IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 18, 2022, accepted March 1, 2022. Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.3159335

A Toolkit for Data-Driven Discovery of Governing Equations in High-Noise Regimes

CHARLES B. DELAHUNT[1,2], (Member, IEEE), AND J. NATHAN KUTZ[1], (Senior Member, IEEE)
[1]Department of Applied Mathematics, University of Washington, Seattle, WA 98195, USA
[2]Global Health Labs, Bellevue, WA 98007, USA

Corresponding author: Charles B. Delahunt (delahunt@uw.edu)

The work of J. Nathan Kutz was supported in part by the National Science Foundation AI Institute in Dynamic Systems under Grant 2112085, and in part by the Air Force Office of Scientific Research under Grant FA9550-19-1-0011.

ABSTRACT We consider the data-driven discovery of governing equations from time-series data in the limit of high noise. The algorithms developed describe an extensive toolkit of methods for circumventing the deleterious effects of noise in the context of the *sparse identification of nonlinear dynamics* (SINDy) framework. We offer two primary contributions, both focused on noisy data acquired from a system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$. First, we propose, for use in high-noise settings, an extensive toolkit of critically enabling extensions for the SINDy regression method, to progressively cull functionals from an over-complete library and yield a set of sparse equations that regress to the derivate $\dot{\boldsymbol{x}}$. This toolkit includes: (regression step) weight timepoints based on estimated noise, use ensembles to estimate coefficients, and regress using FFTs; (culling step) leverage linear dependence of functionals, and restore and protect culled functionals based on Figures of Merit (FoMs). In a novel Assessment step, we define FoMs that compare model predictions to the original time-series (i.e., $\boldsymbol{x}(t)$ rather than $\dot{\boldsymbol{x}}(t)$). These innovations can extract sparse governing equations and coefficients from high-noise time-series data (e.g., 300% added noise). For example, it discovers the correct sparse libraries in the Lorenz system, with median coefficient estimate errors equal to 1%−3% (for 50% noise), 6%−8% (for 100% noise), and 23%−25% (for 300% noise). The enabling modules in the toolkit are combined into a single method, but the individual modules can be tactically applied in other equation discovery methods (SINDy or not) to improve results on high-noise data. Second, we propose a technique, applicable to any model discovery method based on $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, to assess the accuracy of a discovered model in the context of non-unique solutions due to noisy data. Currently, this non-uniqueness can obscure a discovered model's accuracy and thus a discovery method's effectiveness. We describe a technique that uses linear dependencies among functionals to transform a discovered model into an equivalent form that is closest to the *true* model, enabling more accurate assessment of a discovered model's correctness.

INDEX TERMS Data-driven discovery, noise mitigation, SINDy.

I. INTRODUCTION

The derivation of governing equations for physical systems has dominated the physical and engineering sciences for centuries. Indeed, it is the dominant paradigm for the modeling and characterization of physical processes, engendering rapid and diverse technological developments in every application area of the sciences. Since the mid 20th century, governing equations have become even more influential due to the rise of computers and scientific computing. Scientific computing allows one to emulate diverse and complex systems that are high-dimensional, multi-scale and potentially stochastic in nature. In modern times, the rapid evolution of sensor technologies and data-acquisition software/hardware, broadly defined, has opened new fields of exploration where governing equations are difficult to generate and/or produce. Biology and neuroscience, for instance, easily come to mind as application areas where first-principle derivations are difficult to achieve, yet data is now becoming abundant and of exceptional quality. Measured coarse-grained macroscopic behavior is also often difficult to derive or characterize from known microscopic descriptions. The ability to discover

The associate editor coordinating the review of this manuscript and approving it for publication was Brian Ng.

governing equations directly from time-series data is thus of paramount importance in many modern scientific and engineering settings. Confounding the discovery process is the ubiquity of noisy experimental data. The scope of this work is centered around the data-driven discovery of a system's governing equations given highly noisy experimental data.

Time-series measurements for which the signal-to-noise ratio is low represents significant challenges for any analysis of the underlying signal. While the techniques described here can apply to various discovery or signal processing methods, we consider in particular the *Sparse Identification of Nonlinear Dynamics* (SINDy) algorithm [1], [2]. SINDy is a regression method that leverages time-series data to discover the governing equations of a system of differential equations (or partial differential equations [3], [4])

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \tag{1}$$

with state $\boldsymbol{x} \in R^n$ and smooth dynamics $\boldsymbol{f}(\boldsymbol{x}) : R^n \to R^n$. SINDy assumes that the governing equation of each variable x_j is a linear combination of a small number of terms, i.e.,

$$\dot{x}_j = \sum_{f_i \in L} \xi_{ij} f_i(\boldsymbol{x}) \tag{2}$$

where $f_i(\boldsymbol{x})$ are candidate functionals in library L and the ξ_{ij}, representing their weights or loadings, are assumed to be mostly zero. Indeed, only a few non-zero terms ξ_{ij} are assumed to be relevant in discovering the parsimonious dynamical model (Eqn. 1). The basic SINDy method starts with a pre-defined, overcomplete library of functionals $F = \{f_i(\boldsymbol{x})\}$ (e.g., polynomials up to some degree), and assumes $\dot{\boldsymbol{x}}(t) = \Xi F(\boldsymbol{x}(t))$ for a coefficient matrix Ξ. It then culls coefficients until Ξ is sparse. Then for a given variable x_j the non-zero entries in the j^{th} row of Ξ correspond to the non-zero $\{\xi_{ij}\}$ in (Eqn. 2). The original formulation of SINDy estimated and culled functionals using *Sequentially Thresholded Least Squares* (STLSQ) [1], [5], as a tractable alternative to ℓ_1 regression. Other culling methods have been developed. In particular [6] used a robust method termed SR3 (sparse relaxed regularized regression) to extract the sparse, non-zero coefficients of the dynamical model.

Since its introduction, SINDy has been applied to a wide range of systems, including reduced-order models of fluid dynamics [7]–[14] and plasma dynamics [15], [16], turbulence closures [17]–[19], nonlinear optics [20], numerical integration schemes [21], discrepancy modeling [22], [23], boundary value problems [24], multiscale dynamics [25], identifying dynamics on Poincare maps [26], [27], tensor formulations [28], and systems with stochastic dynamics [29], [30]. It can also be used to jointly discovery coordinates and dynamics simultaneously [31], [32]. The integral formulation of SINDy [33] has also proven to be powerful, enabling the identification of governing equations in a weak form that averages over control volumes; this approach has recently been used to discover a hierarchy of fluid and plasma models [34]–[37]. The open source software package, PySINDy,[1] has been developed in Python to integrate the various extensions of SINDy [38], [39]. For actuated systems, SINDy has been generalized to include inputs and control [40], and these models are highly effective for model predictive control [41]. It is also possible to extend the SINDy algorithm to identify dynamics with rational function nonlinearities [42], [43], integral terms [33], and based on highly corrupt and incomplete data [6], [44]. SINDy was also recently extended to incorporate information criteria for objective model selection [45], and to identify models with hidden variables using delay coordinates [46]. The diversity of methods and applications highlight the broad reach and flexibility of the underlying regression architecture (Eqn. 2). The high-noise methods introduced here can help with all these formulations since noise severely limits the usefulness of SINDy in such a parameter regime.

A. CHALLENGES OF SINDy

This paper seeks to address common challenges for the SINDy method and its variants. These include:

1) SINDy's regressions fit the derivatives $\dot{\boldsymbol{x}}$, not the original time-series $\boldsymbol{x}$. Given noisy data, this has (at least) two effects: First, the solutions are not unique (as noted in [33]), especially given an overcomplete starting library. There can exist several plausible sparse libraries, and for a fixed sparse library a range of coefficients, each of which fit the derivatives well but give different estimates $\hat{\boldsymbol{x}}(t)$ when evolved in time. Thus, fitting the derivatives is not a sufficient method to find a sparse model which reproduces the original time-series.
2) What level of sparsity to enforce is not known, so the output of the SINDy algorithm is often too dense or too sparse and hyper-parameter tuning is required. A standard solution is to do multiple regressions, sweeping the sparsity parameter λ, then choose between the resulting models by some method, e.g., (i) expert assessment of models on a Pareto front [42]; (ii) finding the knee where error (vs $x(t)$) stabilizes [33]; or (iii) examining error of the fit to $\dot{x}$ via cross-validation [29] or on a holdout trajectory [43].
3) The progressive culling of functionals from the library is greedy, so if a vital functional is culled early the method cannot recover (observed for SINDy by [29], and for Lasso by [47]).
4) Library functionals are culled based on their coefficients' magnitudes. This penalizes functionals with large-valued $f_i(t)$. For example, if a time-series $x(t)$ hovers around the value 10, then the functional $x^2(t)$ can have a coefficient ξ_2 that is $10\times$ smaller than the coefficient ξ_1 of $x(t)$ while exerting the same effect in the regression, since $\xi_2 x^2(t) \approx \xi_1 x(t)$. Since functionals with small coefficients are culled first, $x^2(t)$ will be culled despite having equal impact

[1] https://github.com/dynamicslab/pysindy

in Eqn. 2. A gaussian prior can be imposed on the coefficients [48], [49], but this may be a poor match for natural systems, whose coefficients can have a wide range of magnitudes. Library functional time-series $f(t)$ can be normalized to unit variance [50], but this risks imposing an opposite bias on coefficient size, where $x^2(t)$ has equal footing with $x(t)$, even though its actual coefficient would be $10\times$ smaller, and in noisy conditions more volatile.

5) Noisy data is especially problematic because both the regression target $\dot{\boldsymbol{x}}$ and the library functionals' time-series $f_i(t)$ are estimated from this data. Existing approaches to handling noise are discussed in the next section.

B. DEALING WITH HIGH NOISE IN THE SINDy CONTEXT

High noise is the central challenge addressed in this paper. Noisy data disrupt SINDy primarily by compromising derivative estimates and by distorting estimates of library functional values $f_i(\boldsymbol{x}(t))$. The original SINDy formulation [1] handled 12% gaussian noise ($\sigma_{noise}/\sigma_{data} = 0.12$) in the Lorenz system using total variation regularization of derivative estimates, with some error in the final estimated coefficients Ξ. However, given higher noise levels the method fails because the estimated derivatives become too noisy. The special case of partial corruption (i.e., some timepoints contain noise, whereas others are noise-free) has also been considered [6], [44]. By trimming outliers, Champion *et al.* [6] handled 10% of datapoints corrupted with heavy noise (spreading to 30% of derivative values corrupted) in the Lorenz system. The theoretical bounds of the least median of squares method [51]) suggests that higher levels of corruption might be addressable with this approach. Tran *et al.* [44] recovered correct equations and coefficients for the Lorenz system given up to 72% of points corrupted, subject to (i) low levels of noise; (ii) sufficiently long time-series; and (iii) a corruption pattern alternating long clean segments with corrupted segments. Lejarza and Baldea [52] handled 7% added noise in the Lorenz system by culling functionals based on consistency of their coefficient estimates (instead of their magnitude as in SINDy), and by optimizing a fit to x rather than $\dot{x}$, though with an Euler step-forward constraint approximating the derivative (for more detail, see V-A2).

Rudy *et al.* [53] assumes exact knowledge of the governing sparse library but not its coefficients, and then includes a loss term measuring whether the current model satisfies the implied dynamics. This method identified correct coefficients for the Lorenz system given roughly 120% noise (including a non-zero mean case), suggesting that it might serve as an effective second stage in a chain where the first stage identifies the correct sparse library but not the correct coefficients.

Noise in partial differential equations (PDEs) is especially challenging for SINDy, because noise amplifies with each (spatial) derivative calculation [3], [34]. Rudy *et al.* [3] found that even low noise resulted in much decreased coefficient estimates (e.g., magnitude roughly halved) of discovered sparse library functionals. Ahnert and Abel [54] note that a simple moving average of noisy data always reduces estimates of extrema, which would reduce the functional coefficients ξ_{ij} in Eqn. 2. Schaeffer and McCalla [33] address noise via integration, so that the sparse regression involves not the noisy $\dot{x}$ and $f(t)$s but rather their much cleaner integrated versions. This weak formulation has high potential value for PDEs, where noise amplifies with each partial derivative but can be effectively mitigated by integration. It handled 3% noise in the Lorenz system, but this result may understate the value of this method for PDEs. Reinbold *et al.* [34] also applies integration, via multiplication by a smooth kernel along with integration by parts, to reduce effects of noise. This method handled 5% noise in a reaction-diffusion equation, and usually recovered the correct sparse library elements (with coefficient errors) even at 10% noise.

C. A TOOLKIT FOR NOISY DATA

This paper considers the case of added white noise equivalent to 50% to 300% gaussian noise, affecting all data points. See Fig. 1 for examples. The experiments here are restricted to systems that can be described by polynomial libraries, including Lorenz, harmonic oscillators, and the Hopf Normal form (as in [1]). Systems of PDEs, rational functions, or control, are potential targets for this toolkit but have not yet been addressed. We apply an engineering lens to SINDy to address the exigencies of noisy data, and describe a toolkit of novel, practically-based techniques, including:

In the Regression step, we weight timepoints based on estimated noise, use ensembles to estimate coefficients, and regress using the Fast Fourier Transforms (FFTs) of the derivatives and library functionals. In the Culling step we rescale coefficients, leverage linear dependence of functionals, and restore and protect culled functionals based on Figures of Merit (FoMs). In a novel Assessment step we define FoMs that compare model predictions to the original time-series (i.e., to $\boldsymbol{x}(t)$ rather than $\dot{\boldsymbol{x}}(t)$).

We emphasize that the individual techniques can operate separately, and are intended to be incorporated tactically into other frameworks, including but not restricted to SINDy. Here we present the toolkit combined into a single architecture, and it can be used as such; but it is really several independent modules strung together to produce an effective overall architecture for model discovery in the high noise limit. As with SINDy, we wish to discover a sparse set of governing equations by fitting functionals to the derivatives of the system. We start with an overcomplete library of functionals, and progressively cull functionals with small magnitude coefficients via STLSQ, until we achieve a sparse library of functionals that accurately model the system dynamics. The several differences from existing SINDy programs are described in our methods.

