{bmatrix}^T, \quad (29)$$

Let us define a set of state sequence of length j starting at time i as

$$X_i \stackrel{\text{def}}{=} \begin{bmatrix} x[i] & \cdots & x[i+j-1] \end{bmatrix}. \quad (30)$$

We note the following

$$\text{rank}(Y_f/_{U_f} W_p) = n \quad (31)$$

$$\text{row space}(Y_f/_{U_f} W_p) = \text{row space}(X_f) \quad (32)$$

$$\text{column space}(Y_f/_{U_f} W_p) = \text{column space}(\mathcal{O}) \quad (33)$$

Where $(Y_f/_{U_f} W_p)$ denotes oblique projection, which is the projection of the row space of Y_f along the row space of U_f on the row space of W_p [11].

Typically different variations of SIM start with collecting data, forming Hankel matrices (past and shifted), and using SVD, LS, Regression, or another approach to find the matrices. Here we summarize the robust identification process of [11] as follows:

Step 1: Form input and output matrices $Y_f, U_f, W_p, Y_f^-, U_f^-$ and W_p^+

Step 2: Compute the orthogonal projections $Z_i = (Y_f / \begin{bmatrix} W_p \\ U_f \end{bmatrix})$ and $Z_{i+1} = (Y_f^- / \begin{bmatrix} W_p^+ \\ U_f^- \end{bmatrix})$, and oblique projection $V_i = (Y_f /_{U_f} W_p)$.

Step 3: Compute the weighted projection $H_i = W_1 V_i W_2$. The default values for the weights are $W_1 = I$ and $W_2 = \Pi_{U_f^\perp}$, where $\Pi_{U_f^\perp} \stackrel{\text{def}}{=} I - U_f^T (U_f U_f^T)^\dagger U_f$ projecting U_f's row space onto its orthogonal complement.

Step 4: Calculate the SVD of the weighted oblique projection $(V_i \Pi_{U_f^\perp})$ as

$$USV^T = [U_1 \ U2] \begin{bmatrix} S_1 & 0 \\ 0 & 0 \end{bmatrix} [V_1 \ V_2]^T \tag{34}$$

Step 5: Determine the order by selecting $r \leq \text{rank}(V_i \Pi_{U_f^\perp})$, denote $U_r = U_1(:, 1:r)$,$V_r = V_1(:, 1:r)$, and $S_r = S_1(1:r, 1:r)$.

Step 6: Determine the observability matrix $\mathcal{O}^r$ and $\underline{\mathcal{O}^r}$, which is the $\mathcal{O}^r$ without its last p rows ($y[k] \in \mathbb{R}^p$).

$$\mathcal{O}^r = W_1^{-1} U_r S_r^{1/2} \tag{35}$$

Using Steps 1 through 6, which compute various matrices, we now describe the SIM procedure for determining a model as in (12):

Step 7: The underlying dynamics (36) can be written as

$$\begin{bmatrix} {\mathcal{O}_{i-1}^r}^\dagger Z_{i+1} \\ Y_{i|i} \end{bmatrix} = \begin{bmatrix} A \\ C \end{bmatrix} {\mathcal{O}_i^r}^\dagger Z_i + \mathcal{K} U_f + \begin{bmatrix} \widehat{w} \\ \widehat{v} \end{bmatrix} \tag{36}$$

Step 8: Determine system matrices A, C, and $\mathcal{K}$ matrices, where $\mathcal{K}$ is a linear function of B and D from (36) using the least-squares method. $\widehat{w}$ and $\widehat{v}$ are the Kalman filter innovations and will be determined in step 11.

Step 9: Recompute the (reduced) observability matrix using A and C as $\widehat{\mathcal{O}}_i$ and $\widehat{\mathcal{O}}_{i-1}$ for increased compatibility of B and D matrices (next step) with the determined A and C.

Step 10: Determine B and D matrices using the following minimization problem:

$$\begin{aligned}(B, D) = \quad \underset{\widehat{B}, \widehat{D}}{\arg\min} \Big(\| & \begin{bmatrix} \widehat{\mathcal{O}}_{i-1}^\dagger Z_{i+1} \\ Y_{i|i} \end{bmatrix} - \begin{bmatrix} A \\ C \end{bmatrix} \widehat{\mathcal{O}}_i^\dagger Z_i \\ & - \mathcal{K}(\widehat{B}, \widehat{D}) U_f \|_F^2 \Big) \end{aligned} \tag{37}$$

where $\mathcal{K}$, a linear function of B and D, is defined as

$$\mathcal{K} \stackrel{\text{def}}{=} \begin{bmatrix} \begin{bmatrix} B & \mathcal{O}_{i-1}^\dagger G_{i-1} \end{bmatrix} - A \mathcal{O}_i^\dagger G_i \\ \begin{bmatrix} D, 0 \end{bmatrix} - C \mathcal{O}_i^\dagger G_i \end{bmatrix} \tag{38}$$

Step 11: Determine residuals $\widehat{w}$ and $\widehat{v}$ using the determines matrices from (36).

Step 12: Determine the covariance matrices Q, S, and R as

$$\begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} = \Phi([\widehat{w}, \widehat{v}], [\widehat{w}, \widehat{v}]^T). \tag{39}$$

where, $\Phi_{[V_1, V_2]}$ is the covariance between matrices $V_1 \in \mathbb{R}^{m,i}$ and $V_2 \in \mathbb{R}^{n,i}$.

We note that from Steps 7 through 12, a reduced order model (12) has been completely determined using input-output data. The model is of reduced order by using only the dominant singular values (step 6).

Assumption III.1: In the above algorithm, we assumed the following

1) The inputs $u[k]$ are persistently exciting of order $2i$ and they are uncorrelated with $w[k]$ and $v[k]$.
2) There is no limit on the number of measurements.
3) The unmeasured signals $w[k]$ and $v[k]$ are not identical to zero.
4) The user-defined square weighting matrix W_1 $(pi \times pi)$ is full-rank and W_2 $(j \times j)$ is defined such that $\text{rank}(W_p,) = \text{rank}(W_p, W_2)$.

We note that we determine all matrices of (12) (A, B, C, D, Q, R, S) with the Hankel matrix. Several SIM algorithms such as N4SID, CVA, and MOESP can be derived from such cases with different weights (W_1, W_2) or slight modifications of this method [11]. Two special cases of this identification problem resulting in simplifying the matrices and reducing the computational time are 1) the deterministic case, i.e., $w = v = 0$ and ii) the stochastic case without controllable inputs, $u[k = 0]$. We refer the readers to [11], [13], [19] for these popular methods.

C. Dynamic Mode Decomposition

Dynamic Mode Decomposition (DMD) is a method to create linear models by collecting time-series data and processing the matrix using Singular Value Decomposition (SVD) [20]. This method is computationally efficient for separating the (dominant) invariant dynamic modes of a high-dimensional nonlinear system. The resulting models comprise coherent spatial-temporal structures that approximate the system's dominant eigendecomposition [20]–[23]. DMD process starts by taking n snapshots of the system $y[k]$ with a sampling time T_s and creating a matrix $Y[k, n]$ as in (16). We assume that a linear map A connect each snapshot $y[k]$ to the next one $y[k+1]$ as

$$y[k+1] = Ay[k] \tag{40}$$

with an assumption of constant mapping between the snapshots. The overall time-series data can be formulated as a Krylov sequence as [20]

$$Y[k, n] = \begin{bmatrix} y[k] & Ay[k] & \cdots & A^{n-2} y[k] \end{bmatrix} \tag{41}$$

One can then derive the relation

$$AY[k, n] = Y[k+1, n] = Y[k, n] S + r e_n^T \tag{42}$$

where $e_n \in \mathbb{R}^n$ denotes n^{th} unit vector and

$$S = \begin{bmatrix} 0 & \cdots & 0 & a_0 \\ 1 & 0 & \cdots & a_2 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & a_{n-1} \end{bmatrix} \tag{43}$$

Eigenvalues of S represents the dominant eigenvalues of A. In summary, the DMD approach allows the system dynamics

in (40) to be determined, similar to an SVD method, as

$$A = Y[k+1,n]V\Sigma^{-1}U^* \tag{44}$$

where U and V are left and right singular vectors, respectively, and Σ denotes a rectangular diagonal matrix of singular values of $Y[k,n]$ [20]. For very large systems, a reduced size map A_r may be determined as

$$A_r = U^*AU = U^*Y[k+1,n]V\Sigma^{-1}, \tag{45}$$

using the orthonormal property of U. It is possible to reconstruct the high-dimensional map A using the eigen-decomposition of A_r.

D. Sparse Nonlinear Models

Modern neural network architectures have proven quite powerful at modeling arbitrarily complex dynamics, given sufficient training data [24]–[27]. However, for control, it is often helpful to have a model that is interpretable and explainable and has as few degrees of freedom as possible. There are a number of approaches to learning parsimonious models that may be analytically represented in a simple symbolic mathematical expression. We present one such method, known as SINDy (Sparse Identification of Nonlinear Dynamics) [28]–[30], below. The idea of SINDY is to utilize algorithms that are based on sparse regression to learn the simplest model within an admissible library required to describe observational time-series data. The dynamical system in (1) may be approximated in a library of candidate terms. For example, an autonomous system may be approximated as

$$\dot{x} = f(x) \approx \Theta(x)\xi = \sum_{k=1}^{p} \theta_k(x)\xi_k \tag{46}$$

where each term $\theta_k(x)$ is a function that might be present in the dynamics. For many systems, polynomial terms often suffice, although the functions θ_k be any nonlinearities that may be present in the dynamics [31]. The fundamental assumption in SINDy is that the dynamics may be characterized by relatively few active terms, as selected by the sparse vector ξ. There are several sparse optimization algorithms to find ξ that minimizes the following non-convex objective function:

$$\|\dot{x} - \Theta(x)\xi\|_2 + \lambda|\xi|_0 \tag{47}$$

where λ is the sparsity-promoting hyperparameter. Because several leading algorithms are based on a sequentially-thresholded least-squares procedure [28], it is possible to enforce known symmetries and other constraints on the coefficients of ξ, for example, to enforce energy conservation in a fluid [29]. Global stability constraints may also be incorporated into the objective function and optimization, yielding stable-by-construction models [32]. These are some of the attractive properties of SINDy over other methods based on genetic programming and symbolic regression [33]–[36].

SINDy and DMD have both been integrated into a model predictive control (MPC) framework [37] to control complex nonlinear systems from data. The basic SINDy optimization in (46) is modified to include candidate functions $\theta_k(x,u)$ that include both the state x and the actuation input u. It was shown that SINDy is able to learn accurate models from very few examples in time, making it considerably more data efficient than neural networks for MPC, but with comparable or superior control performance, when the systems are symbolically representable. It was also shown that DMD was even more data efficient, although with a degradaded performance. The recommendation was to first use DMD until a SINDy model can be trained. Further improvements can be made by leveraging ensemble methods and active learning [38].

