Simultaneous Learning of Control Signals, Parameters, and Model Structure

C. Fieseler, J. N. Kutz

Data-driven methods for approximating the underlying dynamics of a complex system have emerged in many different fields of science and engineering. Many approaches posit an autonomous model for the dynamics, such that in the limit of no noise the future state of the system is predictable entirely by its past. Several established methods, such as Dynamic Mode Decomposition (DMD) and Sparse Identification of Nonlinear Dynamics (SINDy), have achieved great success in simultaneously predicting the structure of unknown dynamical systems and their parameter values in autonomous systems. However, many systems of interest, particularly in biology and neuroscience, are connected to an outside environment and thus are not autonomous, and in many cases the stimulation is completely unknown. We propose an extension of these established methods for simultaneously learning an external control signal along with model structure and parameter values. This requires first extending the methods to a Bayesian framework, and successfully separates the underlying dynamical systems and control signals even in chaotic and noisy? systems.

1. Introduction

Data-driven methods for analyzing complex high-dimensional systems have become very popular, and a goal is often to find an interpretable set of governing equations. These data-driven methods generally try to find an autonomous model for the dynamics, for example the recently developed Dynamic Mode Decomposition (DMD) which finds the best-fit linear model along with a low-dimensional set of basis functions, or Sparse Identification of Nonlinear Dynamics (SINDy), which similar but allows nonlinear terms. That is, they posit that the governing equations can be written as a differential equation of the form: $\dot{\mathbf{x}} = f(\mathbf{x})$. However, many systems of interest are connected to the outside world, have unmeasured degrees of freedom, or contain dynamics faster than can be measured. An example with a particularly large number of unmeasured variables is that of experimental neuroscience, where in general only a small portion of the brain can be measured. Thus in practice, the dynamics of the observed portion of the system are not autonomous, and appear to be forced by an outside "control signal," "forcing term," or "exogenous shock" that can be written as:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + \delta(t) + \epsilon(t) \qquad [1]$$

where $f(x)$ are intrinsic dynamics of interest, $\delta(t)$ is an unknown discrepancy, and $\epsilon(t)$ is white noise.

The original DMD algorithm (1) was proposed by Schmidt for approximating autonomous, low-dimensional spatio-temporal dynamics in high-dimensional fluids. It was subsequently used in a wide variety of application areas including computer vision (2, 3), neuroscience (4), disease modeling (5), finance (6), and fluid dynamics (1, 7–9). Control was added to produce DMD with control (DMDc) (10, 11), which has an equivalent mathematical form to Eq. 1. Taking the external control signal into account allowed for both a drastically lower number of relevant dimensions and increased accuracy of the recovered autonomous dynamics. Similar benefits were realized in adding control to nonlinear systems (12). However, these control signals must be known in advance, which is often not the case for natural systems.

One approach for simultaneously learning a model and unknown external forcing is the "discrepancy modeling" framework (13–15). This fully Bayesian approach posits a model of the same form as 1, where $\delta(t)$ is the "discrepancy" and is generally modeled as a Gaussian process. This and similar frameworks have been applied in many different real-world settings, including ecology (16), robotics (17), and control (18). Gaussian processes are very powerful in that they can model nearly any smooth function, but this contributes to a major difficulty with this framework: identifiability (19, 20). That is, unless you have many different data sets (21) or can guess the functional form of the discrepancy (22), there is no clear way to separate out what is the external signal and what is the intrinsic dynamics.

We propose a new partially Bayesian framework called Sparse Residual Analysis (SRA)??? for learning sparsely active control signals purely from data simultaneous with an interpretable model of the intrinsic dynamics, allowing for accurate reconstructions both of the underlying autonomous system and the effects of the external signals. We do this by building the posterior distribution of a set of uncontrolled models, $f(\mathbf{x})$, either linear (DMD) or nonlinear (SINDy). Sampling from the posterior and comparing to data produces an initial guess for dynamics that are outside of the initial model structure, which forms a basis for approximating $\delta(t)$. We test our method on linear, nonlinear, and chaotic systems with external control signals, successfully learning the intrinsic simple system as well as the control signals.