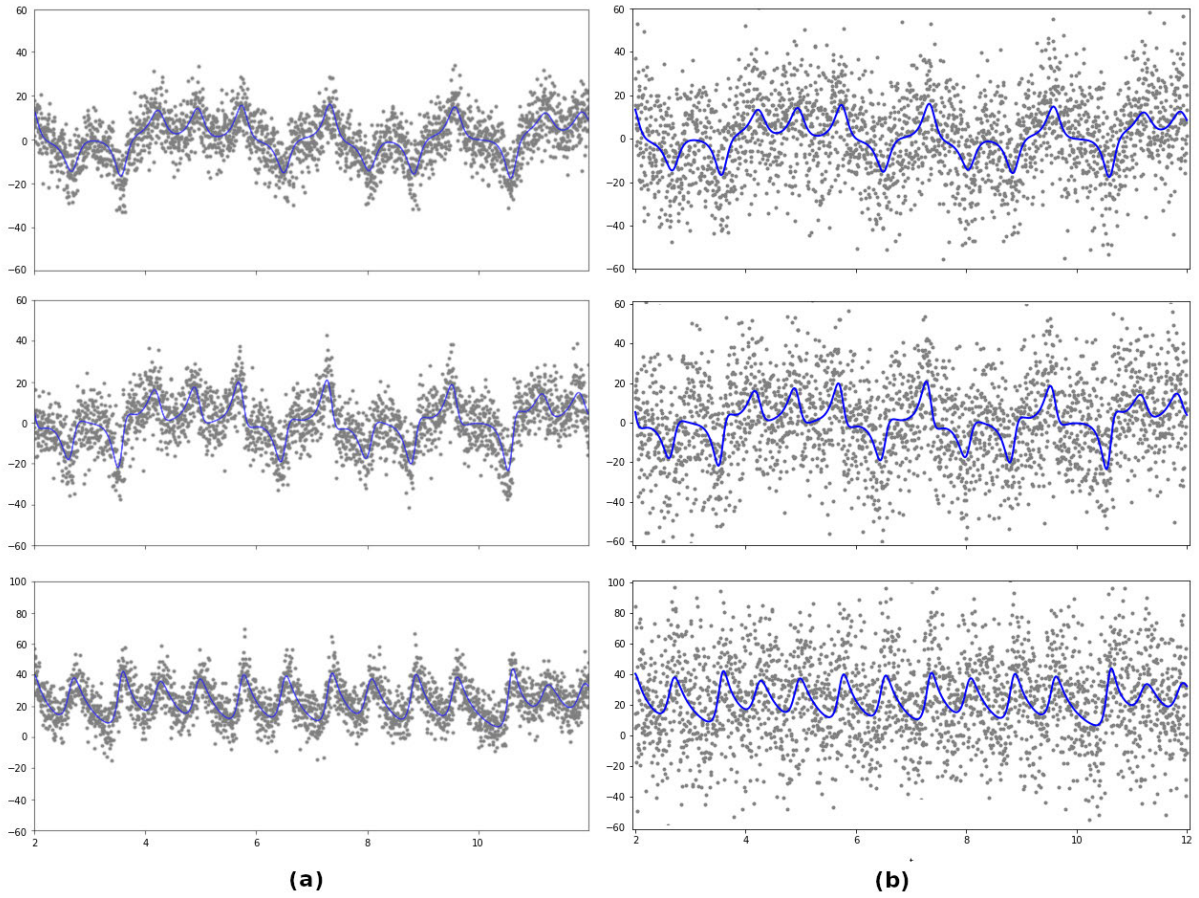


FIGURE 1. The noisy data regime. Lorenz time-series (from top, $x(t)$, $y(t)$, $z(t)$) with added white noise. Blue lines are true, clean trajectories. The effective added noise levels, $\sigma_{noise}/\sigma_{data}$, are (a) ≈100% and (b) ≈300 %. The toolkit described here discovers the correct functionals for each variable, with median coefficient estimate errors equal to 6% to 8% (for 100% noise); and 23% to 25% (for 300% noise). See Results for details.

D. CONTRIBUTIONS OF THIS PAPER

We offer two main contributions, both applicable to discovery methods generally, not just to SINDy. First, we describe a toolkit of techniques that enable accurate discovery of sparse governing equations in very high-noise settings (50 - 300% added noise). The various modules and ideas in the toolkit can be separately inserted as needed into other discovery frameworks to improve their performance in high-noise regimes. Second, we address the problem of non-uniqueness of solutions found from high-noise data. A discovered model can appear to have incorrect functionals and/or coefficients, though it is in fact equivalent (transforming by linear dependencies) to a form that is close to the ''true'' model [55]. However, no method currently exists (to our knowledge) to do this. We propose an automated technique to linearly transform a discovered model into an equivalent (linearly dependent) form, subject to the constraints of the input data. This technique has two uses: First, to list alternative viable forms of a discovered model as an aid to domain experts; and second, to evaluate whether a discovered model is accurate when the true system equations are known (i.e., oracle assessment of a discovery method's accuracy on a test case).

E. ROLE OF DOMAIN EXPERTISE

While this toolkit can be run ''plug and play'', we view it (and data-driven discovery methods in general) primarily as an aid to domain experts, to allow them to identify promising sets of discovered governing equations. Domain expertise plays a central role in SINDy (and other) methods, for example to choose coordinate systems and initial functional libraries [1], [56], [57], although [55] automates this to some extent; or to choose the best among several generated models [33], [42]. We posit that the need for domain expertise is inevitable and is not to be avoided or even minimized (except perhaps in control use-cases). For example, apparently automated methods of choosing a ''best model'' (e.g., AIC), while principled, necessarily assume some error function. This error function may be highly task-dependent, with generic error functions unsuitable for a given domain. Besides expert choice of coordinate systems and initial libraries, we call on the user to select an optimal model (out of a sequence of models) based on examination of FoM plots, train and validation trajectory predictions, and possibly properties of the functionals. While any user can interpret the FoM plots, domain expertise presumably leads to more informed choices.

F. EVALUATION OF RESULTS

Clear, universal metrics of success are perhaps not possible in the context of discovery of governing equations from noisy data. First, solutions in noisy settings are non-unique (cf section I-A), and require special assessment methods, which we introduce and describe below. Second, different use-cases have diverse needs. At least three desiderata, of increasing difficulty, are used in the literature:

1) We wish to identify the correct sparse library, i.e., the non-zero functionals in the governing equations. Noise complicates this task because linear dependencies between the functionals in an over-complete library, i.e., $f_k(t) \approx \sum_{i \neq k} \beta_i f_i(t)$ for most t, mean that multiple sets of functionals can represent the system within a noise-induced margin of error (we use β_is for these intra-library dependencies to distinguish them from the sparse dynamical models of Eqn. 2).
2) We wish to accurately estimate the coefficients of the functionals. The presence of noise complicates this in various ways: (i) Regressions on different subsets of points will yield different coefficients, even for the same set of functionals, as noise distorts the trajectory $\boldsymbol{x}(t)$ and the derivatives $\dot{\boldsymbol{x}}(t)$ being fitted. (ii) Any smoothing used to de-noise the data tends to add artifacts and also reduce the sharper transitions (e.g., remove peaks in time-series) which reduces derivatives' apparent magnitudes.
3) We wish to accurately predict trajectories in different parts of the state space. In non-linear or chaotic systems, small differences in model coefficients may yield predicted trajectories $\dot{\boldsymbol{x}}(t)$ which diverge from clean ground-truth even while they live on the same attractor and exhibit the same qualitative behaviors (e.g., attractors, cycles, magnitudes, dominant frequencies, state-space histograms).

We emphasize that solutions are not unique for noisy systems, due to over-complete libraries and quasi-linear dependence of functionals within the noise envelope. Let the "true" functionals and coefficients be those given for a known test system. Various true functionals can be absent from the discovered model but still lie within the linear span of the discovered (perhaps non-true) functionals. Also, the true functionals may themselves have linear dependencies, so that their coefficients can vary noticeably with only slight effect on trajectory behavior. Thus, a discovered solution for the system can appear quite wrong for several reasons: It may contain non-true functionals; lack some of the true functionals; or contain true functionals but with non-true coefficients.

However, it may in fact be an accurate solution: First, it may produce correct trajectories, either qualitatively or quantitatively (item 3). Second, it may have a linear transformation equivalent that is close in form to the true system in various ways: (i) a discovered non-true functional may be in the linear span of the true functionals and can thus be substituted out (item 1); (ii) a missing true functional may be in the linear span of the discovered functionals and can be substituted in (item 1); and (iii) the true functionals may have internal linear dependencies that allow their coefficients to be transformed to more closely match the true form (item 2). The technique in section II-G finds an equivalent model within the linear span of the discovered model that is closest to the true solution (in oracle settings, where this is known), which improves the accuracy of items (1) and (2) above.

We also note that different use-cases require different definitions of success. For a domain expert seeking to use experimental data to gain insight into a system under study, items (1) and partially (2) might suffice. To simulate systems for experimental study, items (2) and partially (3) matter. For prediction and control, item (3) is important. In this paper we report quantitative results for (1) and (2), and qualitative results for (3). Re item (3), we found that in high-noise chaotic systems the predicted trajectories of discovered models, even with accurate sparse libraries and coefficients, tended to fall off of (and onto) the true trajectories, while maintaining very similar qualitative behavior.

II. METHODS

This section describes the high-noise toolkit, as follows: We first briefly walk through how the framework processes a dataset, listing the several modules. We then describe in detail the individual modules, grouped according to context. Miscellaneous other modules are described in the Appendix, because they either (i) gave benefit but require an existing SINDy method (e.g., the pySINDy Python package [38]); or (ii) showed no clear benefit.

A full Python codebase for the toolkit, with detailed commenting, can be found at [58]. The toolkit is also in process of being added to the updated pySINDy package [39].

A. TOOLKIT WALK-THROUGH

The toolkit has three main stages, each with multiple steps: (1) Preparation; (2) Iterations; (3) Final tasks. A schematic is shown in Fig. 2. In the list below, standard SINDy steps are marked with (∗∗), other steps reference the relevant subsection. Given time-series of state variables $x_j(t)$, $j \in \{1 \ldots n\}$, the procedure works as follows:

1) PREPARATION

Done for each variable x_j separately.

1) **Smooth** each x_j with a Hamming filter (or other method) (II-B)
2) Define a library of functionals $\{f_i\}$, possibly different for each x_j (∗∗)
3) Calculate each $\dot{x}_j$ (e.g.via Runge-Kutta) and each f_i, using the smoothed x_js (∗∗)
4) Calculate base weights for the timepoints (for regressions) (II-C1)
5) Calculate *Fast Fourier Transform* (FFT) of each $\dot{x}_j$ (for regressions) (II-C3)
6) Calculate median values of each f_i (for rescaling coefficients during culling) (II-E3)

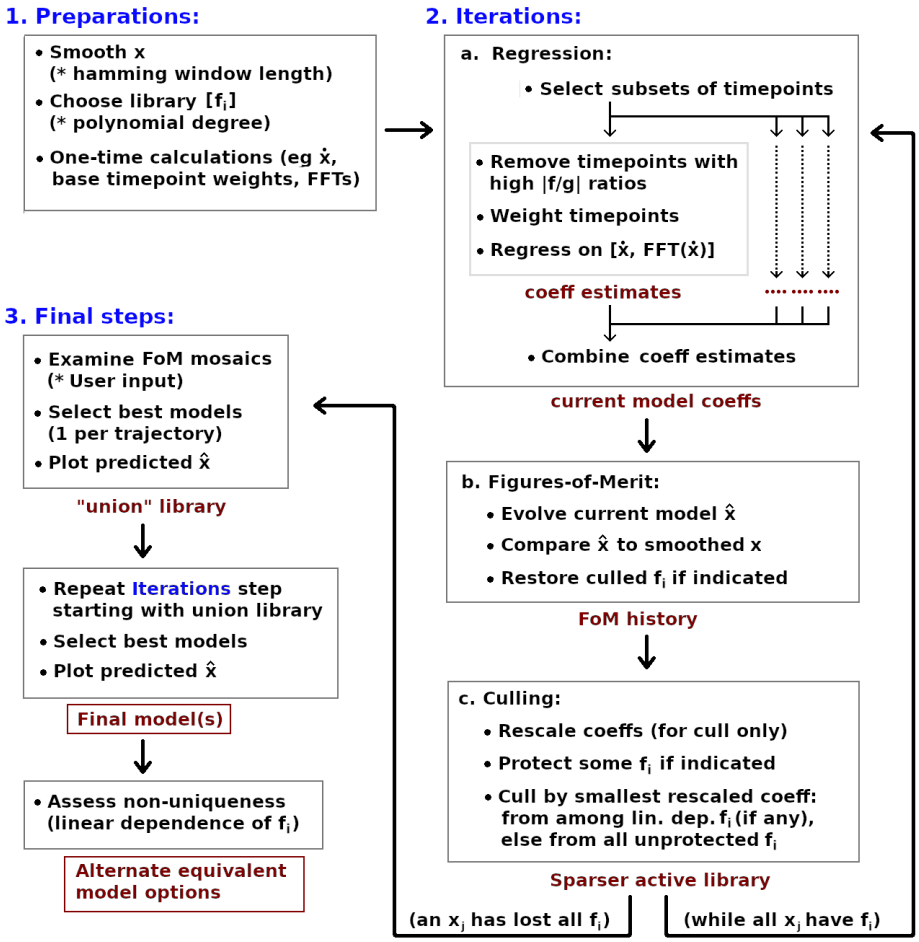


FIGURE 2. Schematic of procedure, organized as in Methods. Red font shows outputs from modules. Asterisks (*) show important parameters or User Input. The condition given to end Iterations assumes there are no noise variables (see Section II-F1).

7) Calculate histograms and FFT power spectra for each x_j (for Figures of Merit).

2) ITERATIONS:

Each iteration includes three key stages: (i) Regression, (ii) Figures of Merit, and (iii) Culling. Some steps of Regression are done for each variable x_j separately. Figures of Merit, Culling, and some steps of Regression are done on the full model (all variables together). Below, "active functionals" correspond to $\{i, j\}$ pairs where $\xi_{ij} \neq 0$ (so if $\xi_{i1}, \xi_{i2} \neq 0$, then f_i for x_1 and f_i for x_2 are distinct active functionals).

a: REGRESSION

1) **Choose *b* subsets of timepoints for regression** (e.g., $b = 15$) (II-C4). For each subset S:
2) For each x_j: **Weight the timepoints during regressions.** Use the timepoint weights based on estimated noise during regressions (II-C1)
3) For each x_j: **Remove timepoints with too-large ratios** of active f_i values from each S, since these can destablize the regression onto $\dot{x}_j$ (II-C2)
4) For each x_j: **Regress onto FFTs.** Use a target consisting of both $\dot{x}_j(t)$ and the FFT of $\dot{x}(t)$ for linear regression to find coefficients ξ_{ij}. Regression onto $x_j(t)$ is standard (**); regression onto FFTs is not (II-C3)
5) For each x_j: **Combine the ξ_{ij} estimates.** For each active f_i, there are b estimates for its coefficient ξ_{ij}, one from each subset S. Set the estimate of ξ_{ij} equal to their median (II-C4)

b: FIGURES-OF-MERIT (FoMs)

We calculate FoMs at each iteration, to help assess which model in the progressively-sparser sequence yields the best estimated time-series $\hat{x}_j$ (distinct from the fit to derivatives $\dot{x}_j$).