E. Koopman Operators

Koopman operators characterize the nonlinearities of a system via a transformation to an intrinsic coordinate system where nonlinear dynamics can be replaced by linear ones [39]. It has been shown that the DMD's identified modes are the systems Koopman operators [39]–[41]. Using the DMD technique as the computational machinery to identify the Koopman operators of nonlinear systems has opened a new direction in data-driven modeling [42]. An extension of these methods is the variational conformation dynamics (VAC) approach [43]. Another DMD method improvement is extended dynamic mode decomposition (EDMD) [44]–[46]. EDMD can compute eigenfunctions, eigenvalues, and eigenmodes of the Koopman operator [46] and its adjoint, the Perron–Frobenius operator [45]. Some variations of EDMD introduce machine learning and recurrent neural networks as a tool for learning the Koopman invariant subspace [47]–[50]. The Koopman operator is handy in accurate low-dimensional embedding for nonlinear partial differential equations (PDEs) with relatively low computational costs [51], [52]. Koopman modes provide proper, coherent structures for studying the system's dynamics and evolution [48]. A brief exposition of Koopman operators follows.

Let us consider state variables ($x \in \mathcal{M}$) on a smooth n-dimensional manifold $\mathcal{M}$ with dynamics $\dot{x}(t) = f(x(t))$. Then, Koopman operator K is an infinite-dimensional linear operator acting on observable functions $g(f(x))$ that are evolved as $\dot{y} = Ky(t)$, which appear linear. We aim to find finite approximations of the systems' Koopman operators. Unlike the traditional linearization methods that provide a linear approximation of the nonlinear dynamics, Koopman modes (ideally) are the exact representation of the system, and correspond to eigenfunctions.

For example, if we define a library consisting of polynomial and sinusoidal components of observable X as:

$$\Theta(X) = \begin{bmatrix} | & | & | & & | & \\ X & X^2 & X^3 & \cdots & \sin(X) & \cdots \\ | & | & | & & | & \end{bmatrix}, \tag{48}$$

we can find the derivative of $\Theta(X)$ using the chain rule as:

$$\Gamma(X,\dot{X}) = \begin{bmatrix} | & | & & | & \\ \nabla X\cdot\dot{X} & \nabla X^2\cdot\dot{X} & \cdots & \nabla\sin(X)\cdot\dot{X} & \cdots \\ | & | & & | & \end{bmatrix}. \tag{49}$$

Since the Koopman operators are eigenfunctions

$$\Big(\Gamma(X,\dot{X}) - \Lambda\Theta(X)\Big)\,\Xi = 0. \tag{50}$$

for any Ξ, for eigenvalues Λ. Then, we can reconstruct the dynamics using Koopman operators composition with observables as $\mathcal{K}\psi = \psi \circ f$.

F. Neural Network-based Models

It is clear from the preceding sections that linear models, whether ARMA, SIM, or DMD, require a certain linear structure with parameters occurring linearly. Such parametrization is present in nonlinear models as well, as is evident from (11) and (46). In addition, in the latter case, the structure of the nonlinearities is assumed to be known. A natural extension of nonlinear, data-driven models, is the case when nonlinearities may be unknown, which forms the focus of neural-network-based models. The basic assumption, in this case, can be summarized as follows:

Assumption III.2: $\forall X \in \mathcal{D}_\delta$, constants $n_f, \lambda_i, \theta_i, z_i, \epsilon_f$, and a function $\epsilon_0 : \mathbb{R}^n \to \mathbb{R}$ exist such that

$$f(X) = \sum_{i=1}^{n_f} \left(\lambda_i \sigma(\theta_i^T \bar{X}) + w_i\right) + \epsilon_0(X) \tag{51}$$

$$|\epsilon_0(X)| \leq \epsilon_f \tag{52}$$

where $\bar{X} = [X^T, 1]^T$ and $\sigma(\cdot)$ is an activation function.
It can be seen that this allows the extension of both (11) and (46) to the case when the nonlinearities are unknown but may be approximated as in (51) where parameters occur linearly and nonlinearly [53]. While the approximation in (51) includes only one hidden layer, a deep neural network incorporates several additional layers, with the number of layers enabling approximations of more complex nonlinearities [54]. Estimation of the weights, however, is difficult to ensure globally due to underlying nonconvexities. The use of the well-known backpropagation algorithm [55] and variations thereof have been shown to be successful in several cases.

IV. Parameter Identification

In this section, we outline three different methods that have been proposed to identify parameters, which include gradient methods, recursive least squares methods, and higher-order methods based on momentum and acceleration.

A. Gradient-based

We note that ARMA models in (6) and Nonlinear ARMA models in (10), are both in the form of linear regression. Several other problems including estimation and pattern recognition problems in discrete-time. and problems in continuous-time plants can all be cast in such a form [56].

With an output prediction as in (8), where $\widehat{y}(t) \in \mathbb{R}$ is the estimated output and the unknown parameter is estimated as $\theta[k] \in \mathbb{R}^N$ we obtain two types of errors, a performance error $e_y[k]$ and a learning error $\widetilde{\theta}[k]$[1]

$$e_y = \widehat{y} - y, \qquad \widetilde{\theta} = \theta - \theta^* \tag{53}$$

where the former can be measured but the latter is unknown though adjustable. From (7) and the estimator, it is easy to see that e_y and $\widetilde{\theta}$ are related using a simple regression relation

$$e_y[k] = \widetilde{\theta}^T[k]\phi[k]. \tag{54}$$

A common approach for adjusting the estimate $\theta[k]$ at each time k is to determine a rule using all available measurements such that $e_y[k]$ converges towards zero. To do so, a squared loss function

$$L_k(\theta[k]) = \frac{1}{2}e_{y,k}^2 = \frac{1}{2}\widetilde{\theta}^T[k]\phi[k]\phi[k]^T\widetilde{\theta}[k], \tag{55}$$

whose gradient can be determined at each iteration k as $\nabla L_k(\theta_k) = \phi_k e_{y,k}$. It is well known that this gradient can be used to determine a recursive algorithm for adjusting the parameter estimate θ (see for example, [5], ch. 3) as

$$\theta[k+1] = \theta[k] - \gamma \nabla \bar{f}_k(\theta[k]), \quad 0 < \gamma < 2, \tag{56}$$

where $\bar{f}_k(\cdot)$ corresponds to a normalized loss function defined as

$$\bar{f}_k(\theta[k]) = \frac{L_k(\theta[k])}{\mathcal{N}_k}, \tag{57}$$

and $\mathcal{N}_k = 1 + \|\phi[k]\|^2$ is a normalization signal employed to ensure boundedness of signals for any arbitrary regressor $\phi[k]$. Such a gradient-descent approach has found widespread applications in estimation, control, signal processing, and ML. A necessary and sufficient condition for the parameter estimate $\theta[k]$ to converge to θ^* asymptotically is persistent excitation of $\phi[k]$ (see Definition II.1).

B. Recursive least squares based algorithm

Rather than use the gradient approach as in (56), one can use an approach based on recursive least squares (RLS). The idea is to replace the constant scalar step size γ with a matrix Γ_k that is time-varying. This results in a significantly faster convergence and takes the form [5]

$$\theta[k] = \theta[k-1] + \frac{\Gamma_{k-1}\phi[k-1]e[k-1]}{1 + \phi[k-1]^\top \Gamma_{k-2}\phi[k-1]}, \tag{58}$$

$$\Gamma_{k-1} = \Gamma_{k-2} - \frac{\Gamma_{k-2}\phi[k-1]\phi[k-1]^\top \Gamma_{k-2}}{1 + \phi[k-1]^\top \Gamma_{k-2}\phi[k-1]}, \tag{59}$$

While the RLS algorithm guarantees an exponential convergence of $\theta[k]$ to its true value, it also can lead to a rapid convergence of Γ_k to zero if the data contains regressors that are persistently exciting. The introduction of a forgetting factor avoids this problem [57] through the introduction of a parameter $\alpha < 1$:

$$\theta_k = \theta_{k-1} + \frac{\Gamma_{k-1}\phi_{k-1}e_{k-1}}{\alpha + \phi_{k-1}^\top \Gamma_{k-2}\phi_{k-1}}, \tag{60}$$

$$\Gamma_{k-1} = \frac{1}{\alpha}\left[\Gamma_{k-2} - \frac{\Gamma_{k-2}\phi_{k-1}\phi_{k-1}^\top \Gamma_{k-2}}{\alpha + \phi_{k-1}^\top \Gamma_{k-2}\phi_{k-1}}\right], \tag{61}$$

We refer the reader to [5], [57], [58] for proofs of convergence for RLS and related algorithms.

[1] In what follows, we suppress the argument $[k]$ unless needed for emphasis.

C. Higher-order tuner

Instead of a gradient-based update as in (56), our goal is to derive a second-order tuner that is Nesterov-type [59] and includes terms that incorporate momentum and acceleration. For ease of exposition, we assume that the noise term is zero. We consider the objective function in (62):

$$f_k(\theta[k-1]) = \frac{L_k}{\mathcal{N}_{k-1}} + \frac{\mu}{2}\|\theta[k-1]-\theta_0\|^2, \qquad \mu > 0. \quad (62)$$

Following an update in the parameter as in (56), we note that the output error in (53) can be updated as well, leading to a-priori and a-posteriori errors

$$e_y[k] = \phi^T[k-1]\theta[k-1] - \phi[k-1]^T\theta^* \quad (63)$$

$$e_y^{ap}[k] = \phi[k-1]^T\theta[k] - \phi[k-1]^T\theta^* \quad (64)$$

This in turn leads to an a-priori gradient for f_k as

$$\nabla f_k(\theta_{k-1}) = \frac{\nabla_{\theta_{k-1}} L_k}{\mathcal{N}_{k-1}} + \mu(\theta_{k-1} - \theta_0) \quad (65)$$

and an a-posteriori gradient

$$\nabla f_k(\theta_k) = \frac{\nabla_{\theta_k} L_k}{\mathcal{N}_{k-1}} + \mu(\theta_k - \theta_0) \quad (66)$$

where $\nabla_{\theta_k} L_k$ is defined as

$$\nabla_{\theta_k}\left(\frac{1}{2}\left(e_y^{ap}[k]\right)^2\right) = \phi[k-1]\left(\phi[k-1]^T\theta[k] - \phi^T[k-1]\theta^*\right) \quad (67)$$

We now propose the high-order tuner (HT):

$$\vartheta_k = \vartheta_{k-1} - \gamma\nabla f_k(\theta_k), \quad (68)$$

$$\bar{\theta}_{k-1} = \theta_{k-1} - \gamma\beta\nabla f_k(\theta_{k-1}), \quad (69)$$

$$\theta_k = \bar{\theta}_{k-1} - \beta(\bar{\theta}_{k-1} - \vartheta_{k-1}). \quad (70)$$

That is, the update from θ_{k-1} to θ_k occurs using a second-order tuner as in (68)-(70), that includes both acceleration based momentum based components. The use of the a-posteriori gradient in (68) provides an acceleration component while the rotation of the update as in (69) can be viewed as a momentum term. It should be noted however that the presence of the time-varying regressor ϕ_k in the overall problem causes this HT to be fundamentally different from Nesterov's algorithm This difference is key to establishing its stability, which cannot be ensured in Nesterov's algorithm when the regressor ϕ_k varies with k.

