SINDy with Control: A Tutorial

Urban Fasel, Eurika Kaiser, J. Nathan Kutz, Bingni W. Brunton, and Steven L. Brunton

***Abstract*— Many dynamical systems of interest are nonlinear, with examples in turbulence, epidemiology, neuroscience, and finance, making them difficult to control using linear approaches. Model predictive control (MPC) is a powerful model-based optimization technique that enables the control of such nonlinear systems with constraints. However, modern systems often lack computationally tractable models, motivating the use of system identification techniques to learn accurate and efficient models for real-time control. In this tutorial article, we review emerging data-driven methods for model discovery and how they are used for nonlinear MPC. In particular, we focus on the sparse identification of nonlinear dynamics (SINDy) algorithm and show how it may be used with MPC on an infectious disease control example. We compare the performance against MPC based on a linear dynamic mode decomposition (DMD) model. Code is provided to run the tutorial examples and may be modified to extend this data-driven control framework to arbitrary nonlinear systems.**

Keywords: Model predictive control, data-driven models, machine learning, system identification, SINDy, DMD

I. INTRODUCTION

Modern systems of interest in turbulence, epidemiology, neuroscience, and finance are high-dimensional and nonlinear, and exhibit multiscale phenomena in both space and time [1]. Controlling these nonlinear systems remains an important challenge, as traditional linear control approaches are often insufficient. There are several control approaches for nonlinear systems, including model predictive control [2] and reinforcement learning [3]. These approaches have different performance tradeoffs and requirements, and careful consideration must be given to factors such as the dimensionality of the system, access to an accurate model of the behavior, and the availability of high-quality and abundant data. Both approaches are developing rapidly, driven by advanced algorithms in machine learning and optimization, the rise of big data, and improved computational hardware [4].

Model predictive control (MPC) [5], [6], [2] is a particularly compelling approach that enables control of strongly nonlinear systems with constraints. However, MPC relies on efficient models that accurately represent the dynamics of the system, and these models have remained elusive for many disciplines that lack known governing equations. Generally, MPC also suffers from the curse of dimensionality, requiring large computational effort and limiting the applicability to low-dimensional problems, often based on locally linear models. Increasingly, data-driven approaches are providing a hierarchy of models of various fidelity and complexity, which may be used for MPC.

System identification has a long and rich history in control theory, and it has a close connection with machine learning, as it builds models from data via regression and optimization [4]. A wide range of data-driven system identification techniques exist in literature, including state-space models from the eigensystem realization algorithm (ERA) [7] and other subspace identification methods, Volterra series [8], [9], autoregressive models [10] (e.g. ARX, ARMA, NARX and NARMAX [11] models), and neural network (NN) models [12], [13], [14], [15]. Nonlinear models based on machine learning, such as NNs, are increasingly used due to advances in computing power, and recently deep learning has been combined with model predictive control [16] and applied to several complex systems [17], [18], [19], [20]. Deep reinforcement learning has also been combined with MPC [21], [22], yielding impressive results in the large-data limit. However, machine learning algorithms often lack interpretability and may suffer from overfitting.

The recent sparse identification of nonlinear dynamics (SINDy) algorithm [23] provides a data-driven model discovery framework, resulting in interpretable models that avoid overfitting, relying on sparsity-promoting optimization to identify parsimonious models from limited data. SINDy is closely related to the dynamic mode decomposition (DMD) [24], [25], [1], which results in a linear model for the evolution of a few key spatiotemporal coherent structures from high-dimensional time-series data. DMD was generalized to include external inputs and control [26], and this approach was later applied to develop SINDy with control [27]. Both approaches have been combined with MPC, enabling interpretable models in the low-data limit for real-time control of nonlinear systems.

In this tutorial article, we review SINDy with control and demonstrate its effectiveness with MPC on an illustrative infectious disease control problem. A primary goal of this tutorial is to provide the tools to apply data-driven system identification methods to a model predictive control problem. Although we demonstrate the approach with SINDy, many other data-driven modeling techniques may be used, as shown schematically in figure 1. Example codes are provided for the tutorial example to promote reproducible research.

This work was supported by the Air Force Office of Scientific Research (AFOSR FA9550-19-1-0386) and the Army Research Office (ARO W911NF-19-1-0045).