6) **Evolve the current model** $\{\hat{\dot{x}}_j = \sum_i \xi_{ij} f_i, \forall j\}$ over train and validation trajectories (II-D1).
7) **Calculate FoMs**, based on the evolutions (II-D2)
8) **Restoration.** A drop in certain key FoMs may signal that the most recent cull degraded the model. This triggers restoration of the culled functional, as follows: (i) restore the culled functional to the active library;

(ii) add a flag to protect it from culling for the next few iterations.

c: CULL A FUNCTIONAL

"Culling a functional" f_i for x_j means removing it from the active library of x_j, i.e., set $\xi_{ij} = 0$. This stage operates on all active functionals combined, i.e., on the matrix Ξ.

9) **Rescale the model coefficients** ξ_{ij} according to the magnitude of their associated functionals f_i. The rescaled coefficients are only used to decide which functional(s) to cull (II-E3)
10) For each x_j: **To cull, first use linear dependence** among the active functionals (i.e., those f_i with $\xi_{ij} \neq 0$). Run leave-one-out ℓ_2 linear regression on the time-series of active f_i to get R^2 values. If some $\{f_i\}$ are dependent (i.e., R^2 values above some threshold, e.g., 0.95), cull the one with the lowest rescaled ξ_{ij}, then skip the rest of culling for this iteration (II-E2). If there are no linear dependence culls:
11) **Exclude some functionals from culling**. Active functionals (i.e., with non-zero ξ_{ij}) can be excluded from culling for two reasons: (i) If they are protected due to recent restoration (II-E5); (ii) We optionally impose a **balance constraint** on the active functional counts of each variable x_j, so that functionals are not lopsidedly culled from only one variable. Exclude all ξ_{*j} for any variable x_j whose active functional count is low enough to violate the balance constraint. (II-E4)
12) **Use rescaled** $\{\xi_{ij}\}$ **to cull functionals**, one per iteration (II-E3), (II-E1)
13) Save results of this iteration to file.

Repeat iterations until all functionals are removed for some variable x_j, i.e., some row of the coefficient matrix Ξ is zeroed out. Then:

3) FINAL STEPS

1) (Optional) **Restart iterations** on the remaining variables, with new libraries, if noise variables are suspected (II-F1)
2) **Choose the likely best model for each training trajectory** by consulting the FoM sequences (and possibly the text culling history) (II-F2)
3) **Create a new library by taking the union** of the active x_j libraries of each training trajectory's best model, i.e., $L = \{f_i : \xi_{ij} \neq 0 \text{ for some } j\}$ (II-F3)
4) **Repeat the full procedure** starting with this new library
5) Print new FoM mosaics, and choose an optimal model(s) (II-F2)
6) **Assess non-uniqueness of solutions**: For each x_j, check (i) if certain functionals in the final sparse library are optional; and (ii) if there are alternative candidate functionals in the span of the final sparse library. (II-F4)
7) Run the chosen optimal model(s) on test trajectories (**)
8) End of program.

The next several subsections describe the various modules in more detail. We suppose we have multiple trajectories of a dynamical system for training. If we have only one trajectory, the Figures of Merit based on validation trajectories cannot be used, but all other methods hold.

B. SMOOTHING

We smooth the noisy trajectory with a Low Pass filter (e.g., Hamming window). De-noising always requires parameter choices, and introduces artifacts. In this case the parameter is Hamming window length, and the artifacts are squiggles due to noisy points within the window that distort the estimated trajectory (see Fig. 3). These squiggles derail the fitting of standard SINDy models by distorting $\dot{x}_j$ and f_i estimates. With our toolkit, by contrast, the squiggles introduced by smoothing are rendered relatively harmless for three reasons: (i) FoMs are relative to properties of the time-series x_j (not the derivatives $\dot{x}_j$), and sparse models yield better FoMs due to their generalizing ability (II-D2); (ii) Weighting timepoints according to their estimated noise reduces the impact of large squiggles (II-C1); (iii) Regressing on the FFT coefficients of the derivatives tends to neutralize the artifacts, since the high frequency coefficients introduced by the squiggles have low magnitude and are thus sacrificed during ℓ_2 optimization (II-C3). The particular Hamming window length is not critical as long as it is not much too big.

We note that other noise reduction methods could be used (in addition or instead) at this point, as long as they yielded continuous time-series $x_j(t)$, to accommodate different noise types. Uniform noise responds well to the Hamming filter, and the corruption scenarios in [6], [44], which contain both clean and noisy timepoints, are arguably easier tasks than full gaussian noise. Impulse or shot noise would, we expect, respond to a two-step approach (median filter followed by a linear filter) or to the methods in [6]. Periodic noise would respond to Fourier domain filtering. However, asymmetrically-distributed noise types with non-zero mean (e.g., Rayleigh, Gamma), or even non-centered gaussian, cause linear filters (such as Hamming windows) to introduce bias and would require other treatments.

C. REGRESSION (ESTIMATING COEFFICIENTS OF ACTIVE FUNCTIONALS)

Standard SINDy uses linear regression on the derivatives $\dot{x}_j$ of the system to estimate coefficients ξ_{ij} of the functionals f_i. Our toolkit offers several modifications to this basic program.

1) WEIGHTING TIMEPOINTS FOR THE TARGET VECTOR y

All timepoints are weighted according to the z-scores of their time-series values $x_j(t)$, as estimated from local noise envelopes, in order to downweight regression targets that are based on very noisy point estimates. Each variable x_j has a different vector of timepoint weights. Weights w_{tj} are assigned to timepoint t for x_j as follows:

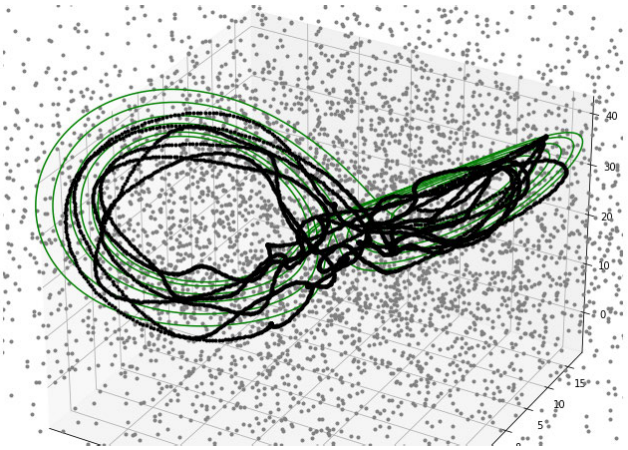


FIGURE 3. Squiggle artifacts introduced by low pass smoothing. Lorenz trajectory with 150 - 200% added noise. Green line= true clean trajectory, gray dots = noisy data, black line = smoothed trajectory. Applying a low-pass filter (e.g., convolving with a Hamming window) creates squiggle artifacts in the smoothed trajectory. These tend to derail standard SINDy methods but their effect is mitigated by other modules in the toolkit (using FoMs, weighting timepoints, and regressing on FFTs).

1) Short windows of the time-series $x_j(t-n)\dots x_j(t+n)$ are affine-transformed to have mean 0 (method: fit a line, then rotate and translate the points such that the line maps to the x-axis)
2) The transformed values define a distribution
3) Interim weights $w_{interim}$ are defined as the inverses of the Mahalanobis distances (i.e., z-scores) of the transformed values, so that high z-scores correspond to low weights
4) Log-scale the timepoint weights, $w = \ln(w_{interim} + 1)$
5) The weight w_t for timepoint t is a combination of the weights w_{sj} of the timepoints s used to generate $\dot{x_j}(t)$ (e.g., the four timepoints $\{t-2, t-1, t+1, t+2\}$ in a Runge-Kutta approximation)

This timepoint weighting has (we believe) the effect of favoring regression targets $\dot{x_j}(t)$ that depend on timepoints with low noise, which coincide with the sections of the smoothed trajectory with less squiggle.

2) IGNORE TIMEPOINTS WITH EXTREME POINT VALUES

A second form of timepoint weighting (in fact exclusion), overlaid on the $\boldsymbol{w_j}$ above at each iteration, is based on the notion that sometimes the ratio between the values of library functionals is so great that a regression will be unstable. For example, suppose one functional $f_1(t) \approx 10 \pm 2$, and a second functional $f_2(t) \approx 10sin(t)$. Most of the time the ratio $\frac{f_1}{f_2}$ is not extreme, but it blows up at f_2's zero crossings. A regression using timepoints near these zero crossings will likely result in much different, and more volatile, regression coefficients than a regression that uses only timepoints with reasonably bounded $\frac{f_1}{f_2}$. For this exclusion method, the ratios of an x_j's active (i.e., not yet culled) functionals are calculated, and any timepoints with maximum ratios above some threshold (e.g., 30, exact value not relevant) are removed ($w_{tj} = 0$) from the regression target of that x_j. This method serves to exclude the most pathological regions from the regression.

3) REGRESS ON THE FFT OF $\dot{x}$ AS WELL AS ON $\dot{x}$

SINDy typically uses estimates of $\dot{x_j}$ as a regression target. To these targets we add the coefficients associated with the derivatives' FFT. If we regress on the complex FFT coefficients, this is in theory equivalent to regressing on $\dot{x_j}$ itself. Alternatively, we can transgress mathematical correctness and use just the real part of the coefficients, the coefficient magnitudes, or the coefficients of the FFT power spectrum (this latter is in fact our default). Though none of these are strictly correct, they all serve to focus the regression on matching a characteristic of the derivatives distinct from the derivative values themselves.

In particular, the smoothed trajectory contains (i) lower frequencies associated with the trajectory's true behavior; and (ii) higher frequency artifacts (squiggles) created by the moving window filter. We suspect that regression on the FFT power spectrum coefficients knocks out the high frequency artifacts because ℓ_2 optimization minimizes its loss by preferentially failing on small-magnitude coefficients in order to match large-magnitude coefficients. This pushes the regression to ignore the artifacts, reducing their harmful effect.

4) DO MULTIPLE FITS ON SUBSETS OF TIMEPOINTS (BOOSTING)

Especially in a noisy setting, the coefficient ξ_{ij} for f_i from regression on an $\dot{x_j}$ target vector is only a single draw from a distribution of possible coefficient values. In a greedy setting like STLSQ, an aberrant ξ_{ij} can have catastrophic effects downstream if it leads to the cull of an important functional. It also can cause inaccurate evolutions of train and validation trajectories, and thus inaccurate FoMs. Thus, at each regression we wish to estimate the distribution of possible ξ_{ij}s and choose the most representative option. [48] developed a (computationally expensive) Bayesian method to estimate the distribution of each ξ_{ij}. Here we apply the simpler and cheaper method of boosting. We draw b (e.g., 15) random subsets S (alternatively, sequential sections) of timepoints and fit a model to each S. This yields a set of b ξ_{ij}s for each functional f_i (and x_j), from which to choose a suitable ξ_{ij} (we take the median). This method effectively controls the risk of aberrant ξs at low cost. This is ensembling at a local level (per iteration). In concurrent, independent work, [59] examines a variety of ensembling techniques for mitigating noise, applied to data and libraries at the model level.

D. FIGURES-OF-MERIT BASED ON MODEL EVOLUTIONS

Standard SINDy judges a model based solely according to how well it fits $\dot{\boldsymbol{x}}$, i.e., each $\dot{x_j}$. However, a good fit to $\dot{x_j}$ is usually a given and does not distinguish good models from bad. Thus we wish to judge models according to whether their evolved time-series matches the given x_js. To do this, we generate Figures of Merit (FoMs), which are various

statistics that compare the predicted $\hat{x}_j(t)$ to the smoothed trajectories $x_j(t)$. If we have multiple training trajectories $\boldsymbol{x}^k$, we create multiple models, where model m^k is trained on $\boldsymbol{x}^k$ (its home trajectory) and the other trajectories $\boldsymbol{x}^{h\neq k}$ act as validation trajectories. To generate FoMs for m^k, we evolve it on both its home and validation trajectories.

1) EVOLVE THE CURRENT MODEL

At each iteration, we use an ODE solver that takes initial conditions (e.g., Python Scipy's solve_ivp or odeint methods [60]) to generate a prediction $\hat{\boldsymbol{x}}(t)$ for the home and for each validation trajectory. Initial conditions for the evolution are found via weighted averaging of a small neighborhood of points around the nominal starting point, with more trustworthy points weighted heavier (cf section II-C1). This matters in chaotic systems where small changes in initial condition can strongly affect the evolution.

We note that these evolutions are the most computationally costly step of the toolkit, especially if a particular model has points of stiffness, or if we are tracking model stability (since we need to do multiple evolutions to see if these diverge). This cost can be mitigated in three ways: (i) By parallelization, since the different train-validation splits can run in parallel, and evolutions of a particular model at each iteration can also run in parallel; (ii) By doing only one evolution, if stability of a model is not in doubt; (iii) By skipping evolutions in early iterations when active libraries are still large and FoMs tend to be uninformative.

2) FIGURE-OF-MERIT HISTORIES

For each training trajectory $\boldsymbol{x}^k$, iterative culling of functionals gives a sequence of progressively sparser models m^k. FoMs are recorded for each model in the sequence. These FoMs are then plotted vs iteration number as shown in Fig. 4, allowing the user to select which among the models best matches the desired time-series behavior. Pan *et al.* [61] employ a similar manual examination of an FoM followed by refitting, to downselect from a library of eigenvectors in the context of Koopman operators. This method (do one complete STLSQ run and record the models at each step) is similar to how Lasso is done, and is an alternative to sweeping the sparsity parameter λ with multiple complete SINDy runs. The model at each iteration is also printed to text file for later inspection.

Examples of FoMs include:

1) Whether multiple evolutions match each other. This assesses stability of the model.
2) Fraction of evolved trajectory within upper and lower bounds (of the smoothed trajectory). This detects egregious blow-ups.
3) Fraction of evolved trajectory within an envelope about the smoothed trajectory. This is a valuable, finer measure of whether the evolved trajectory tracks the true.
4) Relative error of the std dev of evolved trajectory $v(t)$ vs smoothed trajectory $x(t)$, $\frac{\sigma(v)-\sigma(x)}{\sigma(x)}$. This comparison of std devs is a valuable measure, with good models having relative error close to 0 in all variables.
5) FFT power spectrum correlation between evolved and smoothed trajectories.
6) Histogram correlation between evolved and smoothed trajectories.