Theorem IV.1: For the linear regression error model in (53) with loss in (55), the HT in (68)-(70) with its hyperparameters chosen as $\mu = 0$, $0 < \beta < 1$, and $0 < \gamma \leq \frac{\beta(2-\beta)}{16+\beta^2}$ leads to the following Lyapunov function

$$V_k = \frac{1}{\gamma}\|\vartheta_k - \theta^*\|^2 + \frac{1}{\gamma}\|\theta_k - \vartheta_k\|^2, \quad (71)$$

with $\Delta V_k \leq -\frac{L_k}{\mathcal{N}_{k-1}} \leq 0$. In addition to $V \in \ell_\infty$, it can be shown that $\lim_{k\to\infty} L_k = 0$.

In addition to the above stability properties, the HT has another attractive property, of accelerated convergence rate when the regressors are constant and no noise is present. It can be shown that HT in (68)-(70) is equivalent to Nesterov's algorithm when ϕ_k is a constant [60]. We now state the acceleration property of the HT in Theorem IV.2. The reader is referred to [61], [62] for its proof. The algorithm starts with the Nesterov equation

$$\begin{aligned} \theta_k &= \nu_{k-1} - \bar{\alpha}\nabla f(\nu_{k-1}), \\ \nu_k &= (1+\bar{\beta})\theta_k - \bar{\beta}\theta_k, \end{aligned} \quad (72)$$

The following choice of hyperparameters in (72) is made:

$$\begin{aligned} \theta_0 = \nu_0 \qquad \mu &= \epsilon/\Psi \qquad \bar{L} = 1+\mu \\ \bar{\alpha} = 1/\bar{L} \qquad \kappa &= \bar{L}/\mu \;\; \bar{\beta} = (\sqrt{\kappa}-1)/(\sqrt{\kappa}+1) \end{aligned} \quad (73)$$

where $\Psi \geq \max\{1, \|\theta_0 - \theta^*\|^2\}$ for unnormalized loss functions and $\Psi \geq \max\{1, \mathcal{N}\|\theta_0 - \theta^*\|^2\}$ for normalized loss functions.

Theorem IV.2: For a $\bar{L}$-smooth and μ-strongly convex function f, the iterates $\{\theta_k\}_{k=0}^{\infty}$ generated by (72) with with hyperparameters as in (73) satisfy

$$f(\theta_k) - f(\theta^*) \leq \frac{\bar{L}+\mu}{2}\|\theta_0 - \theta^*\|^2 \exp\left(-\frac{k}{\sqrt{\kappa}}\right).$$

The results of Theorem IV.2 can be utilized to conclude that an ϵ-optimal convergence can be guaranteed for a number of iterations that is $\mathcal{O}(1/\sqrt{\epsilon}\cdot\log(1/\epsilon))$ with the HT in (68)-(70) both for the unnormalized and the normalized loss functions. Further details of the high-order tuner can be found in [60], [63].

V. Direct Data-Driven Control

Once the model structure is determined, and its parameters are identified, all using input-output data, one can employ one of several methods in the broad and deep canon of control systems. We have highlighted three methods in this section, one based on a static model in §V-A, one that employs differentiable predictive control with a goal of optimality in §V-B, and the third based on the adaptation of control parameters with the goal of real-time control and learning (§V-E. We outline the difficulty of using neural network-based control methods in the last subsection.

A. Static Models

In this section, we present a class of static models that often occur in several slow processes or have dynamics that can be ignored with respect to the sampling rate, and are nonlinear but smooth [64], [65]. As shown in Fig. 1, the desired output y_d is achieved by iterative processes of optimizing the inputs u to the plant based on a data-driven model, represented by the inputs u and outputs y. We briefly describe the method below.

The input $u[k]$ and output $y[k]$ are assumed to be related using a static nonlinearity

$$y[k] = S(u[k]) \quad (74)$$

where $S(u[k])$ is a nonlinear and smooth function. The objective of the closed-loop system is to find the input vector $u[k]$ that minimizes the error,

$$\|e[k]\|_2 = \|y[k] - y_d[k]\|_2 \quad (75)$$

where, $y_d[k]$ is the desired output vector at time k.

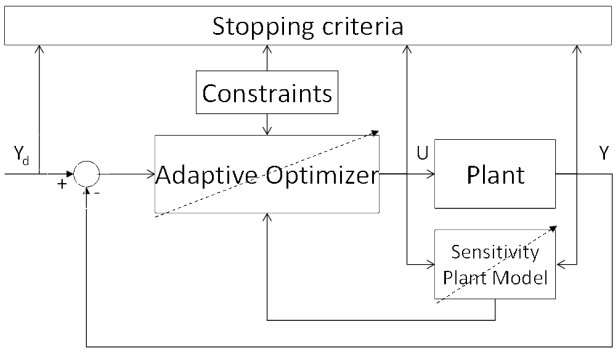


Fig. 1: Block Diagram of the Algorithm

We start with the linearized model

$$\Delta y[k] = \mathbb{J}[k]\Delta u[k] \tag{76}$$

where,

$$\begin{aligned} \Delta u[k] &= u[k] - u[k-1] \quad , \qquad \Delta u[k] \in R^r \\ \Delta y[k] &= y[k] - y[k-1] \quad , \qquad \Delta y[k] \in R^q \end{aligned} \tag{77}$$

It follows from (77) that for a known Jacobian $\mathbb{J}[k]$, the optimal input update minimizing the cost in (75) can be calculated from the pseudo-inverse $\mathbb{J}^\dagger[k]$ as [66]

$$u[k+1] = u[k] + \mathbb{J}^\dagger[k](y_d[k] - y[k]) \tag{78}$$

To avoid singularities, we regularize $\mathbb{J}^\dagger[k]$ using $H[k]$ as

$$\begin{aligned} H[k] &= \mathbb{J}^T[k](\mathbb{J}[k]\mathbb{J}^T[k] + \rho I)^{-1} \quad \text{for} \quad r \geq q \\ H[k] &= (\mathbb{J}^T[k]\mathbb{J}[k] + \rho I)^{-1}\mathbb{J}^T[k] \quad \text{for} \quad r \leq q \end{aligned} \tag{79}$$

where ρ is a small positive constant and I is the identity matrix.

The system's estimated Jacobian can be used as a feedback control law of the form:

$$u[k+1] = u[k] + \widehat{H}[k]G(y_d - y[k]) \tag{80}$$

where G is a diagonal matrix representing control gains with its diagonal elements $g_i \in (0, 2)$. For $r \geq q$, $\widehat{H}[k]$ is defined as

$$\widehat{H}[k] = \widehat{\mathbb{J}}^T[k]G(\widehat{\mathbb{J}}[k]\widehat{\mathbb{J}}^T[k] + \rho I)^{-1} \tag{81}$$

Further, the Jacobian can be estimated using RLS [4] as

$$\widehat{\mathbb{J}}_j^T[k] = \widehat{\mathbb{J}}_j^T[k-1] + \frac{P_j[k-1]\Delta u[k](\Delta y[k] - \widehat{\mathbb{J}}_j[k-1]\Delta u[k])}{R_j + \Delta u^T[k]P_j[k-1]\Delta u[k]} \tag{82}$$

$$P_j[k] = P_j[k-1] - \frac{P_j[k-1]\Delta \mathrm{u}[k]\Delta u^T[k]P_j[k-1]}{R_j + \Delta u^T[k]P_j[k-1]\Delta u[k]} + Q_j, \tag{83}$$

where matrix Q_j represents the drift factor used to force the system when it is not excited and ensures that the covariance matrix $P_j[k]$, does not increase exponentially. R_j denotes a forgetting factor that also controls the evolution of the covariance matrix. When noise is present, R_j is often chosen as the noise covariance [67], [68].

In order to accommodate any input constraints that may be present, a constraint optimization problem can be stated as

$$u^*[k] = \underset{u[k]}{\arg\min}(\|y_d - \widehat{y}[k]\|^2 + \gamma\|u[k] - u[k-1]\|^2) \tag{84}$$

$$\text{s.t.} \qquad \widehat{y}[k] = y[k-1] + \widehat{\mathbb{J}}[k](u[k] - u[k-1])$$

The update in (84) can be done using the readily available QP solvers, e.g., the FMINCON and LSQLIN functions in the Matlab Optimization Toolbox.

In the following, we analyze the convergence of the algorithm based on input update (84). By application of expression (80) to the linearized model (77), we obtain a first-order difference equation that covers the dynamics of the updating feedback loop:

$$\begin{aligned} y[k] &= y[k-1] + \mathbb{J}[k]H[k]G(y_d[k-1] - y[k-1]) \\ &= A_c[k-1]y[k-1] + \mathbb{J}[k]H[k]Gy_d[k-1], \end{aligned} \tag{85}$$

where the closed loop matrix A_c is defined as

$$A_c[k] = I - \mathbb{J}[k+1]\widehat{\mathbb{J}}^T[k]G(\widehat{\mathbb{J}}[k]\widehat{\mathbb{J}}^T[k] + \rho I)^{-1}. \tag{86}$$

As long as the estimated model $\widehat{\mathbb{J}}[k]$ approaches the true Jacobian $\mathbb{J}[k+1]$, i.e.

$$\mathbb{J}[k+1] \approx \widehat{\mathbb{J}}[k], \tag{87}$$

the matrix of closed loop system $A_c[k]$ can satisfy the inequality

$$A_c^T[k]IA_c[k] < I \tag{88}$$

for sufficiently low values of ρ and diagonal entries of G. This in turn guarantees stability and for $y[k]$ to track y_d [67].

B. Differential Predictive Control for Dynamical Systems

1) Differentiable Programming

Differentiable programming (DP) [69] is a programming paradigm that leverages automatic differentiation (AD) to compute the gradients of complex computer programs. The main benefit of DP is that gradient-based optimization methods can be used to estimate the unknown parameters of the computer program. From this perspective, deep learning (DL) based on backward AD through the network architecture can be seen as a subset of DP. Therefore DP allows for a systematic combination of machine learning (ML) and physics-based models in a unifying framework. For instance, authors in [70] show how to differentiate convex optimization problems and use them as layers in deep neural network architectures. Examples of DP in control include differentiable MPC [71], [72] proposed as safe imitation learning methods to learn system dynamics models from expert trajectories. These methods differentiate analytically constructed Kharush-Kuhn-Tucker (KKT) optimality conditions to obtain the solution. Others have computed the gradients via backpropagation through the unrolled computational graph of the MPC problem [73]. Authors in [74], [75] have combined MPC in the inner loop with reinforcement learning (RL) at the outer loop in the so-called RL-MPC method. This approach uses RL to tune the parameter of the differentiable MPC problem to provide adaptation in real-time control with performance and safety guarantees.

2) Parametric Optimal Control Problem

Let's consider the following optimal control problem:

$$\min_{\theta} \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k), \tag{89a}$$

$$\text{s.t. } \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k),\ k \in \mathbb{N}_0^{N-1}, \tag{89b}$$

$$h(\mathbf{x}_k) \leq \mathbf{0}, \tag{89c}$$

$$g(\mathbf{u}_k) \leq \mathbf{0}, \tag{89d}$$

$$[\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}] = \boldsymbol{\pi}_\theta(\mathbf{x}_0, \boldsymbol{\xi}), \tag{89e}$$

$$\mathbf{x}_0 \in \mathcal{X}_{init}. \tag{89f}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$, $\mathbf{u}_k \in \mathbb{R}^{n_u}$ are system states and control inputs, respectively. The prediction horizon is defined by N, and the explicit predictive control policy (89e) is parametrized by parameters θ that need to be optimized. The initial conditions $\mathbf{x}_0$ belong to the set of feasible initial states $\mathcal{X}_{init}$, and $\boldsymbol{\xi}$ represents additional control parameters such as time-varying reference trajectories or boundary conditions.