This paper proceeds as follows: Section II introduces the background methods from statistical and machine learning; Section III introduces our modified optimization problem and practical subtleties with the algorithm; Section IV shows the results of the method to decompose dynamics; and Section V discusses future directions. The code for this framework is freely available on GitHub in the programming language Julia, heavily using advanced features from the DifferentialEquations (23) and the Turing probabalistic programming packages (24).

2. Background Methods

Statistical Learning. As a complement to Machine Learning methods which produce a black box prediction given data, statistical learning methods seek to learn explicit and analyzable governing equations for a system, which can be linear or nonlinear.

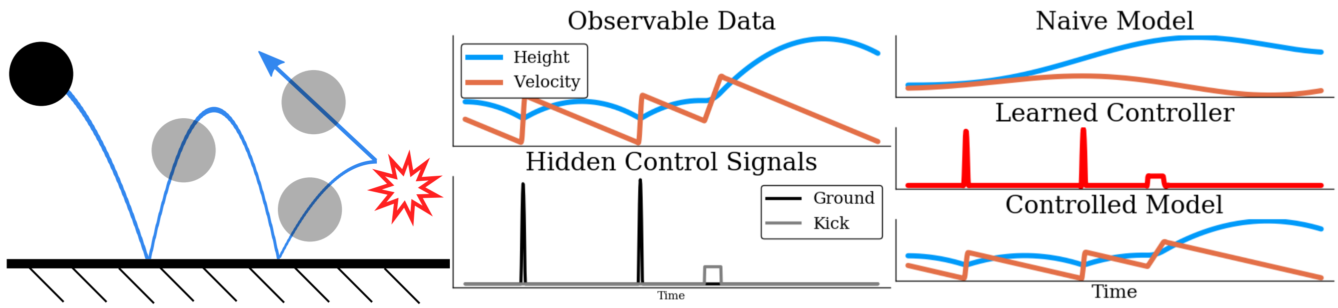

Fig. 1. a) A simple example system: a bouncing ball. The intrinsic dynamics is the acceleration due to gravity, which is linear in the velocity-acceleration basis. The external spatially-dependent forcing is provided by the ground, and in this example there is an additional time-dependent forcing, e.g. a kick. b) The observed data in this simple case have obvious discontinuities, and the two types are easy to distinguish. c) The control signal provided by the ground, which is actually a function of space, and the external kick, which is purely a function of time.