Two FoMs ("fraction in-bounds" and "evolution correlation") readily detect big failures, i.e., exploding trajectories and unstable models. "Relative error of std dev", "fraction in-envelope" and "histogram correlation" detect cases when a reasonably accurate model starts going moderately off-track. FoMs about derivative estimates are not useful, because even bad models can match the derivatives well. As is usual in machine learning, strong FoMs on validation trajectories often indicate a good model (see Fig. 4).

E. CULLING FUNCTIONALS

STLSQ involves culling functionals with the lowest non-zero coefficients in the estimate $\hat{\dot{x}}_j = \Sigma\xi_{ij}f_i$ where the active (i.e., not yet culled) functionals for x_j are those with non-zero ξ_{ij}. The toolkit offers several modifications to this sequential culling:

1) CULL ONE FUNCTIONAL PER ITERATION

We wish to capture the importance of a given functional's loss in the FoM sequences. To tie individual functionals to the FoMs, we cull just one functional (from all variables combined) per iteration. We continue culling until one variable loses all its functionals (with an optional restart, see II-F1). A faster alternative, but with less granularity in terms of FoMs, would be to cull multiple functionals per iteration. A workable compromise is to cull multiple functionals during early iterations, then to reduce to one functional per iteration later as the active libraries get sparser.

2) CULL VIA LINEAR DEPENDENCE OF FUNCTIONALS

Various functionals in a library may be de facto (i.e., given the noise level) linearly dependent, with an "incorrect" functional in the span of a "correct" functional. For example, in Lorenz, $\dot{x} = \xi_1 x + \xi_2 y$, but xz and yz are in the span of x and y (see Fig. 5). In this case, the coefficients chosen by ℓ_2 regression are but one choice among many, and other linear combinations of functionals are effectively equivalent.

We address this as follows: Each iteration, we check the linear dependence of the active functionals (for each variable separately), calculating R^2 values for leave-one-out ℓ_2 fits. If some functionals show linear dependence via high R^2 values (above some threshold e.g., 0.95), we cull the one with the lowest rescaled (cf II-E3) coefficient. That is, we cull based on coefficient magnitude (as usual), but we restrict that iteration's culling candidates to just those functionals that are linearly dependent. This method handles large initial libraries well, reliably culling excess functionals.

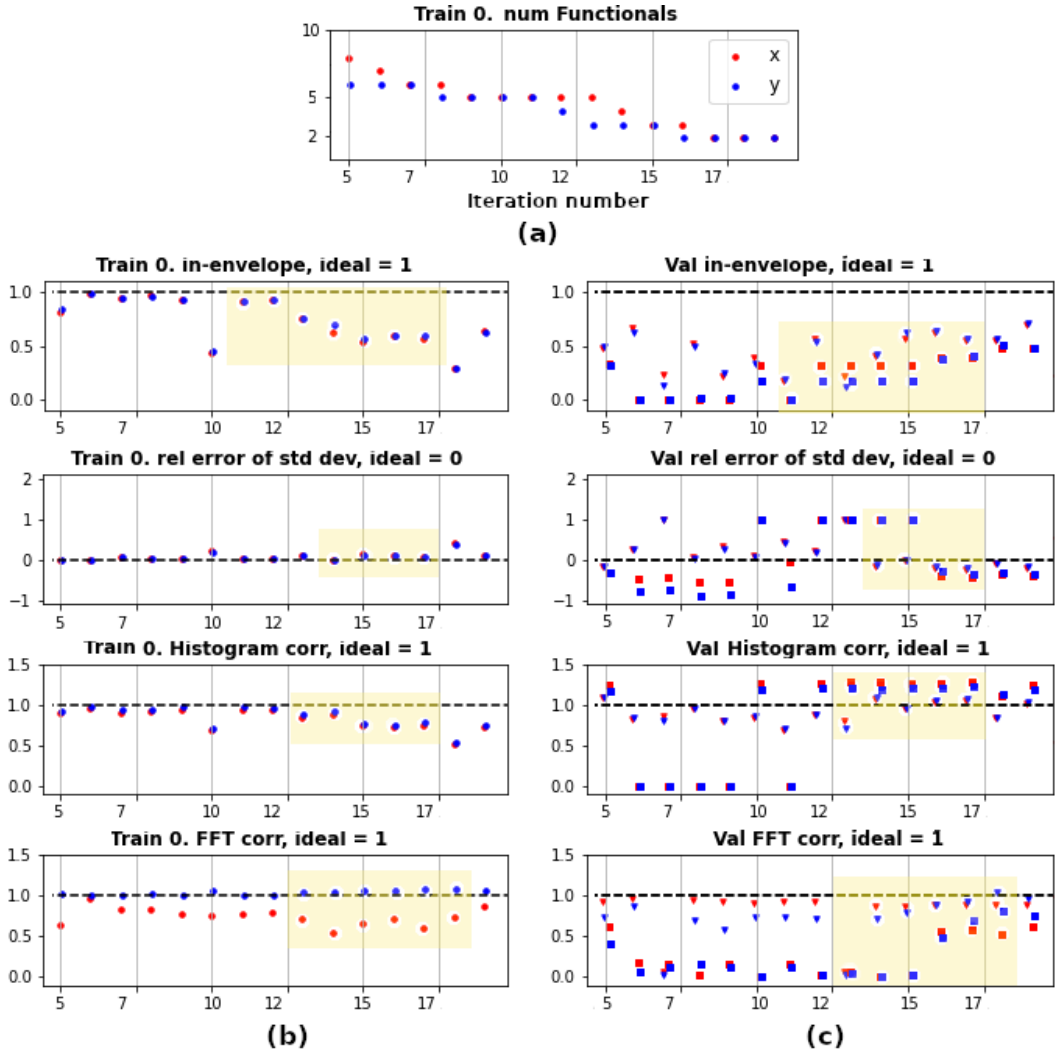


FIGURE 4. Figures of Merit for train and validation trajectories. Harmonic cubic oscillator; true model has 2 functionals per variable. Plots show useful FoMs, and how validation trajectory FoMs add information. Yellow highlighted blocks mark iterations where validation FoMs improve while training FoMs drop or hold steady. Dotted lines show ideal values. (a) The number of functionals for x and y in the model, as sparsity increases through iterations. (b) (rows 2–5) FoMs for the home (training) trajectory. (c) (rows 2–5) FoMs for the two validation trajectories (one trajectory as squares, one as triangles). Row 2: Fraction of the evolved trajectories "in envelope". Note the trade-off in "in envelope" FoM between home trajectory and validation trajectories after iteration 12: training accuracy decreases, but validation accuracy increases. Row 3: Relative error of std dev of evolved trajectory (0 is ideal). Note that the home accuracies are consistently good whereas validation accuracies vary substantially by iteration. Rows 4 and 5: Correlation to histograms of evolution values (row 4) and FFT power (row 5). In both cases, home accuracies are generally high, whereas validation accuracies vary by iteration. In row 4, home accuracy drops after iteration 14 whereas validation accuracy remains high. In rows 4 and 5 of the validation (right-hand) column, some models show good fit to one trajectory (triangles) but not the other (squares).

3) RESCALE THE FUNCTIONAL COEFFICIENTS ACCORDING TO MAGNITUDE OF FUNCTIONAL VALUES

Library functionals often have radically different value ranges, which can strongly bias culling based on coefficient magnitude. For example, suppose $x(t)$ is generally around 10. Then $x \approx 10, x^2 \approx 100$, etc. This translates directly into much smaller coefficients ξ for some functionals, which increases the likelihood they will be culled by thresholding, regardless of whether they are part of the "true" governing equation. To mitigate this bias, we cull coefficients not based on their regression coefficients ξ, but on rescaled versions $v_i\xi$ such that the contribution of each $v_i\xi_i f_i$ term is (very roughly) equal. We approximate this ideal for a particular x_j as follows: For each active functional we calculate a median value over the fitted timepoints ($=med(f_i)$), then normalize by some percentile of all active functionals' medians to get rescaling factors v:

$v_i = \frac{med(f_i)}{M}$ where $M = m^{th}$ percentile of $\{med(f_i)\}$ over all active f_i. Extreme values of v_i (< 0.1 or > 10) are clipped. For choice of m, see II-H.

The parameter m determines which active functional is considered "standard", i.e., has $v = 1$. Then $v_i\xi_i > \xi_i$ if $med(f_i) > M$ (relatively high values of f_i lead to relatively low values of ξ_i, which v_i offsets). Similarly, $v_i\xi_i < \xi_i$ if

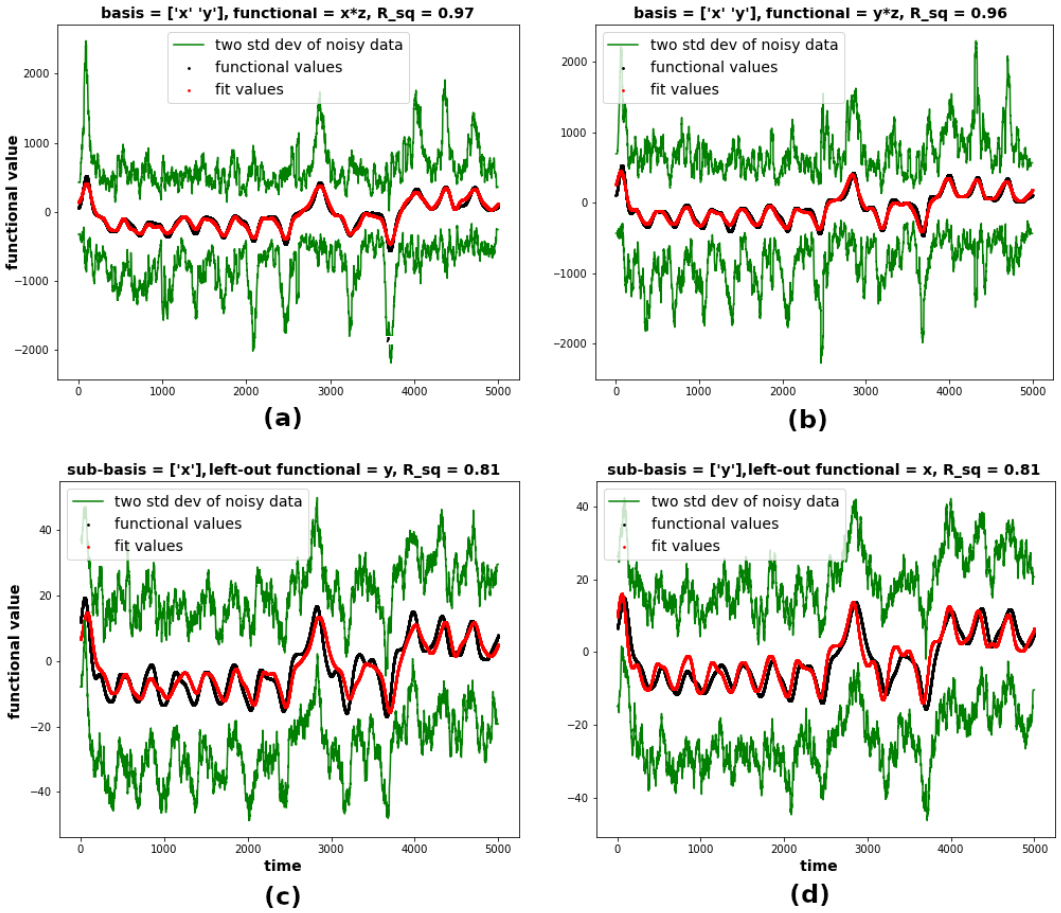


FIGURE 5. **Linear dependence of functionals.** Lorenz system. For each subplot: x-axis = time, y-axis = functional value. Black = functional to be fitted (values using smoothed data), red = linear fit by basis functionals, green = noise envelope (2 std dev). (a, b) Given a discovered sparse set of governing equation functionals, we can test whether other (culled) functionals are in their linear span, relative to the noise envelopes. If yes, they are potential candidates for the "true" governing equations even though they were culled. In Lorenz's $\dot{x}$, xz and yz appear to be within the span of the discovered set {x and y} ($R^2 \approx 0.97$), so they are plausible"true" functionals, perhaps instead of x or y. (c, d) We can also test via leave-one-out whether any of the discovered active functionals are redundant. In Lorenz's $\dot{x}$, the selected functionals x and y have relatively poor leave-one-out fits ($R^2 = 0.81$), indicating they are both essential (linearly independent).

$med(f_i) < M$ (relatively low values of f_i lead to relatively high values of ξ_i, which v_i offsets).

The v_i depend on which f_i are active ($\xi_{ij} \neq 0$), so they differ for each x_j and also change as functionals are culled. This rescaling method replaces the "do-nothing" default that benefits small-valued functionals with a deliberate choice that balances large- and small-valued functionals by increasing (for culling purposes only) the coefficients of large-valued functionals. This rescaling does not guarantee that the correct functionals will be preferred; rather, it allows the user to tune this aspect of culling based on domain expertise and the chosen library.

4) CONSTRAIN THE IMBALANCE IN COEFFICIENTS PER VARIABLE

STLSQ, if applied to the full system (ie all variables $\dot{x}_j$) removes the functional with smallest coefficient, regardless of which variable that functional acts on. This can lead to imbalances, where one variable retains a dense library of active functionals while another variable becomes overly sparse, since the complexity in the overall system is incorrectly captured by one variable's library. A domain expert might have educated guesses about the relative symmetry of a system. We encode this as a constraint on the maximum allowed difference in library sizes between variables (e.g., 3). If in a particular iteration one variable has many fewer active functionals, those functionals are ignored by the culling. In this case some other variable loses a functional, bringing the total counts into closer balance.

5) RESTORE AND PROTECT CULLED FUNCTIONALS IF FoMs DROP

A basic problem with greedy algorithms such as STLSQ is that a "true" functional may get culled early and is then permanently lost, which hurts the performance of downstream models.

Because we collect FoMs at each iteration, we have immediate notice if culling a particular functional degrades model

performance. Sufficient degradation (e.g., 50% reduction in some FoM for some variable) triggers (i) restoration of the culled functional, and (ii) protection of that functional from culling for the next few iterations. This allows time for other functionals to be culled instead, which changes the landscape of coefficient values. If the restored functional's coefficient increases, it gets preserved going forward; whereas if its coefficient remains low, it gets culled later. See Fig. 6 for an example of restoration.