Assumption V.1: The system dynamics model (89b) is known and controllable.

Assumption V.2: The control objective $\ell(\mathbf{x}_k, \mathbf{u}_k)$ and constraints $h(\mathbf{x}_k)$, and $g(\mathbf{u}_k)$ are at least once differentiable.

3) Differentiable Predictive Control Method

Differentiable Predictive Control (DPC) [76] is a recent method that uses automatic differentiation (AD) to compute the gradients of the parametric optimal control problem (89). This allows DPC to learn explicit predictive control policies from sampled control parameters $\mathbf{x}_0$, and $\boldsymbol{\xi}$. In DPC the problem (89) is implemented as a differentiable program that can be conceptually represented in Fig. 2. First, we sample

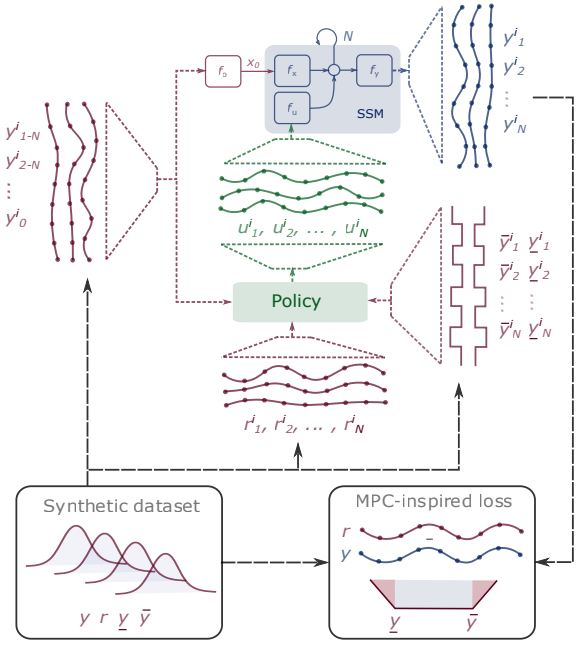


Fig. 2: Differentiable predictive control methodology.

the control parameters to generate a synthetic training dataset to train the control policy based on a physics-based or data-driven state space model (SSM) (89b). The control policy together with the SSM form a differentiable discrete time closed-loop system,

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \pi_\theta(\mathbf{x}_k, \boldsymbol{\xi})). \tag{90}$$

whose response to control parameters can be simulated over N-step ahead prediction horizon. We can use the closed-loop system (90) to generate trajectories as a simulated response to control parameters $\mathbf{x}_0$ and $\boldsymbol{\xi}$) over given N-step ahead prediction horizon. This is equivalent to a single shooting method in online model predictive control (MPC) [77].

The difference between DPC and MPC based on online optimization is that the MPC solution computes the sequence of optimal control actions, while DPC is optimized over explicit control policy parameters (89e). In principle, the control policy can be parametrized as any parametrized differentiable function. In our case, we assume the policy to be represented by a fully connected deep neural network defined as:

$$\boldsymbol{\pi}_\theta(\mathbf{x}_0, \boldsymbol{\xi}) = \mathbf{H}_L \mathbf{z}_L + \mathbf{b}_L \tag{91a}$$

$$\mathbf{z}_l = \boldsymbol{\sigma}(\mathbf{H}_{l-1}\mathbf{z}_{l-1} + \mathbf{b}_{l-1}) \tag{91b}$$

$$\mathbf{z}_0 = [\mathbf{x}_0, \boldsymbol{\xi}] \tag{91c}$$

with $\mathbf{z}_i$ representing hidden states, $\mathbf{H}_i$, and $\mathbf{b}_i$ defining weights and biases, and i indexing the network layers. While $\boldsymbol{\sigma} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ represents element-wise application of an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, e.g., sigmodid or ReLU.

In DPC, we simulate the control policy in conjunction with the system model to generate the closed-loop control trajectories used for evaluation of constraints and objective function of the parametric optimal control problem (89). To do so we consider the Lagrangian loss function given as follows,

$$\mathcal{L}_{\text{DPC}} = \frac{1}{mN} \sum_{i=1}^{m} \Big(\sum_{k=0}^{N-1} \big(Q_l \ell(\mathbf{x}_k^i, \mathbf{u}_k^i) + Q_h p_h(h(\mathbf{x}_k^i)) + Q_g p_g(g(\mathbf{u}_k^i))\big) + Q_N p_N(\mathbf{x}_N^i) \Big), \tag{92}$$

with m being the number of samples in the training dataset. Here Q_l, Q_h, Q_g, and Q_N represent weight factors for individual terms, where $\ell(\mathbf{x}_k^i, \mathbf{u}_k^i)$ is the control objective, and $p_h(h(\mathbf{x}_k^i))$, $p_g(g(\mathbf{u}_k^i))$, and $p_N(\mathbf{x}_N^i)$ are penalties for state, input, and terminal set constraints, respectively. The differentiability of the closed-loop system (90) and the loss function (92) allows us to use the backpropagation algorithm to compute the policy gradients for subsequent control policy optimization via stochastic gradient descent.

Remark V.3: The terminal penalty term $p_N(\mathbf{x}_N^i)$ in the DPC loss function (92) is is a widely used method in MPC literature [78]–[80] to guarantee stability and recursive feasibility of the closed-loop system.

The DPC policy optimization algorithm (Algorithm 1) has been extended to handle stochastic parametric optimal control problems [81], and integrated with neural Lyapunov functions [82] and control barrier fuctions [83] for stability and constraints satisfaction guarantees. and has been successfully applied to applications in building control [84], dynamic economic dispatch in power systems [85], with

demonstrated deployment on a microcontroller governing a laboratory hardware device [76].

C. Adaptive Control and Machine Learning

Adaptive control and machine learning are two fields that leverage data to improve the performance of algorithms, where the data-driven learning process is often encoded by parameter updates. Parameter update algorithms in both fields have evolved largely in parallel over the past multiple decades [86], [87]. Adaptive control has primarily focused on real-time engineering systems (e.g., aerospace, robotics) for problems of tracking and estimation in the presence of uncertainties. Machine learning, on the other hand, has largely focused on deploying learned systems where the parameter estimation has already occurred offline (pre-deployment). This section presents common aspects between adaptive control and machine learning as viewed through the lens of common parameter updates for learning in dynamical systems.

Large classes of real-time learning-based algorithms consider parametric dynamical system models in the form of (1). In the control problem, the goal is to design the input u, such that an output tracking error $e_y = y - y_d$ converges to zero, where $y_d(t) \in \mathbb{R}^p$ represents the desired tracking outputs, and such that other signals in the closed loop system remain bounded. The control input is often of the form

$$u(t) = g_1(e_y(t), \phi(t), \theta(t)), \tag{93}$$

where the regressor $\phi \in \mathbb{R}^M$ represents measured data and computed variables, and $\theta \in \mathbb{R}^N$ is an estimate of θ^*. In both control and estimation problems, the process of learning is encapsulated in the parameter estimate, θ, which is often adjusted using an iterative approach. In this section, we consider a loss-based method in the iterative approach, where available data encoded in the regressor ϕ and output error e_y are employed such that a user-defined scalar-valued performance-based loss $L(\theta, \theta^*, e_y, \phi)$ is minimized via the adjustment of θ. Both continuous time and discrete index representations have been considered in the adjustment of the parameter. In the interest of space, the discrete index representation is presented in this section.

Algorithm 1 DPC policy optimization

1: **Input** known system dynamics model $f(.)$
2: **Input** control objective $\ell(\mathbf{x}, \mathbf{u})$ and constraints $h(\mathbf{x})$, and $g(\mathbf{u})$
3: **Construct** closed-loop system dynamics model (90) and DPC loss function (92)
4: **Input** optimizer $\mathbb{O}$
5: **for** epochs=1:$N_{\texttt{epoch}}$ **do**
6: **Sample** initial conditions $\mathbf{x}_0$ and parameters $\boldsymbol{\xi}$
7: **Evaluate** forward simulation of the closed-loop system (90)
8: **Differentiate** DPC loss (92) w.r.t. policy parameters to obtain gradients and $\nabla_\theta \mathcal{L}_{\texttt{DPC}}$
9: **Update** control parameters $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_{\texttt{DPC}}$
10: **end for**
11: **Return**: trained explicit control policy $\boldsymbol{\pi}_\theta$

The parameter update law is often considered to be of the form [86]

$$\theta_{k+1} = \theta_k + \beta_k(\theta_k - \theta_{k-1}) - \Gamma_k \nabla_\theta L_a(\theta_k, \theta^*, e_y, \phi) \tag{94}$$

where $\beta_k \in \mathbb{R}_{\geq 0}$ is referred to as a momentum parameter, $\Gamma_k \in \mathbb{R}^{N \times N}$ is a full-matrix learning rate, and L_a is the augmented loss function

$$L_a(\theta_k, \theta^*, e_y, \phi) = L(\theta_k, \theta^*, e_y, \phi) + \sigma \mathcal{R}(\theta_k, e_y), \tag{95}$$

with a user-designed scalar-valued regularization function $\mathcal{R}$, and regularization weighting $\sigma \geq 0$. In the remainder of this section, each term in (94), (95) is discussed under the contexts of both adaptive control and machine learning.

While the performance-based loss L often contains explicit dependence on the unknown parameter θ^*, the regularization function $\mathcal{R}$ is often designed in a manner that does not contain such explicit unknown parameter dependence to promote robustness to perturbations such as unmeasurable disturbances/noise and unmodeled dynamics. Continuous time adaptive updates have considered regularization with a particular analysis of robustness to bounded perturbations with the σ-modification $\nabla_\theta \mathcal{R} = \theta$ [88], and the e-modification $\nabla_\theta \mathcal{R} = \|e_y\|\theta$ [89]. In machine learning, regularization is often considered in the context of robustness to hold out data during the parameter update tuning [90]–[92]. An ℓ_p regularization of the form $\mathcal{R} = (1/p)\|\theta\|_p^p$ is often considered with $p = 1$ to promote sparsity in the parameter space. It can be noted that $p = 2$ corresponds with the aforementioned σ-modification in adaptive control.

Adaptive control has commonly considered full-matrix learning rates, Γ_k, often based on recursive least squares formulations (c.f. [58], [93] and references therein). A full matrix learning rate enables online rotations of the gradient-based learning direction using available measured data. This has connections to Hessian-based learning akin to Newton's method, where data-driven normalizations are employed for stable learning in dynamical systems [93]. The field of optimization for machine learning commonly considers diagonal learning rates, Γ_k, due to commonly considered high-dimensional representations. The diagonal structure of Γ_k provides for a coordinate-wise re-scaling to accommodate sparse data and gradients of differing magnitudes [94]–[96].

A higher-order tuner as described in Section IV-C has also been shown to have a strong potential for fast convergence, both in control and in identification. In particular, momentum-based update laws, which consider a linear combination of current and previous iterates in the parameter update, have seen widespread use in machine learning to provide for faster rates of convergence in training [97], [98], and are shown in (94) with weighting parameter β_k. Continuous time adaptive laws with higher-order learning have appeared in [99] and have been considered with a variational approach introduced in the machine learning community in [100] with corresponding certifications of faster rates of convergence for classes of convex problems in [101]. We should note

that when the excitation level is not sufficient or it is not persistently exciting, the parameters do not converge to the true values, i.e., leads to imperfect learning [56].