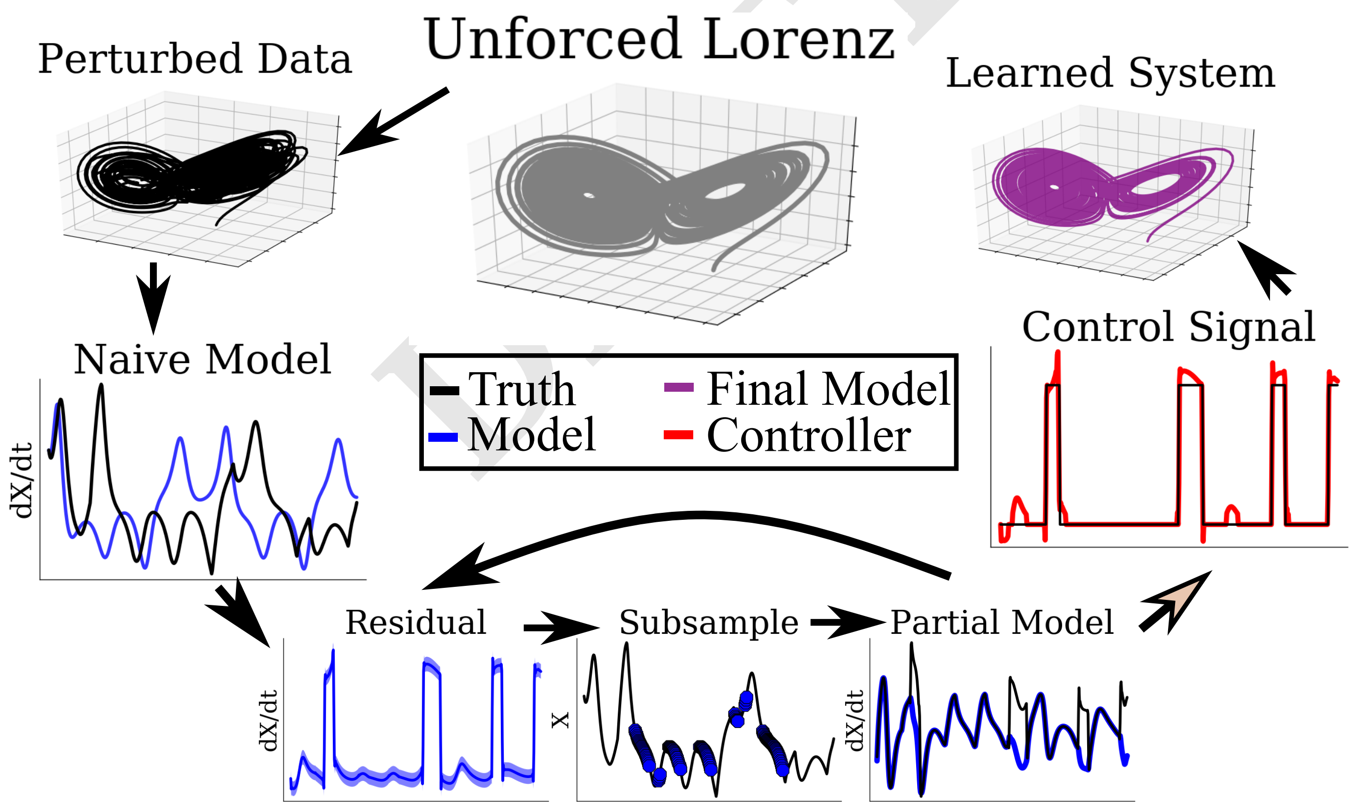

Fig. 2. Beginning with data (in this case the Lorenz attractor with time-dependent external forcing), there are six steps to the model: 1. Fit a naive ODE. When integrated, this reconstruction will be very poor. 2. Find the posterior distribution of residuals of this naive ODE to numerically calculated derivatives. Note that this uses a collocation method, not integration. 3. Subsample the data, choosing the data points with small residual in the naive model. 4. Fit a "partial" model on the smaller sample of data. The control signals will not be captured, but the intrinsic dynamics may be. If they are not fit well, then looping back to step 2 will increase the quality of the subsample. 5. Using the residual of the final "partial" model, determine the control signals. 6. Fit the full control model, using control signals and data. This example is a chaotic system, so the individual trajectory will never be well reconstructed. Rather, the goal is to reconstruct the attractor.

DMD and DMDc. DMD provides a linear model for data matrices constructed using temporal snapshots of the state space, $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_m]$ and $\dot{\mathbf{X}} = [\dot{\mathbf{x}}_1\ \dot{\mathbf{x}}_2\ \ldots\ \dot{\mathbf{x}}_m]$ where $\mathbf{x}_j = \mathbf{x}(t_j)$. Specifically, it finds the best fit linear dynamical system

$$\dot{\mathbf{X}} = \mathbf{A}\mathbf{X} \qquad [2]$$

passing through the m snapshots of the statespace. There are a number of variants for computing $\mathbf{A}$ (1, 25**?** –29), with the *exact DMD* (25) simply constructing

$$\mathbf{A} = \dot{\mathbf{X}}\mathbf{X}^{\dagger} \qquad [3]$$

where † denotes the Moore-Penrose pseudo-inverse, which is a least-squares fitting procedure. However, in practice due to the size of matrix $\mathbf{A}$ in (3), the data is first projected onto the dominant correlated modes via the singular value decomposition before an eigen-decomposition is computed (30), i.e. a low-rank approximation is first computed.

DMDc (11) leverages the advantages of DMD and provides the additional innovation of disambiguating between the underlying dynamics and the effects of a known actuation signal. For a control input matrix $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_{m-1}]$ where $\mathbf{u}_j = \mathbf{u}(t_j)$ is the actuation signal at time t_j, DMDc regresses instead to the linear control system

$$\dot{\mathbf{X}} = \mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{U} \qquad [4]$$

The DMDc method regresses to find the best matrices $\mathbf{A}$ and $\mathbf{B}$ in a least-squares sense given $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{U}$. Thus DMDc does not require knowledge of the underlying governing equations, only time snapshots of the state space and control input, making it compelling for systems whose governing equations are unknown. As with DMD, the DMDc algorithm capitalizes on underlying low-dimensional structure in the data by using the singular value decomposition to compute $\mathbf{A}$ and $\mathbf{B}$ in practice.