F. FINAL STEPS

1) RESTART PROCESS ON REMAINING VARIABLES

The culling iterations continue until all functionals have been removed for one variable. The procedure can optionally restart the STLSQ iterations on the remaining variables. The new iterations use the original full library of functionals, minus any functionals containing the removed variable. This is an effective way to detect and remove pure noise variables whose $\dot{x} =$ constant (modulo noise effects). It is an irrelevant step if all the original variables were salient.

2) CHOOSE BEST MODELS

By consulting an FoM mosaic, one can select an optimal model that performs well on both train and validation trajectories, and is sufficiently sparse. In addition, the FoM sequences give clues as to whether a potentially relevant functional was incorrectly dropped early, since this often causes a drop in FoMs in the iteration when it was culled. Validation FoMs give insight into generalizability. The text print-out of model coefficients at each iteration can be matched with the FoM mosaics to glean insights into which functionals may be most important. Also, selected models can be evolved over train and validation trajectories, to allow visual inspection of their predictions.

3) COMBINE BEST MODELS

If there are multiple training trajectories, each trajectory produces a sequence of models and an FoM mosaic, and thus a different optimal model. These models often have different sparse libraries due to differences in training trajectories. In this case, the active functionals of the various models can be combined (a union of sparse libraries), and the full procedure restarted on all trajectories with this new initial library (which is much smaller). Note that the union of libraries respects the differences between variables: For example, if some model has $\xi_{i1} \neq 0$, but all models have $\xi_{i2} = 0$, then in the union library f_i is included for x_1 but not for x_2. Functionals suspected of having been incorrectly dropped (based on a drop in FoMs at some iteration) can also be reinstated at this point.

This method usually improves the discovered models substantially. It combines the positive findings of each training trajectory to create a concentrated starting library of highly-likely candidate functionals. It also mitigates greedy culls of "true" functionals, since the loss is reversed if the functional was preserved by another trajectory.

4) FIND ALTERNATE ACCEPTABLE LIBRARY FUNCTIONALS

As mentioned, a set of functionals may be effectively linearly dependent, $f_k(t) \approx \Sigma_{i \neq k} \beta_i f_i(t)$, if the approximation is well within the noise envelope of the data. Thus there may be multiple plausible sparse models for a system. Given a final model, we do two types of checks for linear dependence (examples are from the Lorenz system with 150 - 200% added noise):

1) We apply linear regression to each culled functional's time-series, using the retained functionals' time-series as features. The goodness of fit, e.g., shown by R^2 values, indicates whether the culled functional might be a viable alternative to the chosen functionals for defining governing equations. Examples:
(i) $\dot{x} = \beta_x x + \beta_y y$; but $xz \approx \beta_1 x + \beta_2 y$ ($R^2 = 0.97$) and $yz \approx \beta_3 x + \beta_4 y$ ($R^2 = 0.96$) (see Fig. 5 row 1). So xz and yz are potentially viable substitutes for x.
(ii) $\dot{y} = \beta_x x + \beta_y y + \beta_{xz} xz$. But $yz \approx \beta_1 x + \beta_2 y + \beta_3 xz$ ($R^2 = 0.99$), so yz is a viable substitute for one of the "true" functionals.
2) We apply "leave-one-out" linear regression to the set of retained functionals, fitting each functional's time-series using the time-series of the other functionals. This gives clues as to whether the retained functionals are all necessary. Examples:
(i) $\dot{x} = \xi_x x + \xi_y y$, and neither can be well approximated by the other ($R^2 \approx 0.81$, see Fig. 5 row 2). So they are both essential.
(ii) $\dot{y} = \xi_x x + \xi_y y + \xi_{xz} xz$. However, x or xz might be redundant: $x \approx \beta_1 y + \beta_2 xz$ ($R^2 = 0.98$) and $xz \approx \beta_3 y + \beta_4 x$ ($R^2 = 0.97$). But y is not in the span of x and xz ($R^2 \approx 0.89$), so it is not redundant. "Leave-one-out" in-span behavior is not necessarily symmetric.

This method acknowledges that solutions may not be unique given high-noise data. The output is a list of possible alternative functionals that are in the linear span (in the sense described above) of the discovered functionals. This list of alternatives enables domain experts to identify other functionals potentially viable for use in governing equations, rather than being constrained to just a single discovered model.

G. ASSESSMENT OF DISCOVERED MODELS USING LINEAR DEPENDENCIES

In the previous section, linear dependence was used to help domain experts identify alternate candidate functionals. In this section, our sole purpose is to assess whether a discovered model has an equivalent form that is close to a known set of governing equations, to determine whether the discovery method works (oracle assessment). This problem and technique are not specific to SINDy, but apply to data-driven equation discovery in general given noisy data.

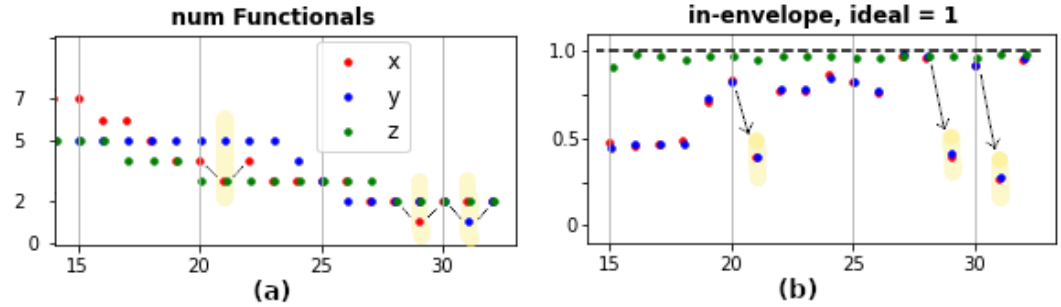


FIGURE 6. Restoring culled functionals. Large degradation of the "in-envelope" FoM due to culling a functional triggers restoration of the culled functional. In the 3-D linear harmonic oscillator, the "true" functionals for $\dot{x}$ and $\dot{y}$ are x and y. At iteration 21 x was culled from the active functionals for $\dot{x}$ (a), causing degradation of the "in-envelope" FoM (b). This triggered restoration of x at iteration 22 and temporarily placed it under protection from culling. The two incorrect functionals were subsequently culled from $\dot{x}$ leaving only x and y. At iteration 29, x was again culled from $\dot{x}$, degrading the "in-envelope" FoM and triggering restoration. Similarly, at iteration 31 y was culled from $\dot{y}$ causing degradation, triggering its restoration.

Our ability to assess whether a discovered model matches the original "true" model is complicated by the non-uniqueness of solutions in noisy regimes: There may be multiple sparse coefficient matrices Ξ such that $\dot{\boldsymbol{x}} = \Xi F$ accurately reproduces dynamics of the system. We describe construction of a non-degenerate linear transform that converts the discovered model coefficients $\hat{\Xi}$ to a new, equivalent form $\hat{\Xi}'$ which as closely as possible matches the "true" Ξ, while maintaining the same dynamical behavior. Our goal is to accurately assess errors in discovered sparse functional libraries and in estimated coefficients. For this purpose we assume oracle knowledge of the "true" model. The method is as follows:

1) Take as features the time-series of the "true" model's library L over a set of timepoints T. For each x_j, do linear fits of each functional $g(t)$ in the discovered model, $g(t) = \Sigma_{i\in L}\beta_i f_i(t) + \epsilon(t)$ (where $\epsilon(t)$ is the fitting error), and record the R^2 value of the fit (cf section II-F4).
2) Based on a pre-set R^2 threshold (e.g., 0.95) that indicates a sufficiently close fit, see if any g are in the span of the true functionals. This threshold partly depends on the size of the noise envelopes.
3) If a discovered g is in the span of L, use the linear relationship to substitute g out of the discovered model, replacing it with true functionals. After this step, the transformed sparse library overlaps the "true" sparse library as closely as the R^2 threshold allows.
4) Using leave-one-out linear fits, calculate the linear dependencies within the transformed library.
5) If there are linear dependencies with sufficiently high R^2 values, apply iterative substitutions to shrink the largest error in the modified vs "true" coefficients $|\frac{\hat{\xi}_{ij}-\xi_{ij}}{\xi_{ij}}|$. After these iterations, the discovered model has a form whose coefficients match the "true" model as closely as the R^2 threshold allows, in the sense of having smallest maximum error. A side effect of minimizing errors of true functionals is to minimize the coefficients of surplus functionals.
6) Evolve the transformed model, to confirm that trajectories are unchanged (or improved).

Examples: Fig. 7 shows missing "true" functionals that are linear combinations of discovered functionals. Fig. 5 shows linear dependencies as used in step 3 above. Fig. 8 shows linear dependencies as used in step 5 above. Results of oracle transformation are evident in Tables 1 to 4, and Tables 7 and 8.

H. HYPERPARAMETER CHOICES

The toolkit appears to have many tunable parameters, but most of them are effective over large ranges and can be left as-is. Below we list the hyperparameters with the most effect on outcomes. However, in the experiments described here only one hyperparameter ("polynomialLibraryDegree") was tuned per system, and one hyperparameter ("hammingWindowLengthForData") was changed for one system. The rest of the parameter values were shared by all systems. Full details can be found in the codebase [58].

polynomialLibraryDegree

When choosing a polynomial library (I-C), too small a degree might miss crucial functionals while too large a degree has several ill effects: It increases run-time because there are more functionals to cull (though culling by linear dependence early on mitigates this considerably); it increases the likelihood of a spurious functional displacing a "true" functional in the sparse solutions; and it can cause the ODE solver to hang.

hammingWindowLengthForData

This controls the amount of low pass filtering applied to the raw noisy data to create the smoothed time-series that the toolkit works on (II-B). Too large a window risks removing important high frequency information. Too small a window risks creating time-series that are too convoluted. However, the squiggles created by a small window length are more easily mitigated than too much smoothing, so this parameter can be set lower, to err on the side of less filtering. The window size also depends heavily on the data set's sampling

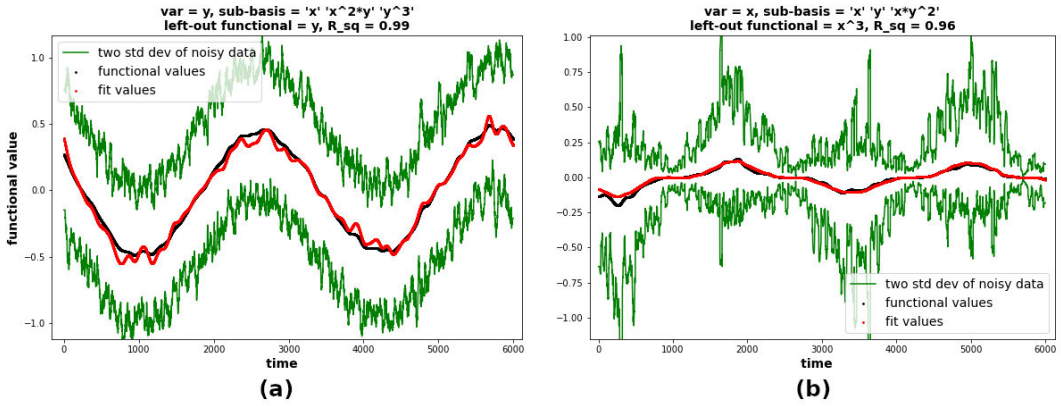


FIGURE 7. **Culled "true" functionals are often in the linear span of discovered models. From the discovered model for one of the training trajectories, Lorenz with 220 - 300% added noise. (a) The discovered sparse library for $\dot{x}$ was $\{x, xz, yz\}$. The "true" functional y was culled, but was in the span of the discovered functionals ($R^2 \approx 0.98$). (b) Similarly, the discovered model for $\dot{z}$ was $\{x, x^2, y^2\}$. The true functional xy was culled, but was in the span of the discovered functionals ($R^2 \approx 0.97$).**

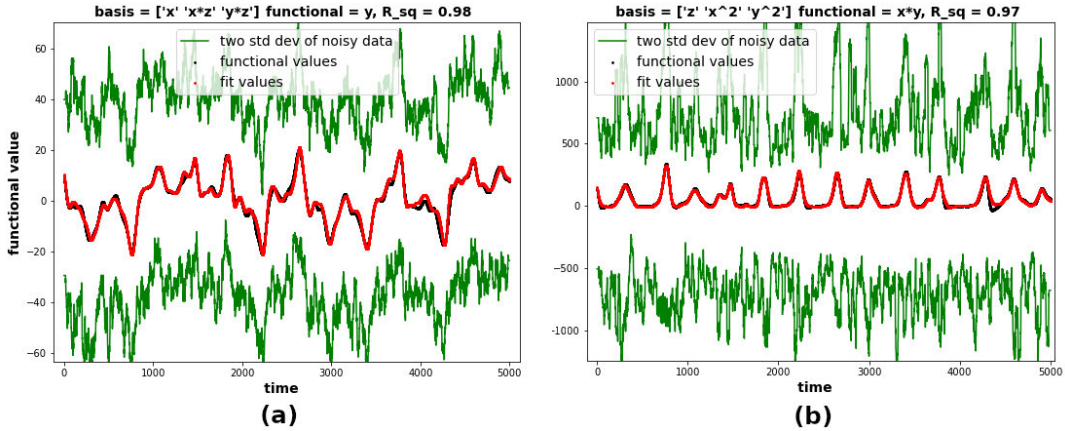


FIGURE 8. **Linear dependence of functionals, Hopf Normal 2-D. For each subplot x-axis = time, y-axis = functional value. Black = functional to be fitted (values using smoothed data), red = linear fit by basis functionals, green = noise envelope (2 std dev). Each plot shows the linear dependence of a left-out functional ((a) y, (b) xy) on the other "true" functionals of $\dot{y}$, in the form of a linear regression with $R^2 > 0.95$. An extreme example is shown on the left: $y = 0.02x + 4.34y^3 + 4.54x^2y \implies 0 = -y + 0.02x + 4.34y^3 + 4.54x^2y$ with $R^2 = 0.99$. These equalities can be substituted in to find a version of the discovered model that is closest in form to the "true" model.**

rate. It can be somewhat optimized by inspecting plots of smoothed data.