D. Reinforcement Learning

The concepts of optimality and dynamic programming have significant contributions to Reinforcement Learning (RL) [102], [103]. In RL, we seek a control policy $u[k] = \pi(x_k)$ that minimizes a cost function of the form

$$J(\pi(x_0)) = \lim_{k\to\infty)} \frac{1}{k}\sum_{i=0}^{k} c(x_i, \pi(x_i)), \qquad x_0 \in X \quad (96)$$

The control policy π maps $x \in X$ to $u \in U(x)$, with X and U being the desired sets. The underlying assumption here is that there exist policies such that

$$0 \leq c(x_k, U[K]) \leq \infty, \forall x_k \in X, \quad u_k \in U(x_k) \quad (97)$$

The *optimal policy* u^* results in an *optimal value* J^*, and it should satisfy the well-known Hamilton-Jacobi-Bellman equation []. The problem can be solved using dynamic programming that relies on the principle of optimality to optimize the current assuming an optimal cost-to-go after the current step $J^*(x[k+1)$ in the form of:

$$u^*(x) = \arg\min_{u\in U(x)}\{c(x,u) + J^*(x[k+1)]\} \quad (98)$$

However, the solution to this optimal problem, except for special cases, is very computationally expensive and increases with the dimensions of the problem ("curse of dimensionality"). Therefore, newer methods aim at approximate solutions (hence, the name Approximate Dynamic Programming). Several methods exist for solving this problem, including Q-learning, value-iteration, and policy iteration. A successful approach to solve this problem has been offered by using deep neural networks, where, for example, a policy $\widehat{Q}_{\theta_\pi}(x[k], u[k] = g(x[k], u[k], \theta_\pi)$ can be learned using a neural network to find the optimal policy.

E. Neural Network-Based Control

One of the common methods for determining estimates of the weights in the neural-network representation in (51) is the use of a gradient, similar to (55) for a suitably defined loss function. These loss functions are often in the form of a reward function that is to be optimized through the adjustment of the weights θ_i, λ_i, and w_i. Unlike the simple convex (quadratic) representations in (54), the loss function in a neural network is non-convex due to the presence of activation functions $\sigma(\cdot)$. Suppose one is concerned with determining a control policy in the form of $u = \pi_\theta(x)$ where the nonlinearity $\pi_\theta(x)$ is represented as in (51). The problem is to determine an appropriate loss function that will train the policy appropriately while ensuring that the closed-loop system that consists of the nonlinear plant and the policy remains stable. Often, these two goals are at odds with each other for arbitrary nonlinear systems and require several conditions to be satisfied [56], [104]–[106].

VI. Application Examples

Here, we show a few applications of the presented methods. The first one focuses on flight control (§VI-A), then we show a couple of applications in the automotive industry (sectionss VI-B.1-VI-B.2) and maneuvers optimization of a sailboat (§VI-B.3), followed by a review of a series of applications in fluid dynamics in §VI-C, and finally a data-driven control of buildings in §VI-D.

A. Data-driven flight control

This section details an academic planar rendezvous problem as a prototypical example in flight control [107], to elucidate many challenges which must be overcome in data-driven methods for control. In this example, a vehicle is commanded to move to a time-varying rendezvous location. The rendezvous location is assumed to be generated by model double integrator dynamics of the form

$$\dot{x}_m(t) = v_m(t), \quad (99)$$
$$\dot{v}_m(t) = a_m(t), \quad (100)$$

with model position $x_m \in \mathbb{R}^2$, velocity $v_m \in \mathbb{R}^2$, and acceleration $a_m \in \mathbb{R}^2$. In this section, the model point acceleration, a_m, is assumed to be zero (i.e., constant velocity/non-accelerating model point). The acceleration-commanded vehicle is given by

$$\dot{x}(t) = v(t), \quad (101)$$
$$\dot{v}(t) = a(t), \quad (102)$$

with position $x \in \mathbb{R}^2$, velocity $v \in \mathbb{R}^2$, and controlled acceleration $a \in \mathbb{R}^2$. It is assumed that the vehicle is only able to command accelerations normal to its velocity vector, must complete the rendezvous in finite time (as defined by a small range to the model point within a prescribed threshold), and should optimize a performance functional of the form

$$\mathcal{J} := c_1 \left[\|x(t) - x_m(t)\|\right]|_{t=t_f} + \int_t^{t_f} L(x(\tau), v(\tau), a(\tau), x_m(\tau), v_m(\tau), \tau)d\tau \quad (103)$$

where $c_1 > 0$ is a large constant to penalize a rendezvous error at a final time t_f (may be fixed or free), and L is a running cost. The optimization formulation may additionally include constraints on the vehicle state (state path constraints) and control limits (acceleration limits).

This outline of an academic planar two-point boundary value problem (TPBVP) demonstrates multiple challenges which must be overcome for data-driven methods in the design of the vehicle control (acceleration) policy. A short list includes:

- **Sparse rewards:** The first term in (103) represents a large penalty on the position error at the final time of the rendezvous. For data-driven and reinforcement learning methods based on simulation, sparse rewards must be carefully considered in reward shaping as this penalty is only evaluated at the final time.
- **Running cost:** The reward shaping furthermore must accommodate a running cost on states and commands,

in the second term of (103). As compared to sparse rewards, the running cost generically contributes at each time t and thus must be appropriately weighted in the reward/cost shaping to not overshadow the sparse reward/cost.

- **Finite time completion:** The goal of completing the task in finite time may preclude classes of asymptotic control strategies and control strategies which do not have certifiable guarantees of finite time completion.
- **Dynamical system equality constraints:** The control strategies and the resulting state trajectories must satisfy the dynamics in (99)-(102).
- **Control (acceleration) limits:** Due to vehicle performance limitations, there may exist a limit on the magnitude of the acceleration command, $a(t)$.
- **Control (acceleration) state-dependent direction constraints:** In this problem formulation, it is assumed that the commanded acceleration must be normal to the velocity vector of the vehicle. This state-dependent control constraint may be due to the vehicle configuration.
- **State path constraints:** Constraints at each time, t, may be included on the vehicle state.

Overcoming each of these items using generic (deep) neural network function approximators for the acceleration policy may prove challenging. Designing the appropriate training/reward function to incorporate each item may require significant hand-tuning. Furthermore, generic (deep) neural networks only provide for a finite approximation level, thus potentially resulting in a policy that may violate one or more of the listed items, e.g., the acceleration command must be *exactly normal* to the vehicle velocity (not *approximately normal*), due to e.g., vehicle configuration.

To overcome many of these challenges, this tutorial introduces the consideration of a *parameterized acceleration command using prototypical control structures*. One example structure of an acceleration command for rendezvous problems is called proportional navigation [107], where a normal acceleration command is chosen as

$$a_{normal}(t) = N\dot{\lambda}(t)V_c(t), \tag{104}$$

where $\dot{\lambda}$ is the line of sight rate, V_c is the closing velocity, and N is a gain. As shown in [107], the gain N can be chosen in classes of rendezvous settings such that many of the listed challenges are immediately overcome as a result of the choice of the particular control structure in (104), such as successful rendezvous (sparse reward), finite time completion, and acceleration direction constraints (normal acceleration). This *parametric control policy* can be viewed as a significant dimensionality reduction in policy search spaces, as the *parametric structure* can automatically satisfy many of the listed challenges for ranges of the gain N.

Thus this tutorial draws a distinction between learning a direct control (acceleration) policy and learning gains of a control-based parametric structure. In the example rendezvous problem statement, the two approaches may be mathematically expressed as

$$a_{direct}(t) = \pi_{direct}(\mathcal{I}(t); \theta_d), \tag{105}$$

$$a_{parametric}(t) = \pi_N(\mathcal{I}(t); \theta_N) \cdot \dot{\lambda}(t)V_c(t) \begin{bmatrix} -\sin(\psi) \\ \cos(\psi) \end{bmatrix}, \tag{106}$$

where π_{direct} and π_N are respectively the direct control policy and the gain policy, $\mathcal{I}(t)$ represents measurable inputs to the policy at each time t, ψ is the vehicle heading angle, and θ_d, θ_N are weights that may be learned for each policy using a data-driven approach.

Figure 3 is provided to illustrate the shapes of example rendezvous trajectories for the setting of a model point moving at a constant velocity in the X-direction, at a constant Y-position. The vehicle employs the parametric acceleration-based control in (106) with a fixed gain policy - fixed $\pi_N = N$. The policy search space is considered to be finite with values of $N \in \{3, 3.5, 4, 4.5, 5\}$. For each value of the gain N depicted, the resulting rendezvous is successful (sparse reward), completed in a finite time, and is accomplished with a normal acceleration command obeying vehicle equality constraints - thus satisfying many of the listed challenges by the parametric structure of the control (acceleration command). Therefore, the gain-based policy search can focus on other listed challenges and the respective optimization.

As an example, consider a minimum time objective with the depicted normal acceleration limit. From the plots of Fig. 3, it can be seen that the fixed policy of $N = 5$ results in the minimum time rendezvous in the policy search space. However, the gain $N = 5$ is observed to violate the notional normal acceleration limit (as does the gain choice $N = 4.5$). Thus the optimal fixed-gain policy, among the five policies in the policy search space, to minimize the time of rendezvous while satisfying the notional normal acceleration limit can be seen to be $N = 4$.

In this example, the structure of the parameterized control with a *data-driven gain policy search* via trajectory experience was demonstrated to greatly simplify and streamline the data-driven control optimization. The parametric structure automatically accommodated a sparse reward and classes of constraints, enabling the data-driven optimization to focus on a scalar gain, thereby greatly reducing the complexity of the data-driven policy optimization. Promising future directions exist in leveraging more expressive prototypical control structures, with continuous state-dependent gain policies tuned with a data-driven optimization.

B. Application Examples of Static Models Method

The static model method in section V-A is able to control the process to meet certain objectives, such as meeting targets, obeying constraints, and minimizing/maximizing output to a wide range of applications [68], [108]–[114]. In this section, we will show three different applications of the algorithm.

1) PID Gain Tuning

PID controller has been widely used in all industries for a great variety of applications. So far, a lot of efforts have been

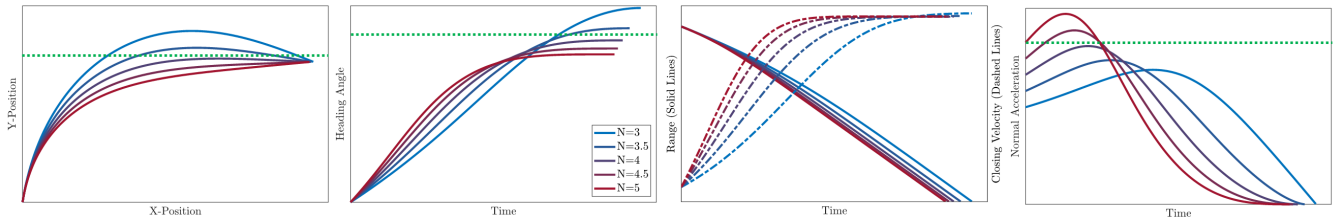


Fig. 3: Example proportional navigation-based rendezvous histories, for five different values of the gain N. The green-dashed lines are example state limits (to remain below), where it can be seen that certain values of N result in policies that satisfy the limits.

made to develop tools to minimize time to search for optimal PID gains. In this section, we apply the algorithm to tune the PID gains to deliver the best closed-loop response. The process, in this case, is the gain tuning of a closed-loop motor PID controller, where the inputs to the tuning process are the PID gains, and the outputs of the process are the closed-loop system step response measurements of the steady-state error, overshoot, and settling time. The objective of the tuning process is to have the closed-loop step responses meet desired performance criteria of 1% overshoot and steady-state error, and minimum settling time. Fig. 4a shows the step response improvement over ten optimization iterations. Both the steady-state error and overshoot are within the targets after the 4^{th} iteration. The settling time was further minimized in the following iterations, as also shown in Fig. 4b. The optimization was stopped when there was no significant improvement in settling time.