SINDy and SINDYc. The SINDy algorithms have a parallel structure to the DMD equations above:

$$\dot{\mathbf{X}} = \mathbf{\Phi}(\mathbf{X})\mathbf{\Xi} \qquad [5]$$

And the formulation with control is similar:

$$\dot{\mathbf{X}} = \mathbf{\Phi}(\mathbf{X})\mathbf{\Xi} + \mathbf{B}\mathbf{U} \qquad [6]$$

where $\mathbf{\Phi}(\mathbf{X})$ is a library of nonlinear functions of the original data rows and $\mathbf{\Xi}$ is the sparse matrix of coefficients. Sparsity is enforced via an L1 norm or sequential least squares thresholding (12, 31), and significant parameters are determined via information theory metrics like AIC, as described in previous work (32).

Discrepancy modeling. Many inverse problems, that is learning a model structure and parameter values from data, are ill-posed TODO. There are multiple possible sources, and a major one is systematic discrepancy between the model and data due to the data being produced by a process that does not conform to the assumptions of the model. Seminal work has demonstrated how direct modeling attempts can fail to recover accurate dynamics, but they can be recovered if a discrepancy term is added, as in Eq. 1 (13–15). This is done by positing that $\delta(t)$ can be modeled by a Gaussian Process, i.e. a smooth function.

We take an opposite approach, positing sparsely active or spike-like control signals. In this view, although the intrinsic dynamics of the system could be modeled by Eqs. 5 or 2, a direct approach will not work because these external perturbations are unknown. A key assumption is that these external perturbations are sometimes weak or entirely absent, allowing a subsampling procedure somewhat similar to (20). Our approach then does not posit an explicit function form for $\delta(t)$, but instead treats this control function as the statistically significant deviations between the intrinsic dynamics and the data, where statistically significant refers to an explicit noise model made possible due to the Bayesian framework. This method is more fully explained in the next section and in algorithm 1.

3. New method: SRA??

Residual Analysis??. Data generated from a linear process with external shocks of the form in equation 4 can be fit using an uncontrolled framework, for example via the least squares method described in 3. In this case the regression matrix, called $\hat{\mathbf{A}}$, may be very different from the true linear dynamics $\mathbf{A}$, because it is trying to account for the external input $\mathbf{B}\mathbf{U}$. However, if we knew the dynamics, it would be very easy to discover the control signals via rearranging equation 4:

$$\dot{\mathbf{X}} - \mathbf{A}\mathbf{X} = \mathbf{B}\mathbf{U} \qquad [7]$$

Certainly, we do not know the true dynamics $\mathbf{A}$, but in many circumstances (TODO) $\hat{\mathbf{A}}$ can be used to approximate this residual, and thus the control signals themselves. This extends to the nonlinear case, and if any parameters are known in advance these can be explicitly specified in this step.

Probabilistic Programming. It is possible to analyze a single residual directly from equation 7. However, the residual of any single model realization will be very sensitive to the exact training data and noise. In some cases this sensitivity can be mitigated, as shown in Fig. S3. A more statistically sound alternative is to explore an ensemble of models, producing a distribution of residuals. The presence of outliers beyond the noise envelope is then very obvious, as the noise envelope is explicitly modeled and fit. These outliers are then the initial guess for the control signal, as shown in Fig. 2.

In addition, this Bayesian extension of the original SINDy algorithm automatically produces a posterior distribution for the model parameters as shown in Fig. 4.

SRA??. The full algorithm consists of a multi-step loop and some preprocessing stages, as explained graphically in Fig. 2 and more generally in algorithm 1. Initially, the model structure must be chosen, in this case either linear, as in Eq. 2 or nonlinear as is Eq. 5. This "naive" model is fit to the derivative data. In some cases, particularly if the control signals span multiple orders of magnitude, taking a random intial subsampling can dramatically improve this naive model.