BalancedCullNumber

This determines how similar the numbers of functionals per variable in the sparse models will be (II-E4). If some level of symmetry is known, this is a powerful prior. Even when no particular symmetry is known, a moderate value like "3" (i.e., the functional counts of any two variables will differ by at most 3) prevents situations where all the system's complexity gets lopsidedly encoded into one variable with many terms.

percentileOfImputedValuesForWeights

Weighting coefficients based on functional magnitude (II-E3) basically defines a "central" value, and weights functionals proportionally while clipping the extremes (>10× ratio). A low parameter value (e.g., 10) shifts preference towards smaller-valued functionals, which typically means that lower-degree monomials get weighted more heavily than by a larger parameter value. This offsets the combinatorially larger number of high-degree functionals in a polynomial library. The effect of this parameter decreases substantially as the libraries get smaller; its main effect is in the initial culling stage.

numEvolutionsForFom

Evolving models multiple (e.g., 3) times detects unstable models, with more evolutions doing this more reliably (II-D1). However, these evolutions carry by far the largest cost in the toolkit. So if stability is not an issue, one evolution is best.

restoreBasedOnFoMsFlag AND fomChangesDict

The user must decide whether to restore functionals based on drops in FoMs (II-E5), and if so which FoMs should trigger restoration. The most salient FoMs are listed in (II-D2).

III. RESULTS

We give results for several dynamical systems with added noise: Lorenz, linear and cubic harmonic oscillators, linear 3-D, and the Hopf Normal form (2-D). Background on these systems can be found in [1]. The toolkit performs well on all these systems at varying levels of noise. Results for the Lorenz system are given in this section. Results for the other systems are in the Appendix. The Lorenz system (here) and Hopf system (Appendix) clearly show the importance of the linear dependence method (cf II-G) when assessing discovered models.

The toolkit requires various hyperparameters but none of them are brittle, and a wide range of values work well (perhaps because there are so many relatively "easy" gains to be had). The Lorenz system was our test-bed, which perhaps partially explains the higher levels of noise handled for Lorenz. The other systems were handled using the same hyperparameters used for Lorenz, with no additional tuning except as follows: (i) Hamming window length for the harmonic cubic oscillator was 300 instead of 200; and (ii) we varied the degree of the initial polynomial library.

A. ADDED NOISE

In these experiments, white noise was added as follows: Each variable $x_j(t)$ of a trajectory was transformed by FFT to give complex frequency coefficients. Noise drawn from a complex gaussian distribution $\mathcal{N}(0, \sigma)$ was added to each coefficient. An inverse Fourier transform gave a complex-valued trajectory, of which the real part was retained. Noise level (%) was defined as $100 \times \sigma_{noise}/\sigma_x$ where σ_{noise} was standard deviation of the noise-added trajectory minus the clean trajectory, and σ_x was standard deviation of the original clean trajectory.

B. THE LORENZ SYSTEM

We consider the canonical Lorenz system, a 3-dimensional chaotic "butterfly"-shaped attractor with two lobes, shown previously in Fig. 1 (time-series view) and also below in Fig. 9 (3-D view). White noise was added at levels equivalent to 50%, 100%, 200%, and 300%, and results for typical runs are reported.

The true system has ODEs:

$$\dot{x} = -10x + 10y \tag{3}$$

$$\dot{y} = 28x - y - xz \tag{4}$$

$$\dot{z} = -2.67z + xy \tag{5}$$

We used three training trajectories with initial conditions [−8, 8, 27], [5, −7, 29], and [−2, 7, 21]; and two holdout (test) trajectories with initial conditions [8, 7, 15] and [−6, 12, 25]. The initial conditions of the first training trajectory are from [1]; all others were selected at random. The training trajectories were 10 seconds long with 0.002 second timestep. The initial functional libraries consisted of all polynomials up to degree 4 (almost all the degree 4 polynomials were rapidly eliminated by culling based on linear dependence, cf section II-E2).

The method typically recovers the correct sparse functional libraries (or, in some cases, linearly dependent equivalent libraries). However, as noise increases the coefficient estimates become less accurate, which leads to worse prediction on holdout trajectories. If the coefficient error becomes large enough, the qualitative behavior of predicted trajectories changes.

For results at all noise levels, the discovered and "closest to true" models (three models, one per training trajectory) are listed along with coefficient errors. The key is as follows: Column 2 gives the raw discovered equations. Column 3 gives the "closest to true" equations, after transformation using linear dependencies with $R^2 \geq 0.95$. Columns 4 and 5 give the absolute coefficient errors for each true functional, $|(\hat{\xi} - \xi)/\xi|$ as percentage (or "*inf*" if the functional is missing).

In the true library of $\dot{y}$, the y functional is linearly dependent on the x and xz functionals, so it could be substituted in, as seen in the transformed ("closest to true") versions of $\dot{y}$. In $\dot{x}$, some incorrect functionals were in the span of the true library (cf Fig. 7), allowing substitution in some cases. The true library of $\dot{z}$ had no relevant linear dependencies, so no coefficient transforms were possible.

Which discovered model would give the best test set evolutions was in all cases fully predictable based on quality of training and especially validation set predictions.

1) 50% ADDED NOISE

Typical discovered models and coefficient errors for 50% added noise are given in Table 1. In general, correct functionals (an exception is given below) were selected, with very low coefficient error. In all models, the raw $\dot{x}$ and $\dot{z}$ equations had "correct" functionals. All models missed the y functional in the $\dot{y}$ estimate. However, evolutions of validation and test trajectories were still very accurate, which highlights the subjective nature of assessing "correctness" in the model discovery context. Test set predicted evolutions of the best model (determined on train-validation results) are shown in Fig. 9 (left column).

2) 100% ADDED NOISE

Typical discovered models and coefficient errors for 100% added noise are given in Table 2. Raw models 0, 1 gave simple figure-8 trajectories, whereas raw model 2 gave more qualitatitively accurate trajectories. In all cases, the transformed models gave improved, qualititatively realistic trajectories and discovered models contained the correct functionals: The raw $\dot{x}$ equations used xz ("wrong") instead of x, but linear dependencies gave equivalent forms with the "true" functionals x and y. The raw $\dot{y}$, $\dot{z}$ equations contained the true functionals. Test set predicted evolutions of the best model (based on train-validation results) are qualitatively correct, as shown in 9 (center column).

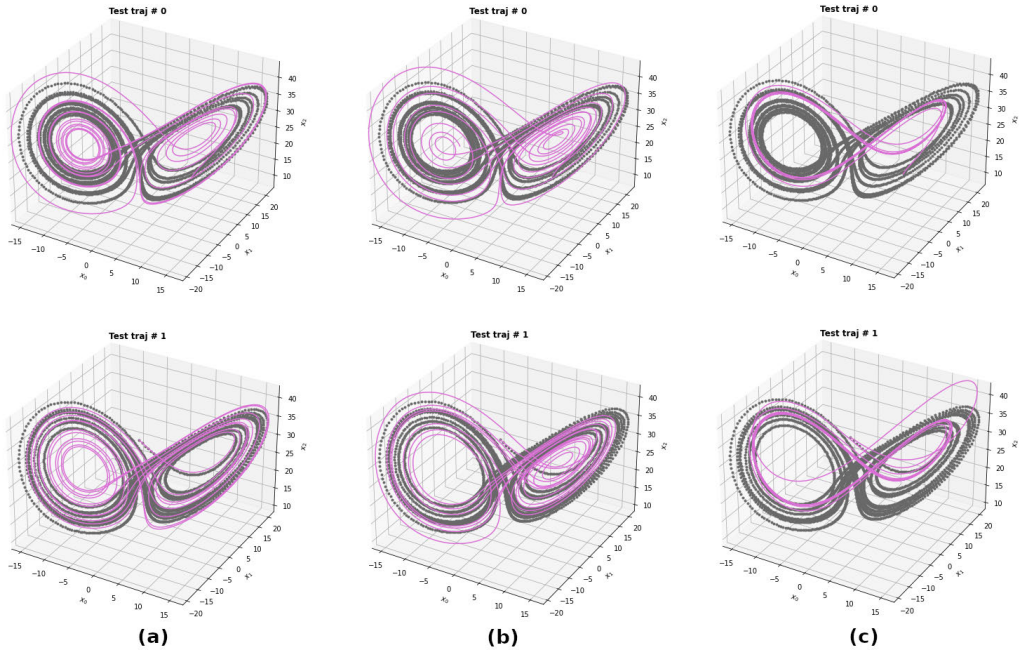


FIGURE 9. Test set evolutions for 50%, 100%, and 200% training data noise. Evolutions on two test trajectories of models trained on Lorenz system data with added noise (cf Fig. 1). Evolutions are from the model with the best train and validation trajectory predictions. Grey dots: original test data. Purple lines: Evolved trajectory. Evolved trajectories are good at lower noise, and deteriorate somewhat as noise increases, though overall trend and qualitatively correct behavior is maintained. At 300% noise (not shown), the predicted test trajectories degenerate into simple figure-8s (i.e., one loop per lobe, rather than multiple loops in a lobe before switching). (a) Noise level 50%. (b) Noise level 100%. (c) Noise level 200%. Top row: Test trajectory 0. Bottom row: Test trajectory 1.

TABLE 1. Lorenz 50% noise: Discovered (raw) and equivalent models, and absolute coefficient errors for each true functional, $|(\hat{\xi} - \xi)/\xi|$ as percentage ("*inf*" indicates a missed true functional). "Raw" errors are for the discovered equation, "closest" errors are for the transformed equation (cf section II-G).

ODE	Raw Eqn	Closest Eqn	Raw Err %	Closest Err %
Model 0				
$\dot{x}$	$-10.13x + 10.16y$	same	(1, 2)	(1, 2)
$\dot{y}$	$24.39x - 0.93xz$	$27.73x - 1.01y - 1.01xz$	(13, *inf*, 7)	(1, 1, 1)
$\dot{z}$	$-2.62z + 1.02xy$	same	(2, 2)	(2, 2)
Model 1				
$\dot{x}$	$-9.8x + 9.91y$	same	(2, 1)	(2, 1)
$\dot{y}$	$0.8 + 24.04x - 0.92xz$	$0.8 + 27.29x - 1.01y - 1.0xz$	(14, *inf*, 8)	(3, 1, 0)
$\dot{z}$	$-2.65z + 1.04xy$	same	(1, 4)	(1, 4)
Model 2				
$\dot{x}$	$-9.72x + 9.7y$	same	(3, 3)	(3, 3)
$\dot{y}$	$25.48x - 0.97xz$	$27.66x - 0.65y - 1.02xz$	(9, *inf*, 3)	(1, 35, 2)
$\dot{z}$	$-2.58z + 1.02xy$	same	(3, 2)	(3, 2)

Transforming using R^2 threshold $= 0.85$ (vs 0.95) gave models with (a) lower errors in coefficient estimates (median errors $= 9$, 6, and 4%); (b) trajectories that were improved vs raw but were not quite as good as when using a 0.95 threshold. These results highlight that care is required choosing an R^2 threshold. We propose two criteria: Substituted-in functionals should have close linear fits (by discovered functionals) that are well within the noise envelope; and equivalent models should have similar evolutions on train-validation trajectories.

3) 200% ADDED NOISE

Added noise was 205% to 220%. Typical discovered models and coefficient errors for 200% added noise are given in Table 3. In all cases, the raw equations had the "true" functionals. Model 2 had the best train-validation trajectories, the lowest errors, and also the best test trajectories (shown in 9, right column). Despite apparently small differences in coefficients (relative to Model 2), Model 0 had poor train-validation trajectories, highlighting how variations in quantitative error do not necessarily reflect changes in behavioral error.

4) 300% ADDED NOISE

Typical discovered models and coefficient errors for 300% added noise are given in Table 4. In each discovered model, $\dot{x}$ contains the linearly dependent term xz ($R^2 \approx 0.97$). Substituting it out during assessment left the "true" functionals x and y. Constant terms (e.g., in model 2's $\dot{y}, \dot{z}$) could not be substituted out. All the discovered models

TABLE 2. Lorenz 100% noise: Discovered (raw) and equivalent models, and absolute coefficient errors for each true functional, $|(\hat{\xi} - \xi)/\xi|$ as percentage ("*inf*" indicates a missed true functional). "Raw" errors are for the discovered equation, "closest" errors are for the transformed equation (cf section II-G) using R^2 threshold 0.95.

ODE	Raw Eqn	Closest Eqn	Raw Err %	Closest Err %
Model 0				
$\dot{x}$	$0x + 5.88y - 0.18xz$	$-6.9x + 7.55y$	(*inf*, 41)	(31, 24)
$\dot{y}$	$21.37x + 1.93y - 0.96xz$	$26.43x + 0.51y - 1.08xz$	(24, 293, 4)	(6, 151 , 8)
$\dot{z}$	$-2.52z + 1.01xy$	same	(6, 1)	(6, 1)
Model 1				
$\dot{x}$	$0x + 5.29y - 0.14xz$	$5.32x + 6.53y$	(*inf*, 47)	(47, 35)
$\dot{y}$	$20.19x + 2.25y - 0.89xz$	$26.96x + 0.42y - 1.06xz$	(28, 325, 11)	(4, 142, 6)
$\dot{z}$	$-2.62z + 1.06xy$	same	(2, 6)	(2, 6)
Model 2				
$\dot{x}$	$0x + 6.48y + -0.22xz$	$-8.37x + 8.67y$	(*inf*, 35)	(16, 13)
$\dot{y}$	$20.12x + 1.83y - 0.83xz$	$27.81x - 0.45y - 1.01xz$	(28, 283, 17)	(1, 55, 1)
$\dot{z}$	$-2.49z + 0.99xy$	same	(7, 1)	(7, 1)

TABLE 3. Lorenz 200% noise: Discovered (raw) and equivalent models, and absolute coefficient errors for each true functional, $|(\hat{\xi} - \xi)/\xi|$ as percentage ("*inf*" indicates a missed true functional). "Raw" errors are for the discovered equation, "closest" errors are for the transformed equation (cf section II-G).

ODE	Raw Eqn	Closest Eqn	Raw Err %	Closest Err %
Model 0				
$\dot{x}$	$-5.28x + 5.85y$	same	(47, 41)	(47, 41)
$\dot{y}$	$16.04x + 4.67y - 0.93xz$	$24.21x + 2.88y - 1.15xz$	(43, 567, 7)	(14, 388, 15)
$\dot{z}$	$-2.22z + 0.86xy$	same	(17, 14)	(17, 14)
Model 1				
$\dot{x}$	$-5.58x + 5.42y$	same	(44, 46)	(44, 46)
$\dot{y}$	$23.92x + 2.77y - 1.07xz$	$27.03x + 2.19y - 1.15xz$	(70, 487, 52)	(3, 319, 15)
$\dot{z}$	$-2.1z + 0.84xy$	same	(21, 16)	(21, 16)
Model 2				
$\dot{x}$	$-6.52x + 7.04y$	same	(35, 30)	(35, 30)
$\dot{y}$	$8.4x + 3.87y - 0.48xz$	$27.63x - 1.01y - 0.98xz$	(70, 487, 52)	(1, 1, 2)
$\dot{z}$	$-2.43z + 0.91xy$	same	(9, 9)	(9, 9)

contained the correct sparse libraries (Table 4), but they had degenerate trajectories consisting of simple figure-8s. So in terms of finding coefficients accurate enough to generate qualitatively correct Lorenz trajectories, the toolkit hits a failure point somewhere between 220% and 300% noise.