It is noted that motor parameters change with the temperature, so the dynamics also change as temperature changes. The algorithm was extended to adapt the controller gains to compensate for the change in temperature. Fig. 4c showed the difference between fixed and adaptive gains when the motor coil temperature changes.

2) Lane change maneuver

Another application of the algorithm is to adapt the reference in a lane change problem without adapting the tracking controller gains. As shown in Fig. 5 the car does not follow the reference trajectory (target vs response 1). To make the response match the target, a virtual command is generated that is not the same as the target, but the response to the virtual command will match the target profile. Therefore, a parameterized profile was generated with eight samples along the trajectory and optimized using the static map method. With this same algorithm, a virtual command (dash line) was generated to have a response that matches the target.

3) High-speed Sailboats

Here, we outline the third application of JL to optimize the sailing maneuvers of an America's Cup AC75 foiling sailboat [68], [114]. Because foiling yachts have multiple articulations for rapid motions and maneuverability, their dynamics are nonlinear, high-dimensional, and unstable. These complex dynamics must be considered to accomplish aggressive and optimal maneuvers instead of analytical optimizations using reduced-order models. JL was also utilized to explore out-of-box articulations. The controllers were incorporated with a high-fidelity sailboat simulator to optimize maneuvers safely and effectively. Constraints imposed by the physical/actuator system on the ideal solutions ensured human maneuverability (sailors). The close-hauled and tacking movements were optimized to maximize Velocity Made Good (VMG) and minimize VMG loss, respectively. The best maneuvers take advantage of favorable wind conditions in the racing setting with a marginal VMG loss of less than 1.5% as shown in Fig. 6.

C. Sparse Nonlinear Models of Fluids

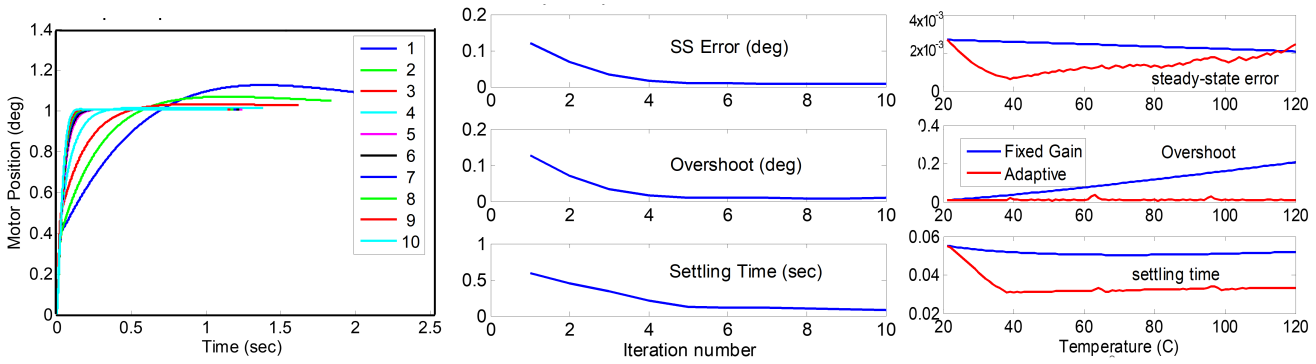
In this section, we outline an application of SINDy discussed in Section III-D. Fluid dynamics are central to several trillion-dollar industries, including health, defense, energy, and transportation. An improved ability to model and control fluids may be an enabler of advanced technology. However, the field of fluid dynamics poses several challenges for control [115], [116]. The dynamics are typically high-dimensional, nonlinear, and multiscale in both space and time. Further, relevant timescales are often quite fast for modern applications, requiring fast control decisions to reduce latency. There are also often strong instabilities and time delays between sensing and actuation.

Recently, considerable progress has been made in the data-driven modeling and control of fluids, with several papers investigating interpretable models with the SINDy framework [28], [29]. Examples have been studied in fluid dynamics [29], [117]–[122], electroconvection [123], and plasmas [32], [124]. Most of these studies have built interpretable mechanistic models for complex flows, but control is typically an ongoing challenge. However, SINDy-MPC is a promising candidate to bridge this gap [37]. It is also possible to combine SINDy for reduced-order modeling with autoencoders for dimensionality reduction, as in Champion et al. [125]. This approach may enable even more compressed models in an appropriate coordinate system.

Related sparse modeling techniques have been used for closure modeling of fluids, enabling both Reynolds averaged Navier-Stokes (RANS) closures [126]–[128] and large eddy simulation (LES) closures [129]. Turbulence modeling, and eventually control, is one of the great open problems in classical physics, with tremendous potential to drive and advance future technologies.

D. DPC Application Example to Building Control

This section briefly provides an example application of the DPC methodology to a building control use case [84]. For the



(a) Step Response Improvement (b) Step Response Performance Matrix (c) Control Adaptation

Fig. 4: a) Step Response Improvement, b) Step Response Performance Matrix, c) Control Adaptation using static models

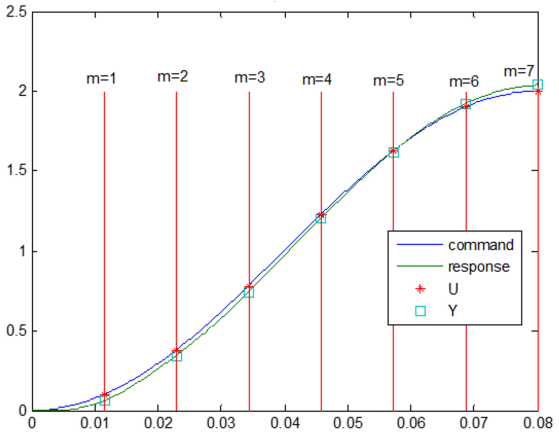


Fig. 5: Parametrization for Tuning

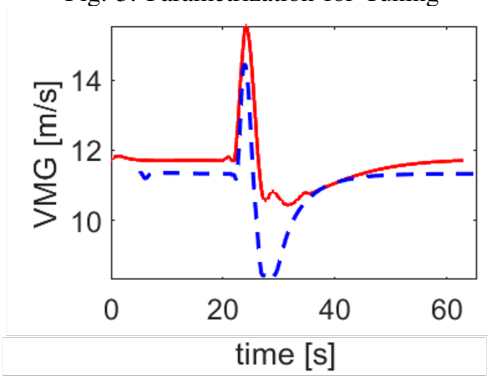


Fig. 6: Baseline trajectory compared to the optimized trajectory (increased VMG and reduced VMG loss)

system identification step of the DPC algorithm, we use the building physics emulator described in [130] to generate the time series dataset $\boldsymbol{D}_{\text{sysID}}$ sampled with $T_s = 15$ min rate, in the form of tuples of control inputs $\mathbf{u}$, disturbance signals $\mathbf{d}$, and system outputs $\mathbf{y}$ given as:

$$\boldsymbol{D}_{\text{sysID}} = \{(\mathbf{u}_t^{(i)}, \mathbf{d}_t^{(i)}, \mathbf{y}_t^{(i)}), (\mathbf{u}_{t+T_s}^{(i)}, \mathbf{d}_{t+T_s}^{(i)}, \mathbf{y}_{t+T_s}^{(i)}), \ldots, (\mathbf{u}_{t+NT_s}^{(i)}, \mathbf{d}_{t+NT_s}^{(i)}, \mathbf{y}_{t+NT_s}^{(i)})\}, \tag{107}$$

with n batches indexed by $i = \mathbb{N}_1^n$ of time series trajectories, each N-steps long. We use this dataset that carries the building operation data to train the parameters of the physics-constrained autoregressive state-space model.

In the second step of the DPC algorithm, we train the control law $\boldsymbol{\pi}_\theta(\boldsymbol{\xi})$ using the following synthetically generated training dataset of sampled past output trajectories $\tilde{\mathbf{y}}$, representing perturbation of the initial conditions for the closed-loop system dynamics. Similarly, we sample $\underline{\mathbf{y}}$, $\overline{\mathbf{y}}$, $\underline{\mathbf{u}}$, and $\overline{\mathbf{u}}$ representing time-varying lower and upper bounds on controlled outputs and control actions, respectively. For measured disturbances $\mathbf{d}$, we assume to have access to their N-step ahead forecast. We select a subset of trajectories from dataset $\boldsymbol{D}_{\text{ctrl}}$ to act as features $\boldsymbol{\xi}$ for the neural control law (91) with their compact notation $\tilde{\mathbf{Y}}_p = [\tilde{\mathbf{y}}_{t-NT_s}^{(i)}; \ldots; \tilde{\mathbf{y}}_t^{(i)}]$, $\underline{\mathbf{Y}}_f = [\underline{\mathbf{y}}_t^{(i)}; \ldots; \underline{\mathbf{y}}_{t+NT_s}^{(i)}]$, $\mathbf{D}_f = [\mathbf{d}_t^{(i)}; \ldots; \mathbf{d}_{t+NT_s}^{(i)}]$. Both datasets $\boldsymbol{D}_{\text{sysID}}$ and $\boldsymbol{D}_{\text{ctrl}}$ contain 8640 time samples corresponding to 90 days of the building operation data. We split each into training, development, and test sets of equal length.

The resulting control policy $\boldsymbol{\pi}_\theta(\boldsymbol{\xi}) : \mathbb{R}^{416} \to \mathbb{R}^{224}$ has 4 layers with 100 hidden units and uses GELU activation functions. The policy has 188624 trainable parameters. 416 features $\boldsymbol{\xi} = [\tilde{\mathbf{Y}}_p; \underline{\mathbf{Y}}_f; \mathbf{D}_f]$ and overall 224 outputs corresponding to vectorized control sequences for seven control actions predicted over 32 horizon $\mathbf{U}_f$.

Fig. 7 plots the 10-day long closed-loop control trajectories of the three controlled states representing building zone temperatures and the associated three control actions representing mass flow rates. The control actions are computed by neural control policy trained with DPC algorithm (1) using the learned neural state-space model. Thus we demonstrate the capability to optimize explicit control policies with near-optimal performance.

Based on the reported results and open-source implementation [131], the proposed DPC algorithm can be deployed in the application domains beyond the computational reach of the traditional MPC. Compared to MPC, DPC has faster online execution than implicit MPC and a smaller memory footprint than explicit MPC [76].