The next step is a loop that refines this initial model guess via modeling a well-selected subset of the data. This subset is the set of gradient points that is well reconstructed by the naive, uncontrolled model. "Well reconstructed" refers specifically to points whose errors are within a factor λ of the noise envelope. Note that these are reconstructions of the gradient itself via equation 1, and no integration of these

Algorithm 1 Unsupervised Learning of Controlled Model

procedure LearnControlModel($\mathbf{X}$, $\dot{\mathbf{X}}$)
  $\hat{f}_0 := FitModelDistribution(\dot{\mathbf{X}})$ ▷ Eq. 2 or 5
  for $i \leftarrow 1, MaxIter$ **do**
    $\mathbf{R}_i := \dot{\mathbf{X}} - \hat{f}_{i-1}(\dot{\mathbf{X}})$ ▷ E.g. eq. 7
    $\dot{\mathbf{X}}_{i,sub} := Subsample(\dot{\mathbf{X}}, \mathbf{R}_i, \lambda)$
    $\hat{f}_i := FitModelDistribution(\dot{\mathbf{X}}_{i,sub})$
  $U_{final} := ProcessResidual(\mathbf{R}_{final})$ ▷ Sparsify
  $Model := Fit(\dot{\mathbf{X}}, \mathbf{X}, \hat{f}_{final}, U_{final})$ ▷ eq. 1

equations is performed. Last, once these "partial" models have converged or after a maximum number of iterations, the remaining residual between the final model and the data is processed to form the final control signal.

There are three free hyperparameters or model choices in this algorithm. First, the class of models must be chosen. In this paper, either linear (DMD) or sparse nonlinear (SINDy) frameworks are chosen. Second, the threshold to use for subsampling the points using the residual and the noise envelope, λ. Third, the convergence criterion or maximum number of iterations. All examples in this paper required a single iteration.

4. Results

Separation of linear dynamics. Fig 1 gives a very simple example for a dataset with linear intrinsic dynamics that can be analyzed using this method. The governing equations are simply the action of gravity:

$$\begin{aligned} \dot{v} &= a \\ \dot{a} &= -g \end{aligned} \quad [8]$$

where the gravitational constant $g = 9.81$. However, the two sources of external disturbance completely change the long-term behavior of the system, which would naturally simply fall forever. In addition, these control signals only act on a single variable directly: the acceleration. This means that a naive linear model, one that does not account for control, will successfully model the first term but not the second. The best fit least-squares model to this data is:

$$\begin{aligned} \dot{v} &= a \\ \dot{a} &= -0.5v + 0.2a + 7.5 \end{aligned} \quad [9]$$

Extra terms appear, and "g" is incorrect. If this model is integrated as in Fig. 1.?? the reconstruction is very bad, however, this model is good enough to produce a good control signal when doing point prediction of the derivative. The residual between these point predictions and the data can be processed as shown in Fig. 2 to produce the control signals shown in Fig. 1.?? These control signals then produce a much more accurate set of intrinsic dynamics, with $g = -9.812 \pm 0.05$??

Learning nonlinear, chaotic dynamics. Many dynamical systems of interest are nonlinear, and many of these are chaotic. One classic example, which was originally designed as a simplified model of atmospheric convection (33):

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z \end{aligned} \quad [10]$$