U. Fasel, E. Kaiser, and S. L. Brunton are with the Department of Mechanical Engineering, University of Washington, USA ufasel@uw.edu

J. N. Kutz is with the Department of Applied Mathematics, University of Washington, USA

B. W. Brunton is with the Department of Biology, University of Washington, USA

Code: https://github.com/urban-fasel/SEIR_SINDY_MPC

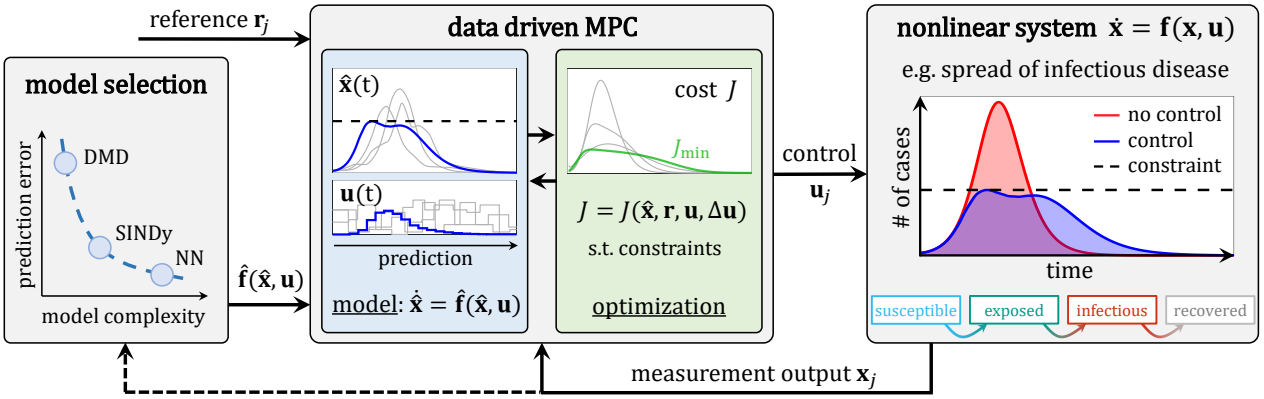

Fig. 1. Schematic of the data-driven MPC framework. Using either linear DMD, sparse nonlinear SINDy, or neural network models for predictive control.

II. DATA-DRIVEN MODELS FOR MPC

Here we describe data-driven control architectures that combine data-driven model discovery with advanced model-based control strategies. In this tutorial, we use the recent SINDy-MPC architecture [27] to rapidly identify a low-order model that is used with model predictive control. We consider a nonlinear dynamical system of the form:

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{x} = \mathbf{f}(\mathbf{x},\mathbf{u}), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{1}$$

with state $\mathbf{x} \in \mathbb{R}^n$, control input $\mathbf{u} \in \mathbb{R}^q$ and dynamics $\mathbf{f}(\mathbf{x},\mathbf{u}) : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^n$. In this section, we describe SINDy with control and MPC. This approach will be used in the next section to provide a step-by-step tutorial and code to compute SINDy-MPC for an infectious disease control problem.

A. Sparse identification of nonlinear dynamics with control

The SINDy [23] and SINDy with control [27] algorithms identify a sparse nonlinear dynamical system from measurement data, based on the assumption that many systems have relatively few active terms in the dynamics. SINDy with control uses sparse regression to identify these few active terms, out of a library $\Theta(\mathbf{x},\mathbf{u})$ of candidate linear and nonlinear model terms in the state $\mathbf{x}$ and actuation $\mathbf{u}$, that are required to approximate the function $\mathbf{f}$ in Eq. (1). Therefore, sparsity-promoting techniques may be used to find models that automatically balance model complexity with accuracy, resulting in parsimonious models [23]. The SINDy with control algorithm is illustrated in figure 2 on a disease model used in the next section. To evaluate Θ, we first measure m snapshots of the state $\mathbf{x}$ and the input signal $\mathbf{u}$ in time and arrange these into two matrices[1]:

$$\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_m]^T \quad \text{and} \quad \mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_m]^T. \tag{2}$$

After collecting the snapshots, we can evaluate the library of candidate nonlinear functions Θ:

$$\Theta(\mathbf{X},\mathbf{U}) = [\mathbf{1}\ \mathbf{X}\ \mathbf{U}\ (\mathbf{X}\otimes\mathbf{X})\ (\mathbf{X}\otimes\mathbf{U})\ (\mathbf{U}\otimes\mathbf{U})\ \cdots], \tag{3}$$

where $\mathbf{X}\otimes\mathbf{U}$ defines the vector of all product combinations of the components in $\mathbf{x}$ and $\mathbf{u}$. This library may include any functions that might describe the data. The choice of a suitable library is crucial. The recommended strategy is to start with a basic choice, such as low-order polynomials, and then increase the complexity and order of the library until sparse and accurate models are obtained.