IV. DISCUSSION

We have presented a toolkit of methods to address noisy data, for data-driven discovery of governing equations. Though the toolkit is presented within the context of the SINDy method, many of its modules can be deployed in other, non-SINDy, architectures. In addition, although the various modules are strung together and presented here as a single architecture, they are intended to be deployed separately. Most of the modules are independent, e.g., regress on multiple sets of timepoints; weight coefficients for culling; ignore timepoints with large functional value differences; cull via linear dependence; and use FFTs as regression targets. Two of the modules have dependencies: Smoothing introduces artifacts that require mitigation by weighting timepoints and by regressing on FFTs; and restoring culled functionals requires generation of FoMs to provide triggering conditions.

The toolkit modules focus on (i) mitigating the effects of noise in various ways (e.g., removing timepoints from regressions; regressing over multiple sets of timepoints); and (ii) using models' predicted trajectories to assess model correctness (via FoMs and validation sets). The second item addresses the fact that many models can closely match a system's target derivatives while still generating poor time-series predictions. Thus optimization and model selection based solely on matching derivatives is insufficient.

A. USING LINEAR DEPENDENCE TO HANDLE NON-UNIQUENESS

In a second main contribution, we propose a method for calculating and assessing linear dependencies between library functionals, in order to address the inevitable non-uniqueness of models given noisy data and over-complete libraries. This method has two goals: First, it helps domain experts identify other valid candidate functionals beyond those found in the sparse discovered equations. Second, it enables identification of equivalent transformations of a discovered model, in order to accurately assess whether a discovered model that appears to be wrong has in fact an equivalent form that closely matches the "true" model. This is important when assessing any data-driven discovery method.

B. LIMITATIONS

The need to evolve models at each step, in order to generate FoMs, introduces a risk is that the solvers can hang, upsetting completion of the algorithm. This is more likely with large libraries of high order polynomials. One partial solution is to avoid evolutions when functionals are culled by linear dependence, since these culls tend to happen early, a time when the solvers also tend to hang. Another partial solution is to set time limits on the solvers, to allow them to exit. Both solutions have the drawbacks that (i) we lose visibility into the

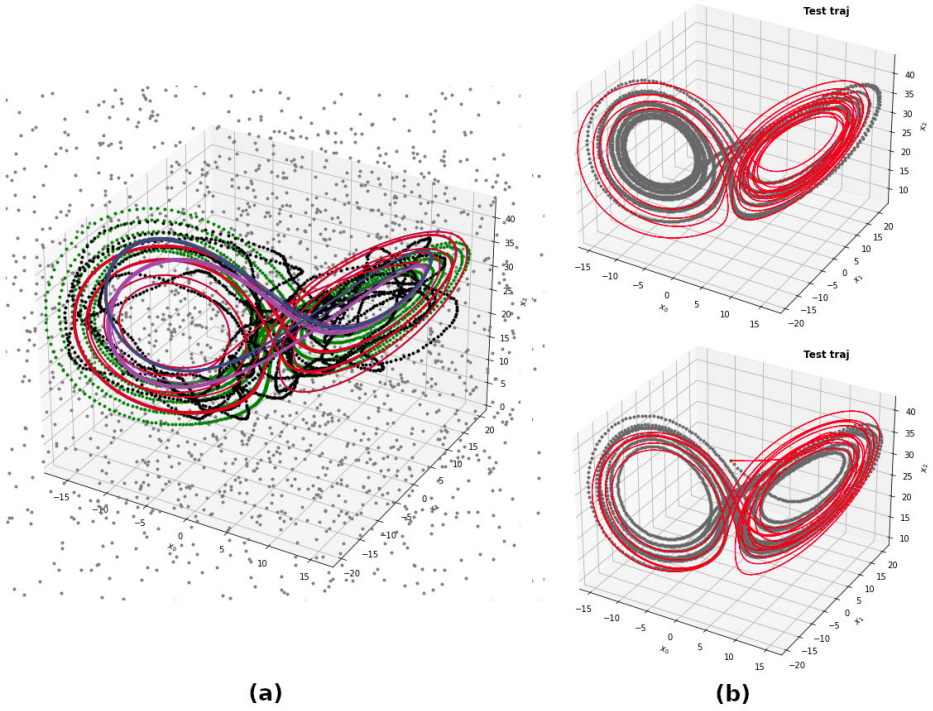


FIGURE 10. High noise (220 - 300%) Lorenz, predictions of training and test trajectories. (a) Training trajectory: Green = true clean trajectory; Grey dots = with 220 - 300% added noise ((cf Fig. 1); Black = smoothed trajectory with artifacts. Red = predictions (correct behavior, i.e., multiple loops per lobe visit); Purples = predictions as validation, by other models (incorrect behavior, i.e., simple figure-8s). (b) Model predictions on the two test trajectories (correct behavior). Grey = true trajectory, Red = predicted.

TABLE 4. Lorenz 300% noise: Discovered and equivalent models, and absolute coefficient errors for each true functional, $|(\hat{\xi} - \xi)/\xi|$ as percentage (*"inf"* indicates a missed true functional, "$*$" an extra incorrect functional). "Closest" errors are for the transformed equation (cf section II-G).

ODE	Raw Eqn	Closest Eqn	Closest Err %
Model 0			
$\dot{x}$	$3.5x + 4.38y - 0.34xz$	$-7.7x + 6.09y$	(23, 39)
$\dot{y}$	$9.67x + 4.08y - 0.65xz$	$26.16x + 0.02y - 1.07xz$	(7, 102, 7)
$\dot{z}$	$-15.91 - 1.36z + 0.81xy$	same	($*$, 49, 19)
Model 1			
$\dot{x}$	$5.59x + 4.09y - 0.34xz$	$-5.61x + 5.82y$	(44, 42)
$\dot{y}$	$14.21x + 4.54y - 0.81xz$	$25.55x + 2.02y - 1.11xz$	(9, 302, 11)
$\dot{z}$	$-2.01z + 0.76xy$	same	(25, 24)
Model 2			
$\dot{x}$	$8.32x + 2.22y - 0.32xz$	$-2.18x + 3.52y$	(78, 65)
$\dot{y}$	$-3.06 + 21.79x + 1.59y - 0.89xz$	$3.06 + 27.17x + 0.63y - 1.03xz$	($*$, 3, 163, 3)
$\dot{z}$	$8.91 - 0.58x - 2.05z + 0.75xy$	same	($*$, $*$, 23, 25)

effects (via FoMs) of culling certains functionals, and (ii) we lose the ''restore'' option (section II-E5) for iterations lacking FoMs.

Another limitation is the relatively slow runtime (a few minutes on a laptop, versus seconds for vanilla SINDy). However, vanilla SINDy fails given even small amounts of noise, so the slower runtime can be viewed as a cost of handling high noise. The slower runtime is due to two things: (i) the serial evolutions of trajectories for FoMs; and (ii) the one-at-a-time culling method. The first problem can be mitigated somewhat by parallelizing training on different trajectories; by parallelizing evolutions of multiple trajectories by the same model (used to assess stability); and by skipping evolutions in the early culling iterations, when the models are too dense anyway. The second problem can be addressed by culling more than one functional per iteration, which however coarsens the view offered by the FoMs since a deterioration from one iteration to the next cannot be ascribed to exactly one culled functional. The runtime means that the FoM-related components of the toolkit, in their current form, would not work for certain use-cases (e.g., applications requiring real-time model discovery).

Finally, the work here has not yet been applied to PDEs, rational functions, or control use-cases. However, the methods presented are eligible for porting to these use cases. There is nothing limiting or constraining the mathematical architecture for applications in spatio-temporal systems governed by PDEs. Indeed, the toolkit offers valuable extensions and improvements to the diversity of methods developed for PDE discovery. An example might be using

this toolkit to generate an initial system estimate from high-noise data, to be followed by the coefficient refining methods in [53].

V. CONCLUSION

This work addresses two central challenges in data-driven discovery of governing equations: high signal noise, and assessment of a discovery method given non-unique solutions. First, the toolkit offers several independently deployable modules that mitigate high noise, especially in the context of a sparse solution assumption (e.g, SINDy). It thus offers solutions to the widespread and difficult problem of noisy data. Second, the toolkit has modules that leverage linear dependence of functionals' time-series to enable principled transformations of a discovered solution to other equivalent solutions. This allows more accurate assessment of a discovery method's effectiveness, by showing whether an apparently incorrect discovered solution is in fact close to the "true" solution. It also generates a range of possible equivalent solutions, giving domain experts broader insight into possible sets of governing equations beyond a single discovered solution.

APPENDIX

The Appendix has two main parts.

First, we briefly list of (i) some additional techniques usable with SINDy (e.g., with the pySINDy package [38]), and (ii) some ideas that did not yield clear benefit.

Second, we give results for several other dynamical systems: 3-D linear, linear harmonic oscillator, cubic harmonic oscillator, and the Hopf normal form (2-D). All are described in [1]. In general the toolkit gives good results on the 3-D linear and harmonic oscillator systems, and fair results on the Hopf system.

A. ADDITIONAL METHODS

1) METHODS FOR USE WITH TRADITIONAL SINDy

1) Use different initial libraries for each variable: Current SINDy methods, e.g., [38], use the same initial functional library for each variable. However, there is often reason to assign different libraries to each variable: (i) domain expertise might include or exclude certain functionals for certain variables; (ii) a first run of SINDy with a weak sparsity parameter might cull the libraries (differently for each variable), preparing for a second run with a tighter sparsity parameter.
2) Incrementally cull functional libraries via iterative applications of SINDy: SINDy can be run iteratively, with progressively tighter sparsity parameters, culling a few functionals each time. The value of this approach is that each new iteration does its regression with a smaller, higher-probability library. Nuisance functionals can be rejected early before the definitive regression (with high sparsity parameter) is run. This method appeared to improve SINDy models given some noise, though at lower noise levels than tolerated by the toolkit described here.
3) Smooth SINDy's derivative estimates then feed them back into SINDy: Because noisy (especially non-continuous) derivative estimates cause such trouble, the following method tends to improve SINDy predictions by improving the derivative estimates: (i) Smooth the initial derivative estimates; (ii) fit a SINDy model; (iii) extract the derivatives generated by the sparse SINDy model; (iv) smooth them; (v) refit the SINDy model, feeding in the smoothed derivatives as the $\dot{\boldsymbol{x}}$ argin. A possible reason this works is: the sparsifying effect of SINDy can remove noise from derivative estimates by simplifying their functional form; thus the SINDy model's derivative estimates can be cleaner than the original estimates based on the noisy data.

2) METHODS THAT HAD DOUBTFUL OR NO BENEFITS

1) Clipping, splining or shrinking noisy data points: These methods of reducing noise were not effective, mainly (we believe) because they did not return continuous derivative estimates. A low-pass filter performed better.
2) Culling functionals based on the (un)reliability of their coefficient estimates: Regressing several times over different sets of timepoints yields a distribution of coefficient estimates for each functional (cf section II-C4). The reliability of the estimates can be measured by, for example, σ/μ (std dev/mean). Suppose that true functionals might have consistent estimates (since they have a true role) whereas spurious functionals have noisy coefficient estimates (since they are spurious): Then functionals with unreliable estimates can be preferentially culled. However, experiments indicate that the coefficients of true and spurious functionals are entwined. If one functional's coefficient values swing wildly, it can cause swings in other functionals' coefficients. We note that [52], in independent work, applied this method successfully in a lower noise setting (e.g., 7% added noise in Lorenz).
3) Regress on the outcomes of short trajectories rather than on derivatives: Instead of regressing on derivatives, regression can be on short predictions, e.g., over 100 timesteps. This tends to reward models that have a longer time horizon than models regressing on, for example, Runge-Kutta derivative estimates. In our experiments, sometimes this alternative regression target worked better and sometimes not, with no clear pattern. In a lower noise setting, [52] successfully optimizes against an Euler 1-step prediction (i.e., a very short trajectory).
4) Ridge or Lasso regression (vs ordinary ℓ_2) did not improve results, and often gave worse results. Lasso introduces confusion because it also enforces sparsity.