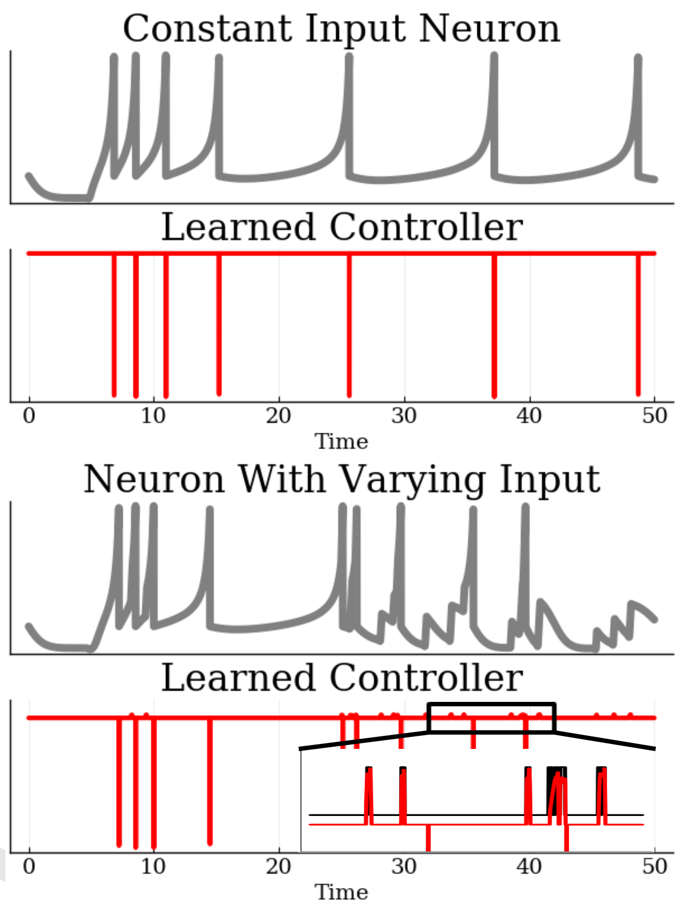

Fig. 3. a) Voltage data from a spiking neuron model. The membrane recovery variable (u) is not shown, but is provided to the algorithm. b) The "control signal" in this case is then the fast nonlinearity, instead of a truly external input. The location is learned very accurately but because it is modeled as a true discontinuity in the equations, the exact amplitude of the derivative will depend sensitively on the sampling rate and the exact method used to numerically differentiate. c) However, if a varying input current is also applied, then this will show up as an additional control signal. d) The reset nonlinearity and and external voltage are learned as a single control signal. Importantly, the learned control signals are of very different orders of magnitude.

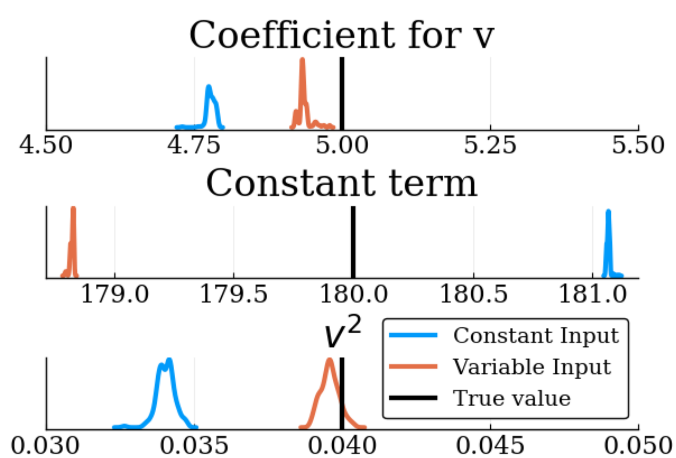

Fig. 4. Selected parameters are shown for the voltage (v) equation of Eq. 12. This algorithm successfully learns parameter values spanning four orders of magnitude with relatively small error. Note that even though there is no measurement error (white noise), some errors accumulate due to numerical differentiation.

where $\rho = 28$, $\sigma = 10$, and $\beta = 8/3$. This model can be sparsely represented with a few analytic terms, and can be recovered purely from data using the SINDy algorithm (31). An unforced version of the attractor is shown in Fig. 2.

However, if there are external perturbations of unknown magnitude, frequency, and input dimension, then the SINDy algorithm will not successfully recover the dynamics. For the perturbed version of the attractor is shown in Fig. 2, the SINDy algorithm with 2nd order library terms produces:

$$\begin{aligned} \dot{x} &= -4.6x + 6.0y - 0.4z + 20.4 \\ \dot{y} &= 23.6x + 1.1y + 18.6 - 0.9xz \\ \dot{z} &= -2.7z + 1.0xy \end{aligned} \qquad [11]$$

In this dataset, control was only applied to the first two variables (x and y), so the z equation is correct. Most of the terms are similar but one has changed signs, and several new erroneous terms have appeared. Integrating this naive model, as shown for the x variable in Fig. 2, produces very poor predictions. However, as Fig. 2 shows, as algorithm 1 is applied, the correct attractor and equations are recovered along with the control signals.

Learning nonlinear, spiking dynamics. A model discrepancy of the form in Eq. 1 may not be a true external input. In particular, it could be a nonlinearity that happens very quickly relative to data collection. One example is that of spiking neurons, in the which the "reset" after a spike is defined as instantaneous. A two dimensional model that can reproduce spiking patterns from different classes of neurons (34) is:

$$\begin{aligned} \dot{v} &= 0.04v^2 + 5v + 140 - u + I \\ \dot{u} &= a(bv - u) \\ \text{if v} \geq 30 \text{ mV, then} & \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \end{aligned} \qquad [12]$$

Where I is the input current, the constants are chosen as in the original paper to give v and t units of mV and ms, respectively. u is a membrane recovery variable, and the parameter values used here are those suggested: $a = 0.02$, $b = 0.2$, $c = -65$, $d = 2$. The input current, I, is 40 for the Constant Input neuron and is either 40 or 240 for the Variable Input case in Fig. 3.