In addition to evaluating the library, we must compute the time derivatives of the state $\dot{\mathbf{X}} = [\dot{\mathbf{x}}_1\ \dot{\mathbf{x}}_2\ \cdots\ \dot{\mathbf{x}}_m]$, typically by numerical differentiation. The system in Eq. (1) may then be written in terms of these data matrices as:

$$\dot{\mathbf{X}} = \Theta(\mathbf{X},\mathbf{U})\Xi. \tag{4}$$

Many dynamical systems have relatively few active terms in the governing equations. Thus, the coefficients Ξ are mostly sparse and we may employ sparse regression to identify the sparse matrix of coefficients Ξ signifying the fewest nonlinearities in our library that results in a good model fit:

$$\xi_k = \underset{\hat{\xi}_k}{\operatorname{argmin}} \frac{1}{2}\|\dot{\mathbf{X}}_k - \Theta(\mathbf{X},\mathbf{U})\hat{\xi}_k\|_2^2 + \lambda\|\hat{\xi}_k\|_0. \tag{5}$$

$\dot{\mathbf{X}}_k$ is the k-th row of $\dot{\mathbf{X}}$, ξ_k is the k-th row of Ξ, and λ is the sparsity-promoting hyperparameter. The term $\|\cdot\|_0$ promotes sparsity in the coefficient vector ξ_k, although it is not convex. To approximately solve this optimization, we use sequential thresholded least squares (STLS) [23].

SINDy is closely related to DMD [24], [25], [1], which extracts spatiotemporal coherent structures from high-dimensional data, along with a linear model for how their amplitudes evolve. DMD performs a similar regression to identify a linear discrete-time model $\mathbf{A}$ mapping $\mathbf{X}$ to $\mathbf{X}'$, a matrix with all columns advanced one time step: $\mathbf{X}' = \mathbf{A}\mathbf{X}$. If we formulate SINDy in discrete-time with linear library elements and with $\lambda = 0$, SINDy reduces to DMD. It was shown that in many cases, an identified DMD model may be sufficiently accurate to perform control, even for strongly nonlinear dynamical systems [27]. DMD may also provide a useful model until SINDy has collected enough data to accurately characterize the dynamics. The *extended DMD* algorithm [28], which seeks linear models in terms of a higher-dimensional state augmented with nonlinear functions of the original variables, has also been used for MPC [29]; these DMD-based control approaches are reviewed in [30].

[1]These matrices are the transposes of the DMD with control matrices [26].