VII. Summary

Data-driven modeling and control have gained a lot of attention in the past few years due to the significant advancements in sensing, computation, communication, and actuation. In this tutorial, we provide an exposition of the current

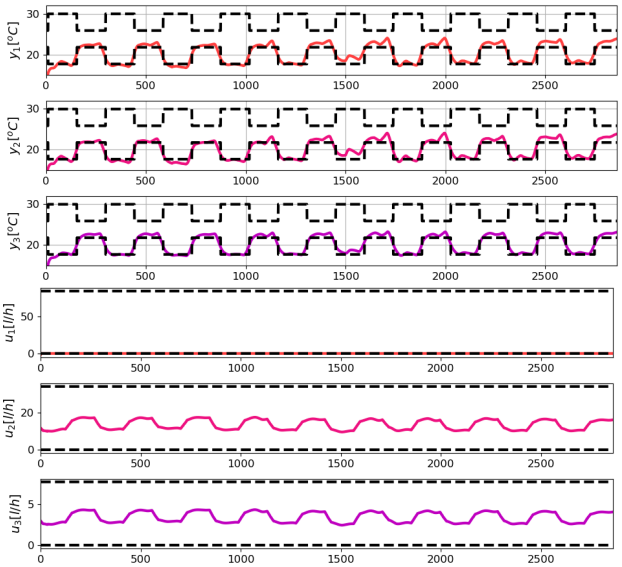


Fig. 7: Closed-loop control trajectories of the building simulation model governed by explicit DPC policy.

state of the art in data-driven modeling and control. We have made an effort to connect the topic to historical developments and highlights in previous years as well as recent methods that have garnered attention. We have provided examples of several applications that have successfully utilized such data-driven methods.

REFERENCES

[1] K. Åström and B. Wittenmark, "Problems of identification and control," *Journal of Mathematical Analysis and Applications*, vol. 34, pp. 90–113, Apr. 1971.

[2] K. S. Narendra and A. M. Annaswamy, "Persistent excitation in adaptive systems," *International Journal of Control*, vol. 45, pp. 127–160, jan 1987.

[3] E. W. Bai and S. S. Sastry, "Persistency of excitation, sufficient richness and parameter convergence in discrete time adaptive control," *Systems & Control Letters*, vol. 6, no. 3, pp. 153–163, 1985-08.

[4] L. Ljung, *System Identification: Theory for the User*. Prentice-Hall, 1987.

[5] G. Goodwin and K. Sin, *Adaptive Filtering Prediction and Control*. Prentice-Hall, 1984.

[6] S. Parker and F. Perry, "A discrete ARMA model for nonlinear system identification," *IEEE Transactions on Circuits and Systems*, vol. 28, no. 3, pp. 224–233, 1981.

[7] K. J. Åström and P. Eykhoff, "System identification—a survey," *Automatica*, vol. 7, no. 2, pp. 123–162, 1971.

[8] M. C. Campi and P. R. Kumar, "Adaptive linear quadratic gaussian control: the cost-biased approach revisited," *SIAM Journal on Control and Optimization*, vol. 36, no. 6, pp. 1890–1907, 1998.

[9] B. Ho and R. Kalman, "Effective construction of linear state-variable models from input/output functions," *at-Automatisierungstechnik*, vol. 14, no. 1-12, pp. 545–548, 1966.

[10] J.-N. Juang and R. S. Pappa, "Effects of noise on modal parameters identified by the Eigensystem Realization Algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 9, pp. 294–303, May 1986.

[11] P. V. Overschee and B. D. Moor, *Subspace Identification for Linear Systems: Theory, Implementation, Applications*. Kluwer Academic Publishers, 1996.

[12] B. D. Moor, "N4SID1 : Subspace Algorithms for the Identi cation of Combined Deterministic-Stochastic Systems 2," *Automaticau*, p. 52, 1992.

[13] M. Verhaegen and P. Dewilde, "Subspace model identification Part 2. Analysis of the elementary output-error state-space model identification algorithm," *International Journal of Control*, vol. 56, pp. 1211–1241, Nov. 1992.

[14] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, pp. 716–723, Dec. 1974.

[15] W. E. Larimore, "System Identification, Reduced-Order Filtering and Modeling via Canonical Variate Analysis," in *1983 American Control Conference*, pp. 445–451, June 1983.

[16] T. F. Chan, "An improved algorithm for computing the singular value decomposition," *ACM Trans. Math. Software*, vol. 8, no. 1, pp. 72–83, 1982.

[17] J.-N. Juang, M. Phan, L. G. Horta, and R. W. Longman, "Identification of Observer/Kalman Filter Markov Parameters: Theory and Experiments," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 2, p. 10, 1993.

[18] C. W. Rowley and S. T. Dawson, "Model Reduction for Flow Analysis and Control," *Annu. Rev. Fluid Mech.*, vol. 49, pp. 387–417, Jan. 2017.

[19] M. Viberg, "Subspace Methods in System Identification," *IFAC Proceedings Volumes*, vol. 27, pp. 1–12, July 1994.

[20] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *J. Fluid Mech.*, vol. 656, pp. 5–28, Aug. 2010.

[21] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proc Natl Acad Sci USA*, vol. 113, pp. 3932–3937, Apr. 2016.

[22] J. Annoni and P. Seiler, "A method to construct reduced-order parameter-varying models," *Int. J. Robust. Nonlinear Control*, vol. 27, no. 4, pp. 582–597, 2017.

[23] K. K. Chen, J. H. Tu, and C. W. Rowley, "Variants of Dynamic Mode Decomposition: Boundary Condition, Koopman, and Fourier Analyses," *J Nonlinear Sci*, vol. 22, pp. 887–915, Dec. 2012.

[24] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks," *Proc. R. Soc. A*, vol. 474, no. 2213, p. 20170844, 2018.

[25] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach," *Physical review letters*, vol. 120, no. 2, p. 024102, 2018.

[26] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian neural networks," *arXiv preprint arXiv:2003.04630*, 2020.

[27] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *arXiv preprint arXiv:1806.07366*, 2018.

[28] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[29] J.-C. Loiseau and S. L. Brunton, "Constrained sparse Galerkin regression," *Journal of Fluid Mechanics*, vol. 838, pp. 42–67, 2018.

[30] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science Advances*, vol. 3, no. e1602614, 2017.

[31] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Inferring biological networks by sparse identification of nonlinear dynamics," *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, vol. 2, no. 1, pp. 52–63, 2016.

[32] A. A. Kaptanoglu, J. L. Callaham, C. J. Hansen, A. Aravkin, and S. L. Brunton, "Promoting global stability in data-driven models of quadratic nonlinear dynamics," *Physical Review Fluids*, vol. 6, no. 094401, 2021.

[33] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 104, no. 24, pp. 9943–9948, 2007.

[34] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[35] M. D. Cranmer, R. Xu, P. Battaglia, and S. Ho, "Learning symbolic physics with graph networks," *arXiv preprint arXiv:1909.05862*, 2019.

[36] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, "Discovering symbolic models from deep learning with inductive biases," *arXiv preprint arXiv:2006.11287*, 2020.

[37] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society of London A*, vol. 474, no. 2219, 2018.

[38] U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton, "Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control," *Proceedings of the Royal Society A*, vol. 478, no. 2260, p. 20210904, 2022.

[39] M. Budišić, R. Mohr, and I. Mezić, "Applied Koopmanism," *Chaos*, vol. 22, p. 047510, Dec. 2012.

[40] I. Mezić, "Spectral Properties of Dynamical Systems, Model Reduction and Decompositions," *Nonlinear Dyn*, vol. 41, pp. 309–325, Aug. 2005.

[41] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," *J. Fluid Mech.*, vol. 641, pp. 115–127, Dec. 2009.

[42] M. Korda and I. Mezic, "Optimal construction of Koopman eigenfunctions for prediction and control," *IEEE Transactions on Automatic Control*, pp. 1–1, 2020.

[43] F. Nüske, B. G. Keller, G. Pérez-Hernández, A. S. J. S. Mey, and F. Noé, "Variational Approach to Molecular Kinetics," *J. Chem. Theory Comput.*, vol. 10, pp. 1739–1752, Apr. 2014.

[44] S. Klus, P. Koltai, and C. Schütte, "On the numerical approximation of the Perron-Frobenius and Koopman operator," *JCD*, vol. 3, pp. 1–12, Sept. 2016.

[45] S. Klus, F. Nüske, P. Koltai, H. Wu, I. Kevrekidis, C. Schütte, and F. Noé, "Data-Driven Model Reduction and Transfer Operator Approximation," *J Nonlinear Sci*, vol. 28, pp. 985–1010, June 2018.

[46] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition," *J Nonlinear Sci*, vol. 25, pp. 1307–1346, Dec. 2015.

[47] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis, "Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator," *Chaos*, vol. 27, p. 103111, Oct. 2017.

[48] S. E. Otto and C. W. Rowley, "Linearly Recurrent Autoencoder Networks for Learning Dynamics," *SIAM J. Appl. Dyn. Syst.*, vol. 18, pp. 558–593, Jan. 2019.

[49] N. Takeishi, Y. Kawahara, and T. Yairi, "Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 1130–1140, Curran Associates, Inc., 2017.

[50] E. Yeung, S. Kundu, and N. Hodas, "Learning Deep Neural Network Representations for Koopman Operators of Nonlinear Dynamical Systems," in *ACC'19*, pp. 4832–4839, July 2019.

[51] S. Peitz and S. Klus, "Koopman operator-based model reduction for switched-system control of PDEs," *Automatica*, vol. 106, pp. 184–191, Aug. 2019.

[52] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Sci. Adv.*, vol. 3, p. e1602614, Apr. 2017.

[53] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[54] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[55] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[56] A. M. Annaswamy and A. L. Fradkov, "A historical perspective of adaptive control and learning," *Annual Reviews in Control*, vol. 52, pp. 18–41, 2021.

[57] A. L. Bruce, A. Goel, and D. S. Bernstein, "Convergence and consistency of recursive least squares with variable-rate forgetting," *Automatica*, vol. 119, p. 109052, 2020.

[58] Y. Cui, J. E. Gaudio, and A. M. Annaswamy, "New algorithms for discrete-time parameter estimation," in *2022 American Control Conference (ACC)*, pp. 3382–3387, IEEE, 2022.

[59] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[60] J. E. Gaudio, A. M. Annaswamy, J. M. Moreu, M. A. Bolender, and T. E. Gibson, "Accelerated learning with robustness to adversarial regressors," *Proceedings of the 3rd Conference on Learning for Dynamics and Control, PMLR 144:636-650*, 2020.

[61] S. Bubeck, "Convex optimization: Algorithms and complexity," *Foundations and Trends in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

[62] Y. Nesterov, *Lectures on Convex Optimization*. Springer International Publishing, 2018.

[63] J. M. Moreu and A. M. Annaswamy, "A stable high-order tuner for general convex functions," *IEEE Control Systems Letters*, vol. 6, pp. 566–571, 2022.

[64] M. Vogt, N. Mu¨ller, and R. Isermann, "On-Line Adaptation of Grid-Based Look-up Tables Using a Fast Linear Regression Technique," *Journal of Dynamic Systems, Measurement, and Control*, vol. 126, pp. 732–739, Dec. 2004.