Discrepancy modeling as a field can have a problem of identifiability (19), where the model for the discrepancy cannot be distinguished from the core model. In this framework the discrepancy can be any time series, with the assumption that it is sparsely active, thus one identifiability problem can be that of multiple control signals. As shown in 1, the control from the ground and the external forcing are learned as part of the single time series that is input onto the second variable (acceleration). A similar phenomenon happens with these spiking neurons, where the "control signal" associated with resetting voltage cannot be disentangled from modulation in external current input, I, as demonstrated in Fig. 3. Nonetheless, the governing equations are reconstructed very well, as shown in Fig. 4.

5. Discussion

Many data-driven modeling techniques posit that the dynamics present are autonomous in the chosen modeling framework. However, this assumption may be violated in many ways, two of which have been treated here: connections to the outside environment, and very fast nonlinearities. Our unsupervised modeling framework explicitly accounts for these external inputs, and successfully models the underlying intrinsic dynamics by pulling out the control signals or fast nonlinearities.

There are several assumptions that are necessary for good numerical performance. For the examples studied in this paper the intrinsic dynamics could be modeled by the imposed uncontrolled model, and in particular for the nonlinear examples, the true dynamics were sparse in the measured basis. This issue of finding the correct basis is an active field of research in this field TODO. A less well studied issue is that of the types of control signals that can be successfully separated out from "intrinsic" dynamics. This paper dealt with smooth intrinsic dynamics and effectively discontinuous control signals, which were thus separable because they could not be modeled with the uncontrolled model. However, the other extreme is more common in many fields. A popular controller in engineering and potentially in many biological systems is a PID controller, which uses terms proportional to simple functions of the state in order to acheive control. This is by design continuous, and in many cases the controlled system can be written as the uncontrolled system with a change of parameters. This algorithm assumes that the structure and parameter values of the intrinsic dynamics are unknown, and must be discovered along with the control signal. However, if there is domain knowledge of the intrinsic dynamics our algorithm may be extendable to cases with smooth controllers, but this is outside the scope of this work.

A separate limitation of this work is that a fully generative model is not produced. That is, extrapolation beyond the training data can only be achieved for the uncontrolled, intrinsic dynamics. Of course, if the control signal is truly external, then extrapolation that requires such knowledge is not possible. However, if the "control signal" is actually a function of phase space, for example the ground in the bouncing ball example and the reset discontinuity in the neuron example, a generative model is possible but is not learned. For future work, it may be possible to separate out which portions of the learned control signal can be modeled as some function of the data, and which cannot.

A universal difficulty is that of white noise, which is magnified in particular by two elements of the algorithm. First, the need to take a numerical derivative, which is known to be very sensitive to measurement noise. This is a well studied problem (**?**), but there are no universal answers. Second, because the residual of a naive model is an object of study, it is vulnerable to large fluctuations. This second issue is mitigated by a Bayesian framework for fitting the uncontrolled model. Analyzing the posterior distribution of these residuals is a much more robust procedure than analyzing a single residual, but there are still no theoretical guarantees. Recent work on simultaneous denoising and derivative calculations could be a fruitful area of future work (35).

Our algorithm adds to the landscape of data-driven algorithms that derive symbolic governing equations and extends the range of applicability into new problem domains.

1. Schmid PJ (2010) Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics* 656:5–28.
2. Grosek J, Kutz JN (2014) *Dynamic Mode Decomposition for Real-Time Background/Foreground Separation in Video. arXiv preprint, arXiv:1404.7592.*
3. Erichson NB, Brunton SL, Kutz JN (2015) Compressed dynamic mode decomposition for real-time object detection. arXiv preprint arXiv:1512.04205.