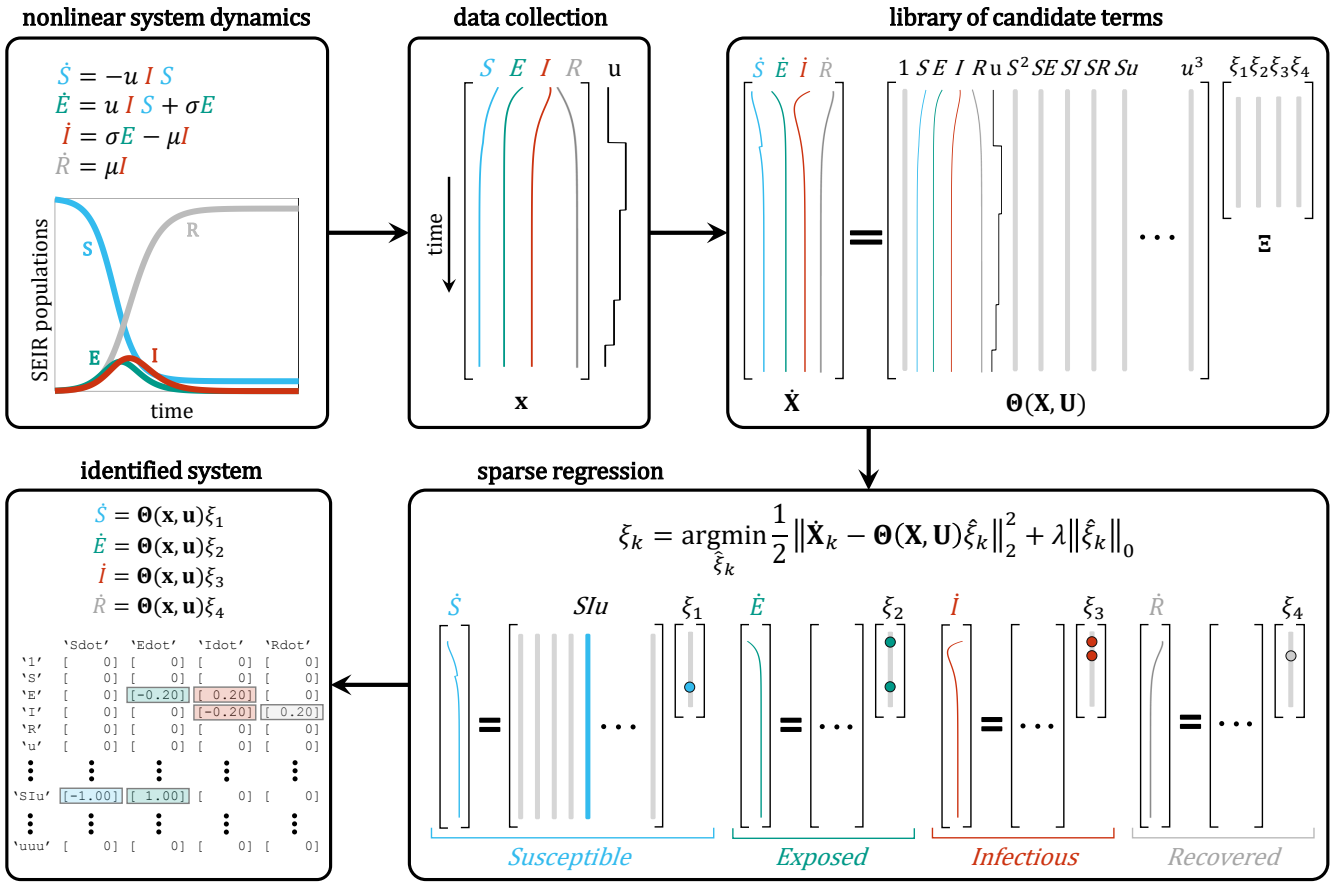

Fig. 2. Schematic of the SINDy with control algorithm. Active terms in a library of candidate nonlinearities are selected via sparse regression.

B. Model predictive control

This tutorial explores the use of SINDy models for model predictive control. MPC is an effective model-based control, which has revolutionized the industrial control landscape [31], [32]. MPC enables the control of strongly nonlinear systems with constraints, time delays, non-minimum phase dynamics, and instability. These systems are difficult to control using traditional linear approaches [5], [6], [2].

In MPC, we compute a control sequence $\mathbf{u}(\mathbf{x}_j) = \{\mathbf{u}_{j+1}, ..., \mathbf{u}_{j+m_c}\}$, given the current state estimate or measurement $\mathbf{x}_j$, by a constrained optimization over a receding horizon $T_c = m_c \Delta t$, with Δt the time step of the model and m_c the number of time steps. At each time step, we repeat the optimization, update the control sequence over the control horizon, and apply the first control action $\mathbf{u}_{j+1}$ to the system. The optimal control sequence $\mathbf{u}(\mathbf{x}_j)$ is obtained by minimizing a cost function J over a prediction horizon $T_p = m_p \Delta t \geq T_c$. The cost function is:

$$J = \sum_{k=0}^{m_p-1} \|\hat{\mathbf{x}}_{j+k} - \mathbf{r}_k\|_{\mathbf{Q}}^2 + \sum_{k=1}^{m_c-1} \left(\|\mathbf{u}_{j+k}\|_{\mathbf{R}_u}^2 + \|\Delta \mathbf{u}_{j+k}\|_{\mathbf{R}_{\Delta u}}^2 \right),$$

subject to the discrete-time dynamics and constraints. The cost function J penalizes deviations of the predicted state $\hat{\mathbf{x}}$ from the reference trajectory $\mathbf{r}$, the control expenditure $\mathbf{u}$, and the rate of change of the control signal $\Delta \mathbf{u}$. Each term is weighted by the matrices $\mathbf{Q}$, $\mathbf{R}_u$, and $\mathbf{R}_{\Delta u}$, respectively. To enable this optimization loop to run in real-time, MPC relies on efficient models and high-performance computing. This is particularly challenging for systems where the control must rapidly respond to disturbances on short time-scales, as time delays from sensors, signal transduction, or processing can destroy the robustness of feedback control, putting limitations on the achievable performance [33].