[65] D. Filev, T. Larsson, and Lixing Ma, "Intelligent control for automotive manufacturing-rule based guided adaptation," in *IECON 2000.*, vol. 1, (Nagoya, Japan), pp. 283–288, IEEE, 2000.

[66] D. P. Filev, S. Bharitkar, and M.-F. Tsai, "Nonlinear control of static systems with unsupervised learning of the initial conditions," in *18th NAFIPS (Cat. No. 99TH8397)*, pp. 169–173, 1999.

[67] T. Larsson, Lixing Ma, and D. Filev, "Adaptive control of a static multiple input multiple output system," in *ACC'2000*, (Chicago, IL, USA), pp. 2573–2577 vol.4, IEEE, 2000.

[68] R. Rodriguez, Y. Wang, J. Ozanne, J. Morrow, D. Sumer, D. Filev, and D. Soudbakhsh, "Adaptive Learning and Optimization of High-Speed Sailing Maneuvers for America's Cup," *Journal of Dynamic Systems, Measurement, and Control*, vol. 145, no. 2, p. 021005, 2022.

[69] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V. B. Shah, and W. Tebbutt, "A differentiable programming system to bridge machine learning and scientific computing," *CoRR*, vol. abs/1907.07587, 2019.

[70] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," in *Advances in Neural Information Processing Systems*, 2019.

[71] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[72] S. East, M. Gallieri, J. Masci, J. Koutnik, and M. Cannon, "Infinite-horizon differentiable model predictive control," in *International Conference on Learning Representations*, 2020.

[73] M. Pereira, D. D. Fan, G. N. An, and E. A. Theodorou, "Mpc-inspired neural network policies for sequential decision making," *CoRR*, vol. abs/1802.05803, 2018.

[74] M. Zanon, S. Gros, and A. Bemporad, "Practical reinforcement learning of stabilizing economic mpc," in *ECC'19*, pp. 2258–2263, 2019.

[75] S. Gros and M. Zanon, "Reinforcement learning based on mpc and the stochastic policy gradient method," in *ACC'21*, pp. 1947–1952, 2021.

[76] J. Drgoňa, K. Kiš, A. Tuor, D. Vrabie, and M. Klaučo, "Differentiable predictive control: Deep learning alternative to explicit model predictive control for unknown nonlinear systems," *Journal of Process Control*, vol. 116, pp. 80–92, 2022.

[77] M. Grötschel, S. O. Krumke, and J. Rambau, eds., *Introduction to Model Based Optimization of Chemical Processes on Moving Horizons*, pp. 295–339. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[78] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[79] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[80] J. Köhler, M. A. Müller, and F. Allgöwer, "A nonlinear model predictive control framework using reference generic terminal ingredients," *IEEE Transactions on Automatic Control*, vol. 65, no. 8, pp. 3576–3583, 2020.

[81] J. Drgoňa, S. Mukherjee, A. Tuor, M. Halappanavar, and D. Vrabie, "Learning stochastic parametric diferentiable predictive control policies," *IFAC-PapersOnLine*, vol. 55, no. 25, pp. 121–126, 2022.

[82] S. Mukherjee, J. Drgoňa, A. Tuor, M. Halappanavar, and D. Vrabie, "Neural lyapunov differentiable predictive control," in *IEEE-CDC*, pp. 2097–2104, 2022.

[83] W. S. Cortez, J. Drgona, A. Tuor, M. Halappanavar, and D. Vrabie, "Differentiable predictive control with safety guarantees: A control barrier function approach," in *IEEE-CDC*, pp. 932–938, 2022.

[84] J. Drgoňa, A. Tuor, E. Skomski, S. Vasisht, and D. Vrabie, "Deep learning explicit differentiable predictive control laws for buildings,"

IFAC-PapersOnLine, vol. 54, no. 6, pp. 14–19, 2021. 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021.

[85] E. King, J. Drgoňa, A. Tuor, S. Abhyankar, C. Bakker, A. Bhattacharya, and D. Vrabie, "Koopman-based differentiable predictive control for the dynamics-aware economic dispatch problem," in *ACC'22*, pp. 2194–2201, 2022.

[86] J. E. Gaudio, T. E. Gibson, A. M. Annaswamy, M. A. Bolender, and E. Lavretsky, "Connections between adaptive control and optimization in machine learning," in *IEEE CDC*, 2019.

[87] A. M. Annaswamy and A. L. Fradkov, "A historical perspective of adaptive control and learning," *Annual Reviews in Control*, vol. 52, pp. 18–41, 2021.

[88] P. A. Ioannou and P. V. Kokotovic, "Robust redesign of adaptive control," *IEEE Transactions on Automatic Control*, vol. 29, pp. 202–211, mar 1984.

[89] K. S. Narendra and A. M. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," *IEEE Transactions on Automatic Control*, vol. 32, pp. 134–145, feb 1987.

[90] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[91] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[92] E. Hazan, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[93] J. E. Gaudio, A. M. Annaswamy, E. Lavretsky, and M. A. Bolender, "Parameter estimation in adaptive control of time-varying systems under a range of excitation conditions," *IEEE Transactions on Automatic Control*, 2022.

[94] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, July 2011.

[95] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2017.

[96] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018.

[97] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, pp. 183–202, jan 2009.

[98] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," vol. 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, PMLR, 2013.

[99] A. S. Morse, "High-order parameter tuners for the adaptive control of linear and nonlinear systems," in *Systems, Models and Feedback: Theory and Applications*, pp. 339–364, Birkhäuser Boston, 1992.

[100] J. E. Gaudio, A. M. Annaswamy, M. A. Bolender, E. Lavretsky, and T. E. Gibson, "A class of high order tuners for adaptive systems," *IEEE Control Systems Letters*, 2021.

[101] J. E. Gaudio, A. M. Annaswamy, J. M. Moreu, M. A. Bolender, and T. E. Gibson, "Accelerated learning with robustness to adversarial regressors," in *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, 2021.

[102] D. Bertsekas, *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[103] A. M. Annaswamy, "Adaptive control and intersections with reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 6, 2023.

[104] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in neural information processing systems*, pp. 908–918, 2017.

[105] B. Pang, Z.-P. Jiang, and I. Mareels, "Reinforcement learning for adaptive optimal control of continuous-time linear periodic systems," *Automatica*, vol. 118, p. 109035, 2020.

[106] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

[107] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control*. Taylor & Francis Group, 1975.

[108] R. Gupta, I. V. Kolmanovsky, Yan Wang, and D. P. Filev, "Onboard learning-based fuel consumption optimization in series hybrid electric vehicles," in *2012 American Control Conference (ACC)*, (Montreal, QC), pp. 1308–1313, IEEE, June 2012.

[109] S. Thornton, A. Annaswamy, D. Yanakiev, G. M. Pietron, B. Riedle, D. Filev, and Y. Wang, "Adaptive shift control for automatic transmissions," in *ASME-DSCD*, vol. 46193, p. V002T27A001, 2014.

[110] D. Filev, Y. Wang, and I. Kolmanovsky, "Learning-based approaches to engine mapping and calibration optimization," *Optimization and Optimal Control in Automotive Systems*, pp. 257–272, 2014.

[111] D. Simon, Yan Wang, O. Tiber, Dawei Du, D. Filev, and J. Michelini, "Trajectory optimization with memetic algorithms: Time-to-torque minimization of turbocharged engines," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, (Budapest, Hungary), pp. 000983–000988, IEEE, Oct. 2016.

[112] A. D'Amato, Y. Wang, D. Filev, and E. Remes, "On the Tradeoffs between Static and Dynamic Adaptive Optimization for an Automotive Application," *SAE Int. J. Commer. Veh.*, vol. 10, pp. 346–352, Mar. 2017.

[113] J. Wang, J. Michelini, Y. Wang, and M. H. Shelby, "Time to torque optimization by evolutionary computation methods," tech. rep., SAE Technical Paper, 2017.

[114] R. Rodriguez, Y. Wang, J. Ozanne, D. Sumer, D. Filev, and D. Soudbakhsh, "Adaptive Takeoff Maneuver Optimization of a Sailing Boat for America's Cup," *Journal of Sailing Technology*, vol. 7, no. 01, pp. 88–103, 2022-10-17.

[115] S. L. Brunton and B. R. Noack, "Closed-loop turbulence control: Progress and challenges," *Applied Mechanics Reviews*, vol. 67, pp. 050801–1–050801–48, 2015.

[116] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annual Review of Fluid Mechanics*, vol. 52, pp. 477–508, 2020.

[117] J.-C. Loiseau, B. R. Noack, and S. L. Brunton, "Sparse reduced-order modeling: sensor-based dynamics to full-state estimation," *Journal of Fluid Mechanics*, vol. 844, pp. 459–490, 2018.

[118] N. Deng, B. R. Noack, M. Morzynski, and L. R. Pastur, "Low-order model for successive bifurcations of the fluidic pinball," *Journal of fluid mechanics*, vol. 884, no. A37, 2020.

[119] N. Deng, B. R. Noack, M. Morzyński, and L. R. Pastur, "Galerkin force model for transient and post-transient dynamics of the fluidic pinball," *Journal of Fluid Mechanics*, vol. 918, 2021.

[120] J. L. Callaham, J.-C. Loiseau, G. Rigas, and S. L. Brunton, "Nonlinear stochastic modelling with langevin regression," *Proceedings of the Royal Society A*, vol. 477, no. 2250, p. 20210092, 2021.

[121] J. L. Callaham, S. L. Brunton, and J.-C. Loiseau, "On the role of nonlinear correlations in reduced-order modeling," *Journal of Fluid Mechanics*, vol. 938, no. A1, 2022.

[122] J. L. Callaham, G. Rigas, J.-C. Loiseau, and S. L. Brunton, "An empirical mean-field model of symmetry-breaking in a turbulent wake," *Science Advances*, vol. 8, no. eabm4786, 2022.

[123] Y. Guan, S. L. Brunton, and I. Novosselov, "Sparse nonlinear models of chaotic electroconvection," *Royal Society Open Science*, vol. 8, no. 8, p. 202367, 2021.

[124] A. A. Kaptanoglu, K. D. Morgan, C. J. Hansen, and S. L. Brunton, "Physics-constrained, low-dimensional models for mhd: First-principles and data-driven approaches," *Physical Review E*, vol. 104, no. 015206, 2021.

[125] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, pp. 22445–22451, 2019.

[126] S. Beetham and J. Capecelatro, "Formulating turbulence closures using sparse regression with embedded form invariance," *Physical Review Fluids*, vol. 5, no. 8, p. 084611, 2020.

[127] S. Beetham, R. O. Fox, and J. Capecelatro, "Sparse identification of multiphase turbulence closures for coupled fluid–particle flows," *Journal of Fluid Mechanics*, vol. 914, 2021.

[128] M. Schmelzer, R. P. Dwight, and P. Cinnella, "Discovery of algebraic reynolds-stress models using sparse symbolic regression," *Flow, Turbulence and Combustion*, vol. 104, no. 2, pp. 579–603, 2020.

[129] L. Zanna and T. Bolton, "Data-driven equation discovery of ocean mesoscale closures," *Geophysical Research Letters*, vol. 47, no. 17, p. e2020GL088376, 2020.

[130] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," *Applied Energy*, vol. 218, pp. 199 – 216, 2018.

[131] A. Tuor, J. Drgona, and E. Skomski, "NeuroMANCER: Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations," 2021.