4. Brunton BW, Johnson LA, Ojemann JG, Kutz JN (2016) Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of Neuroscience Methods* 258:1–15.
5. Proctor JL, Eckhoff PA (2015) Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. *International Health* 2(7):139–145.
6. Mann J, Kutz JN (2016) Dynamic mode decomposition for financial trading strategies. *J. Quant. Finance.*
7. Bagheri S (2013) Koopman mode decomposition of the cylinder wake. *Journal of Fluid Mechanics* 726:596–623.
8. Schmid PJ (2011) Application of the dynamic mode decomposition to experimental data. *Experiments in fluids* 50(4):1123–1130.
9. Schmid PJ, Li L, Juniper M, Pust O (2011) Applications of the dynamic mode decomposition. *Theoretical and Computational Fluid Dynamics* 25(1-4):249–259.
10. Brunton SL, Brunton BW, Proctor JL, Kutz JN (2016) Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one* 11(2):e0150171.
11. Proctor JL, Brunton SL, Kutz JN (2016) Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems* 15(1):142–161.
12. Brunton SL, Proctor JL, Kutz JN (2016) Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine* 49(18):710–715.
13. Brynjarsdottir J, O'Hagan A (2014) Learning about physical parameters: The importance of model discrepancy. *Inverse problems* 30(11):114007.
14. Kennedy MC, O'Hagan A (2001) Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63(3):425–464.
15. Tarantola A (2005) *Inverse problem theory and methods for model parameter estimation*. (siam) Vol. 89.
16. Arhonditsis GB, et al. (2008) Addressing equifinality and uncertainty in eutrophication models. *Water Resources Research* 44(1).
17. Koryakovskiy I, Kudruss M, Vallery H, Babuška R, Caarls W (2018) Model-plant mismatch compensation using reinforcement learning. *IEEE Robotics and Automation Letters* 3(3):2471–2477.
18. Burns JA, Cliff EM, Farlow K (2014) Parameter estimation and model discrepancy in control systems with delays. *IFAC Proceedings Volumes* 47(3):11679–11684.
19. Arendt PD, Apley DW, Chen W (2012) Quantification of model uncertainty: Calibration, model discrepancy, and identifiability. *Journal of Mechanical Design* 134(10):100908.
20. Wutzler T (2018) Efficient treatment of model discrepancy by gaussian processes-importance for imbalanced multiple constraint inversions. *arXiv preprint arXiv:1812.07801*.
21. Arendt PD, Apley DW, Chen W, Lamb D, Gorsich D (2012) Improving identifiability in model calibration using multiple responses. *Journal of Mechanical Design* 134(10):100909.
22. Kaheman K, Kaiser E, Strom B, Kutz JN, Brunton SL (2019) Learning discrepancy models from experimental data. *arXiv preprint arXiv:1909.08574*.
23. Rackauckas C, Nie Q (2017) Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software* 5(1).
24. Ge H, Xu K, Scibior A, Ghahramani Z, , et al. (2018) The turing language for probabilistic programming in *Artificial Intelligence and Statistics*.
25. Tu JH, Rowley CW, Luchtenburg DM, Brunton SL, Kutz JN (2014) On dynamic mode decomposition: theory and applications. *Journal of Computational Dynamics* 1(2):391–421.
26. Jovanović MR, Schmid PJ, Nichols JW (2014) Sparsity-promoting dynamic mode decomposition. *Physics of Fluids* 26(2):024103.
27. Dawson ST, Hemati MS, Williams MO, Rowley CW (2016) Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids* 57(3):42.
28. Hemati MS, Rowley CW, Deem EA, Cattafesta LN (2017) De-biasing the dynamic mode decomposition for applied koopman spectral analysis of noisy datasets. *Theoretical and Computational Fluid Dynamics* 31(4):349–368.
29. Askham T, Kutz JN (2018) Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems* 17(1):380–416.
30. Kutz JN, Brunton SL, Brunton BW, Proctor JL (2016) *Dynamic mode decomposition: data-driven modeling of complex systems.* (SIAM) Vol. 149.
31. Brunton SL, Proctor JL, Kutz JN (2016) Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences* 113(15):3932–3937.
32. Mangan NM, Kutz JN, Brunton SL, Proctor JL (2017) Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473(2204):20170009.
33. Sparrow C (2012) *The Lorenz equations: bifurcations, chaos, and strange attractors*. (Springer Science & Business Media) Vol. 41.
34. Izhikevich EM (2003) Simple model of spiking neurons. *IEEE Transactions on neural networks* 14(6):1569–1572.
35. Rudy SH, Kutz JN, Brunton SL (2019) Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics* 396:483–506.