The advantage of MPC lies in simple and intuitive tuning and the ability to control complex phenomena, especially with known constraints and multiple operating conditions. It also works for systems with time delays and provides the flexibility to formulate and tailor a control objective. The major challenge of MPC lies in the development of a suitable model via system identification [27]. Nonlinear models based on machine learning are increasingly used. However, these techniques often rely on access to massive data sets, have limited ability to generalize, do not readily incorporate known physical constraints, and require expensive and time-consuming computations. Instead, Kaiser et al. [27] showed that simple models obtained via DMD with control [26] and SINDy with control [27] perform nearly as well with MPC on a full nonlinear model, and may be trained in a surprisingly short amount of time.

III. TUTORIAL ON INFECTIOUS DISEASE

We now introduce an infectious disease control problem and demonstrate SINDy-MPC to control the spread of the disease. We present the main code snippets directly in this tutorial. The complete code to generate the data, identify the SINDy model, and run the MPC can be found on GitHub: https://github.com/urban-fasel/SEIR_SINDY_MPC.

A. Intervention strategy for infectious disease

We first generate the data using an SEIR (susceptible-exposed-infectious-removed) epidemic model [34], [35], [36]. The SEIR model is a nonlinear system of ordinary differential equations that describes the transition dynamics between four compartments: susceptible $S(t)$ (individuals at risk of contracting the disease), exposed $E(t)$ (infected individuals but not yet infectious), infectious $I(t)$ (individuals capable of transmitting the disease), and removed $R(t)$ (recovered or dead individuals).

$$\begin{aligned} \dot{S}(t) &= -\beta S(t)I(t), \\ \dot{E}(t) &= \beta S(t)I(t) - kE(t), \\ \dot{I}(t) &= kE(t) - \gamma I(t), \\ \dot{R}(t) &= \gamma I(t) \end{aligned} \tag{6}$$

Susceptible individuals contract the virus at a rate of $\beta I(t)$, with β being the transmission rate per day. The incubation period $1/k$ determines the time to move from the exposed to the infectious compartment. The recovery period $1/\gamma$ determines the time to progress from the infectious to the recovered compartment. The basic reproduction number $R_0 = \beta/\gamma$ is an important estimate of the growth of the pandemic. R_0 indicates how many infections are generated on average by an infectious individual at the start of a pandemic, when most of the individuals are susceptible. If $R_0 \leq 1$, the infectious cases are declining and the spread of the disease eventually goes to zero.

To control the spread of an infectious disease, the reproduction number may be reduced through intervention, such as restricted travel, home confinement, or social distancing [37], [38]. To mitigate the spread, R_0 needs to be reduced, and to suppress the spread, R_0 needs to be reduced below one. However, controlling R_0 may come at a significant social and economic cost, resulting in a challenging optimization problem to design intervention policies. An in-depth discussion of feedback control of infectious disease in the context of COVID-19 was recently reported by Stewart et al. [37].

B. SINDy-MPC

In this tutorial, we use SINDy-MPC to control the spread of an arbitrary infectious disease by directly controlling the transmission rate $\beta = \mathrm{u}(t)$. Using MPC, we can define constraints on the maximum health care capacity and tailor a control objective considering social cost and economic cost by specifying the cost function weights $\mathbf{Q}$, $\mathbf{R}_u$, and $\mathbf{R}_{\Delta u}$. The main challenge of MPC is the development of an accurate and computationally efficient model. To identify the nonlinear dynamical system describing the spread of the infectious disease, we use the SINDy with control algorithm, assuming that the dynamical system has relatively few active terms. Because these models are sparse by construction, they avoid overfitting, and are more computationally efficient than many other models, so they may be used in real-time and may be identified from relatively small amounts of training data, compared with neural networks. The model training and validation, along with the performance of the models for MPC, are shown in figure 3. For comparison, SINDy-based MPC and DMD-based MPC are both presented.

The state of our system is $\mathbf{x} = [S, E, I, R]$, and we assume that we can control the transmission rate β by specific interventions, so the control input is $\mathrm{u}(t) = \beta(t) = \beta_0 - \beta_c(t)$. The constant parameters of the true SEIR dynamics are set to $\beta_0 = 0.5$, $\gamma = 0.2$, and $k = 0.2$. First, we generate the training data with a discrete control input over 100 days. The control input is a pseudo-random binary signal (PRBS) [39], which is a deterministic signal with white-noise-like properties. This represents different potential interventions with respective impact on the transmission rate. Code 1 generates the data for the forced SEIR dynamics.

Code 1. Generating SEIR data.

```
%% Generate Data
p.b = 0.5; p.g = 0.2; p.k = 0.2; % beta0, gamma, k
n = 4; % Number of states
x0 = [0.996, 0.002, 0.002]; % Initial conditions
tspan = 0:0.1:100;
u = prbsForcing(tspan); % PRBS forcing function
[t,x]=ode45(@(t,x) SEIR(t,x,u,p),tspan,x0);

%% Compute true derivatives
for i=1:length(x)
   dx(i,:) = SEIR(0,x(i,:),u(i),p);
end

%% SEIR model
function dx = SEIR(t,x,u,p)
   S = x(1); E = x(2); I = x(3); R = x(4);
   dx = [-u*S*I; u*S*I-p.k*E; p.k*E-p.g*I; p.g*I];
end
```

After collecting the data, we run the SINDy with control algorithm to identify a sparse nonlinear model. First, we build the library of candidate functions Θ. Here, we use polynomials up to third order. We set the sparsification hyperparameter to $\lambda = 0.1$ and run the SINDy algorithm using the sequential threshold least squares approach. Code 2 runs SINDy with the functions `poolData` and `sparsifyDynamics` that are introduced in [4] and can be found on GitHub.

Code 2. Running SINDy with control.

```
%% Build library and compute sparse regression
Theta = poolData(x,n,3); % Up to 3rd order polynom
lambda = 0.1; % Sparsification hyperparameter
Xi = sparsifyDynamics(Theta,dx,lambda,n); % STLS
```

The control objective is to minimize the number of infected individuals at lowest control expenditure. Additionally, given the maximum health care system capacity (e.g. number of ICU beds), we define a constraint on the number of infected individuals. Here, the reference is $\mathbf{r} = (0,0,0,0)$ and the weight matrices are $\mathbf{Q} = \mathrm{diag}(0,0,1,0)$ and $\mathbf{R}_u =$

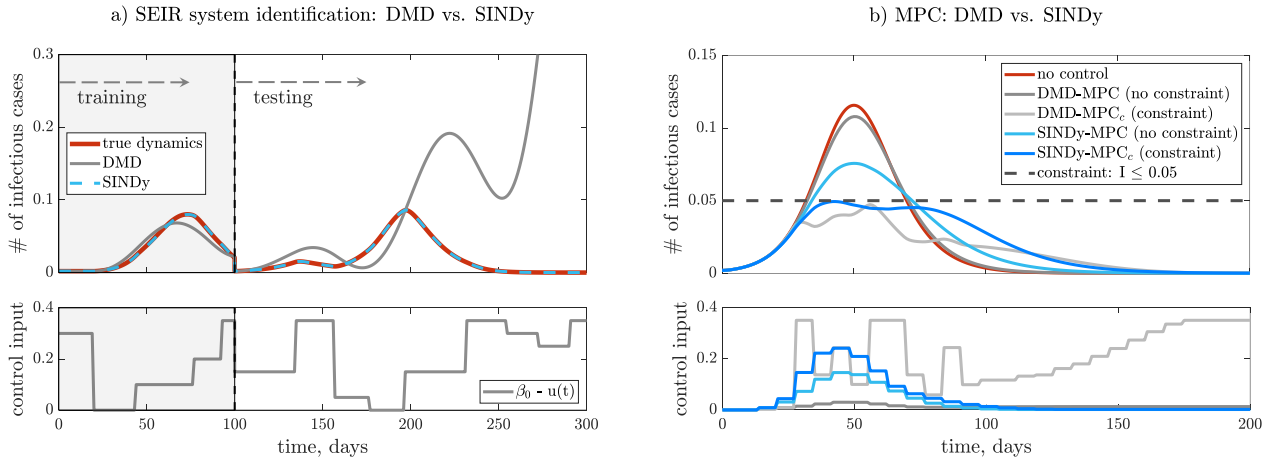

Fig. 3. SINDy-MPC vs. DMD-MPC for infectious disease control problem.

$\mathbf{R}_{\Delta u} = 0.1$. The control input is limited to $u = [0.15, 0.5]$, the control is updated once a week and the control and prediction horizon are $m_p = m_c = 14$ days. The constraint on the maximum number of infected individuals is set to 5% of the total population. Code 3 initializes the MPC problem and runs the SINDy-MPC loop.

Code 3. Initializing and running SINDy-MPC.

```
%% Initialize MPC
pMPC = MPCparams(); % define control parameters
x = x0; uopt = pMPC.uopt0;

%% Run nonlinear MPC with full-state feedback
for i = 1:(pMPC.Duration/pMPC.Ts)
   % Cost and constraint function
   COST = @(u) CostFCN(u,x,pMPC,uopt(1));
   CONS = @(u) ConstraintFCN(u,x,pMPC);
   % Optimization
   uopt = fmincon(COST,uopt,pMPC,CONS);
   % Apply control and step one timestep forward
   x = rk4u(@SEIR,x,uopt(1),pMPC.Ts,1,[],params);
   xHistory(i,:) = x;
   uHistory(i) = uopt(1);
end
```

In figure 3, the results of the MPC infectious disease intervention are shown. We use SINDy to identify nonlinear models and compare SINDy-MPC with linear DMD-MPC. On the left of figure 3 in panel a), we show the identification (training, grey background) and prediction (testing, white background) for DMD and SINDy. The control inputs are the same PRBS for SINDy and DMD, shown on the bottom. We see that the SINDy model perfectly predicts the true SEIR dynamics (we only show the infectious population for clarity). The linear DMD model is unstable, diverging after 150 days, and has low predictive performance compared to the SINDy model. We may therefore presume that the DMD model will perform poorly in MPC.

In panel b) of figure 3, we show the performance of the SINDy-MPC and compare it with the DMD-MPC. We run both methods with and without constraining the maximum number of infectious individuals (constraint: I $\leq$ 0.05, or 5% of the total population). First, we observe that the SINDy-MPC without the constraint can reduce the number of infectious cases from 11.5% to 7.5%. After adding the constraint, the SINDy-MPC$_c$ can successfully reduce the number of infectious cases below 5%. We observe that the DMD model under predicts the effect of actuation: the model suggests that further interventions would not reduce the number of cases significantly, and therefore keeps the control and interventions at a lower level. For the constrained case, DMD-MPC$_c$ is able to reduce the cases below 5%. The strategy is more uncoordinated and at higher cost compared to the SINDy-MPC$_c$. However, given the poor predictive accuracy of the DMD model, the DMD-MPC performs surprisingly well, at least over the first third of the spread of the disease. We can conclude that DMD models can be of great use, even if their predictive performance is poor. Additionally, these models may be trained with very little data. Therefore, they can be of use in the low-data limit until enough data is collected to identify an accurate SINDy model for control. As concluded in [27], SINDy-MPC provides effective and efficient nonlinear control, with DMD as a stopgap until a SINDy model can be identified.

IV. Discussion

In this tutorial, we explored the use of data-driven model discovery techniques to identify computationally tractable and accurate models of nonlinear systems for model-based control. In particular, we demonstrated how SINDy with control can be combined with MPC for infectious disease control. We have included example codes throughout to help clarify these concepts. Our goal in providing open-source code for this tutorial is to encourage the reader to test the assumptions, explore modifications, and adapt these algorithms to their own nonlinear control problems.

We have made certain assumptions to simplify the SINDy modeling and control procedure. We encourage users to implement their own modeling assumptions: changing the numerical differentiation method (assuming we can not measure the derivatives), changing the library functions, the sparse regression algorithm, or investigating different values for the sparsity-promoting hyperparameter λ (e.g. using information criteria such as AIC or BIC [40]). We also encourage the user to investigate different forcing functions and the amount of training data needed to identify models.

Forcing the dynamics with single impulses and steps, phase-shifted sum of sinusoids, or other PRBS forcing, will change the condition number of the library Θ, and hence how accurately the model may be identified from limited or noisy data. Real-world systems will inevitably have measurement noise and disturbances, which is also important to explore. It is also interesting to compare DMD-MPC and SINDy-MPC with MPC based on a neural network. Neural networks require significantly more training data and have higher execution time compared to DMD and SINDy. However, they provide a more flexible representation of the dynamics when the model structure varies in state-space [27]. To test this, the code may be modified so the SEIR parameters vary over the course of the spread of the disease. We also encourage the user to implement different control strategies (e.g. vaccination and quarantine control [41], [42]), test different initial conditions, extend the SEIR model to consider other compartments, or completely replace the SEIR dynamics with other nonlinear dynamical systems.

REFERENCES

[1] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016.

[2] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] S. L. Brunton and J. N. Kutz, *Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[5] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[6] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999.

[7] J.-N. Juang and R. S. Pappa, "An eigensystem realization algorithm for modal parameter identification and model reduction," *Journal of guidance, control, and dynamics*, vol. 8, no. 5, pp. 620–627, 1985.

[8] R. W. Brockett, "Volterra series and geometric control theory," *Automatica*, vol. 12, no. 2, pp. 167–176, 1976.

[9] S. Boyd, L. O. Chua, and C. A. Desoer, "Analytical foundations of Volterra series," *IMA Journal of Mathematical Control and Information*, vol. 1, no. 3, pp. 243–282, 1984.

[10] H. Akaike, "Fitting autoregressive models for prediction," *Ann Inst Stat Math*, vol. 21, no. 1, pp. 243–247, 1969.

[11] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[12] R. Lippmann, "An introduction to computing with neural nets," *IEEE Assp magazine*, vol. 4, no. 2, pp. 4–22, 1987.

[13] A. Draeger, S. Engell, and H. Ranke, "Model predictive control using neural networks," *IEEE Control Systems Magazine*, vol. 15, no. 5, pp. 61–66, 1995.

[14] E. Aggelogiannaki and H. Sarimveis, "Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models," *Computers & Chemical Engineering*, vol. 32, no. 6, pp. 1225–1237, 2008.

[15] T. Wang, H. Gao, and J. Qiu, "A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 416–425, 2016.

[16] I. Lenz, R. A. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control." in *Robotics: Science and Systems*. Rome, Italy, 2015.

[17] T. Baumeister, S. L. Brunton, and J. N. Kutz, "Deep learning and model predictive control for self-tuning mode-locked lasers," *JOSA B*, vol. 35, no. 3, pp. 617–626, 2018.

[18] J. Morton, F. D. Witherden, A. Jameson, and M. J. Kochenderfer, "Deep dynamical modeling and control of unsteady fluid flows," *arXiv preprint arXiv:1805.07472*, 2018.

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

[20] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz, "Deep model predictive flow control with limited sensor data and online learning," *Theoretical and Computational Fluid Dynamics*, 2020.

[21] H. Peng, J. Wu, G. Inoussa, Q. Deng, and K. Nakano, "Nonlinear system modeling and predictive control using the RBF nets-based quasi-linear ARX model," *Control Engineering Practice*, vol. 17, no. 1, 2009.

[22] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," in *IEEE Int Conf Robotics Autom*, 2016, pp. 528–535.

[23] 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.

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

[25] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. Henningson, "Spectral analysis of nonlinear flows," *Journal of Fluid Mechanics*, vol. 645, pp. 115–127, 2009.

[26] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[27] 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.

[28] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data-driven approximation of the Koopman operator: extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 6, pp. 1307–1346, 2015.

[29] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[30] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, "Modern Koopman theory for dynamical systems," *arXiv:2102.12086*, 2021.

[31] A. E. Bryson and Y.-C. Ho, *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[32] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[33] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback control theory*. Courier Corporation, 2013.

[34] R. M. May, *Infectious diseases of humans: dynamics and control*. Oxford University Press, 1991.

[35] C. Castillo-Chavez, J. X. Velasco-Hernandez, and S. Fridman, "Modeling contact structures in biology," in *Frontiers in mathematical biology*. Springer, 1994, pp. 454–491.

[36] G. Chowell, N. W. Hengartner, C. Castillo-Chavez, P. W. Fenimore, and J. M. Hyman, "The basic reproductive number of Ebola and the effects of public health measures: the cases of Congo and Uganda," *Journal of theoretical biology*, vol. 229, no. 1, pp. 119–126, 2004.

[37] G. Stewart, K. Heusden, and G. A. Dumont, "How control theory can help us control COVID-19," *IEEE Spectrum*, vol. 57, no. 6, pp. 22–29, 2020.

[38] V. Grimm, F. Mengel, and M. Schmidt, "Extensions of the SEIR model for the analysis of tailored social distancing and tracing approaches to cope with COVID-19," *Scientific Reports*, vol. 11, no. 1, 2021.

[39] L. Ljung, "System identification-theory for the user 2nd edition ptr prentice-hall," *Upper Saddle River, NJ*, 1999.

[40] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor, "Model selection for dynamical systems via sparse regression and information criteria," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2204, 2017.

[41] R. L. M. Neilan, E. Schaefer, H. Gaff, K. R. Fister, and S. Lenhart, "Modeling optimal intervention strategies for cholera," *Bulletin of mathematical biology*, vol. 72, no. 8, pp. 2004–2018, 2010.

[42] H. Boujakjian, "Modeling the spread of Ebola with SEIR and optimal control," *SIAM Undergraduate Research Online*, vol. 9, pp. 299–310, 2016.