B. RESULTS FOR OTHER SYSTEMS

1) LINEAR 3-DIMENSIONAL SYSTEM

We consider a three-dimensional linear system, with added white noise equivalent to 50% (see Fig. 11). Runs used

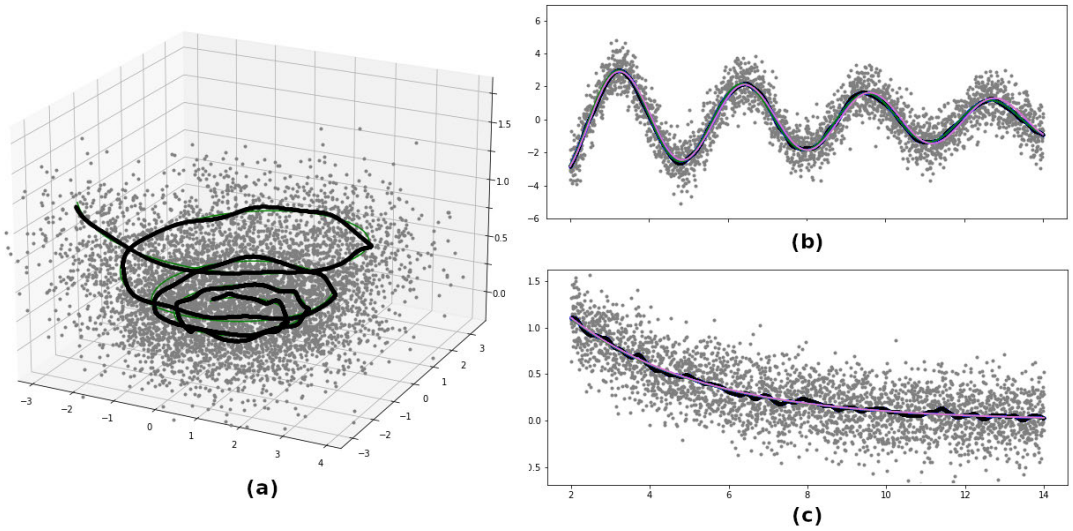


FIGURE 11. **Linear 3-D system validation trajectories. (a) Training trajectory with 50% noise (grey dots), clean (green), and smoothed (black). (b) Time-series of *x* (*y* is similar but with phase shift). (c) Time-series of *z*. Black lines are smoothed trajectories. Grey dots are trajectories with added noise. Fuschia lines are typical predicted validation trajectories.**

TABLE 5. **Linear 3-dimensional model, 50% noise: Discovered models and coefficient errors for runs using an initial library of polynomials up to degree 2 and up to degree 3. Columns 2 and 4 gives the raw discovered equations for each library. Column 3 and 5 give the absolute coefficient errors for each true functional, $|(\hat{\xi} - \xi)/\xi|$ as percentage ("*inf*" indicates a missed true functional, "$*$" an extra incorrect functional).**

ODE	using 2^{nd} degree library	Error %	using 3^{rd} degree library	Error %
Model 0:				
$\dot{x}$	$-0.12x-1.94y-0.62z+1.66z^2$	(20, 3, $*$, $*$)	$-1.98y+0.71xz-0.22x^3$	(*inf*, 1, $*$, $*$)
$\dot{y}$	$1.97x - 0.09y$	(1, 6)	$1.96x - 0.13y$	(2, 28)
$\dot{z}$	$-0.27z$	(10)	$-0.28z$	(7)
Model 1:				
$\dot{x}$	$-0.11x - 1.97y$	(8, 1)	$-2.0y - 0.17xz$	(*inf*, 0, $*$)
$\dot{y}$	$2.0x + -0.11y$	(0, 12)	$2.0x - 0.11y$	(0, 8)
$\dot{z}$	$-0.28z$	(8)	$-0.29z$	(2)
Model 2:				
$\dot{x}$	$-0.08x - 1.99y$	(21, 1)	$-1.99y - 0.02x^3$	(*inf*, 0, $*$)
$\dot{y}$	$1.99x - 0.12y$	(0, 24)	$1.98x - 0.12y$	(1, 19)
$\dot{z}$	$-0.3z$	(1)	$-0.3z$	(1)

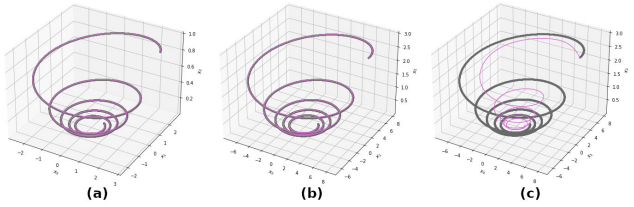


FIGURE 12. **Linear 3-D system test trajectories, effects of initial library. Typical predictions for the two Test trajectories by models trained on data with 50% noise. The two trajectories have different initial conditions (evident in the axes scales). Grey lines are true, fuchsia lines are predictions. (a, b) Using a 2^{nd} order polynomial library gave accurate predictions for both trajectories. (c) Using a 3^{rd} order polynomial library gave predictions with some degeneration for test trajectory 1 (compare to (b)) and still accurate for test trajectory 0 (not shown, similar to (a)).**

libraries of either $\leq 2^{\text{nd}}$ or $\leq 3^{\text{rd}}$ degree polynomials. Results for typical runs are reported.

The true system has ODEs:

$$\dot{x} = -0.1x - 2y \tag{6}$$
$$\dot{y} = 2x - 0.1y \tag{7}$$
$$\dot{z} = -0.3z \tag{8}$$

We used three training trajectories with initial conditions [2, 0, 1], [4, −1, 2], and [3, 3, 3]; and two holdout trajectories with initial conditions [3, 1, 1] and [9, 1, 3]. The initial conditions of the first training trajectory are from [1]; all others were selected at random. Trajectories were 24 seconds long with 0.002 second timestep.

At 50% noise, discovered models were very accurate (correct libraries; median coefficient errors 1 to 8%; accurate predicted trajectories) when the initial library contained degree 2 polynomials (Table 5 columns 1 and 2). When the initial library included degree 3 polynomials, cubic terms displaced the true minor terms in $\dot{x}$, while $\dot{y}$ and $\dot{z}$ estimates remained accurate (Table 5 columns 3 and 4). In all cases, the discovered models made highly accurate predictions for train and validation trajectories (see Fig. 11). Predictions of one test trajectory deteriorated somewhat given the 3^{rd} degree library models (see Fig. 12C).

At 70% added noise (initial library $\leq 3^{rd}$ degree polynomials) the following changes occurred: (i) the displacement of minor functionals increased; (ii) train and validation trajectory predictions remained accurate; (iii) test trajectory

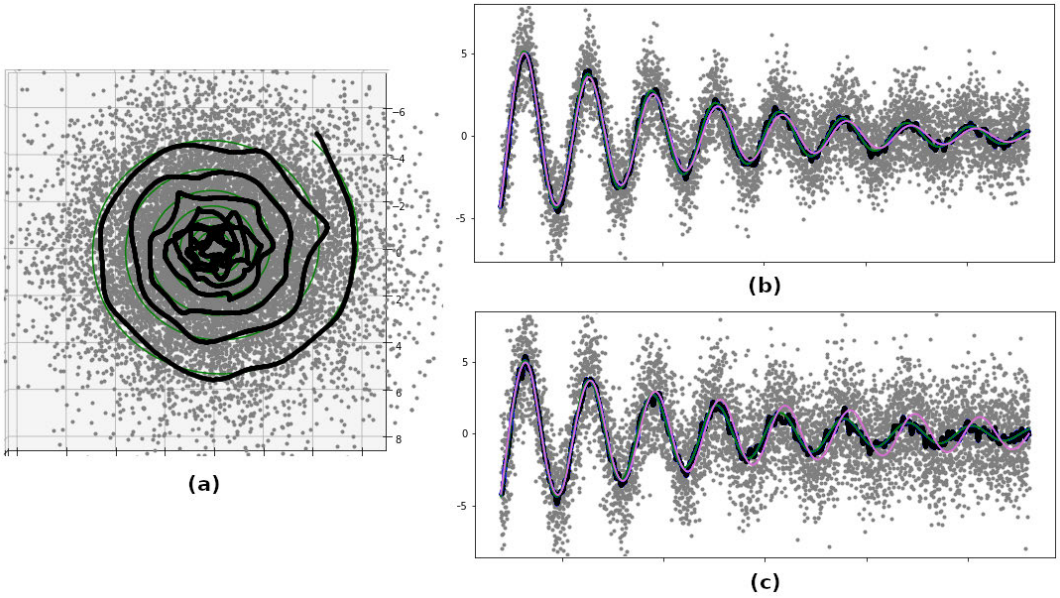


FIGURE 13. Harmonic linear oscillator, 70% and 100% noise. (a) Time-series in x-y plane, with 100% noise (grey dots), clean (green), and smoothed with squiggle artifacts (black). (b) Typical validation trajectory (x shown, y is similar but with phase shift) given 70% noise during training. (c) The same, but given 100% noise during training. Grey dots are trajectories with added noise. Black lines are smoothed trajectories. Fuschia lines are the trajectories as predicted by discovered models. Note the less-accurate prediction in the highly damped region given 100% training noise (subplot C).

TABLE 6. Harmonic linear oscillator, 70 to 100% noise: Discovered models and coefficient errors for 70% and 100% noise. Columns 2 and 4 gives the raw discovered equations for 70% noise and 100% noise. Column 3 and 5 give the absolute coefficient errors for each true functional, $|(\hat{\xi} - \xi)/\xi|$ as percentage ("*inf*" indicates a missed true functional, "$*$" an extra incorrect functional). At 70% noise, Model 1 (which had y^3 instead of y) had clearly inferior validation trajectories, allowing easy identification of Models 0 and 2 as correct.

ODE	Raw Eqn, 70% noise	Error %	Raw Eqn, 100% noise	Error %
Model 0:				
$\dot{x}$	$-0.08x + 2.0y$	(16, 0)	$2.0y - 0.18xy^2$	(*inf*, 0)
$\dot{y}$	$-1.95x - 0.12y$	(2, 17)	$-2.0x - 0.09y$	(0, 11)
Model 1:				
$\dot{x}$	$-0.13x + 1.98y$	(33, 1)	$1.95y$	(*inf* 3)
$\dot{y}$	$-1.99x + 0.01y^3$	(0, *inf*, $*$)	$-1.95x - 0.11y$	(3, 7)
Model 2:				
$\dot{x}$	$-0.11x + 1.94y$	(11, 3)	$1.98y - 0.02xy^2$	(*inf*, 1)
$\dot{y}$	$-2.01x - 0.09y$	(0, 9)	$-1.95x - 0.1y$	(2, 0)

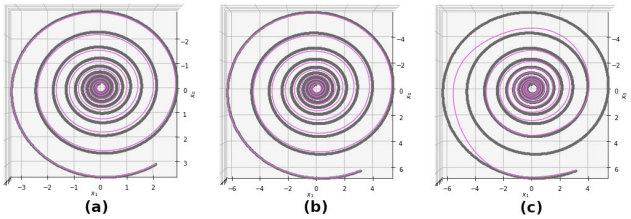


FIGURE 14. Harmonic linear oscillator, test trajectories. (a, b) Typical predictions for Test trajectories 0 and 1, training data with 70% noise. Grey lines are true, fuchsia lines are predictions. (c) Prediction of Test trajectory 1 (as in (b)), training data with 100% noise, showing some degeneration. Test trajectory 0 predictions were highly accurate given training data with 100% noise (very similar to (a)).

predictions deteriorated. The discovered major functionals of $\dot{x}$ and $\dot{y}$, as well as all of $\dot{z}$, remained accurate.

2) LINEAR HARMONIC OSCILLATOR

We consider a two-dimensional harmonic linear oscillator with added white noise equivalent to 70% and 100% (see Fig. 13). Results for typical runs are reported.

The true system has ODEs:

$$\dot{x} = -0.1x + 2y \tag{9}$$
$$\dot{y} = -2x - 0.1\,y \tag{10}$$

Three training trajectories had initial conditions [2, 0], [4, 1], and [7, 1]; and two holdout trajectories had initial conditions [3, 2] and [6, 3]. The initial conditions of the first training trajectory were from [1]; all others were selected at random. Trajectories were 28 seconds long with 0.002 second timestep. The initial functional libraries were all polynomials with degree ≤ 3. Linear dependencies between terms were weak, so the discovered models had no transformed versions.

At 70% noise, the method typically recovered the correct sparse functional libraries and accurate coefficients (Table 6), and gave accurate predictions for all trajectories (train, validation, and test).

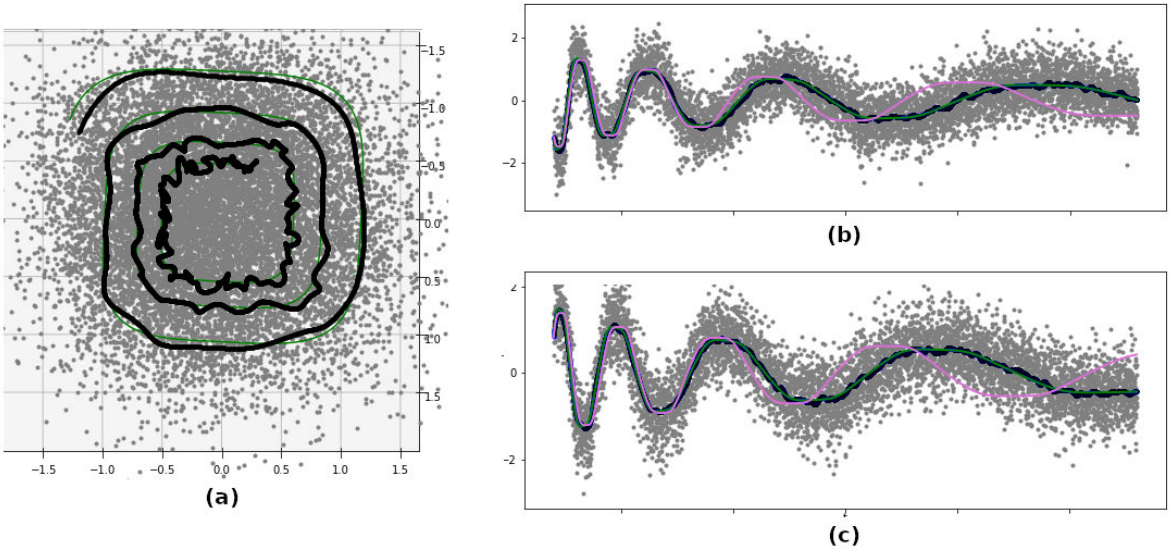


FIGURE 15. Harmonic cubic oscillator, 70% noise. (a) Time-series in x-y plane, noisy (grey dots), clean (green) and smoothed with squiggle artifacts (black). (b) Typical x time-series, with prediction of training trajectory. (c) x time-series, with prediction of validation trajectory. Grey dots are trajectories with added noise. Black lines are smoothed trajectories. Fuschia lines are the trajectories as predicted by discovered models.

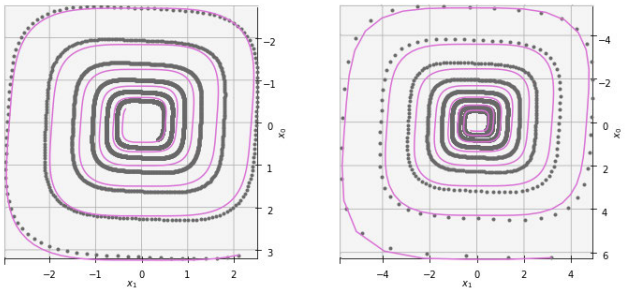


FIGURE 16. Harmonic cubic oscillator Test trajectories. Predicted test trajectories using a typical discovered model trained on data with 70% noise. Grey dots are test trajectories (subsampled). Fuschia lines are the predicted trajectories.

At 100% noise, the discovered models lost the x term in $\dot{x}$, keeping instead xy^2. All other functionals were correct, with accurate coefficients (Table 6), and predicted trajectories (train, validation, and test) were also accurate.

3) HARMONIC CUBIC OSCILLATOR

We consider a two-dimensional harmonic cubic oscillator with added white noise equivalent to 70% (see Fig. 15). Results for typical runs are reported.

The true system has ODEs:

$$\dot{x} = -0.1\,x^3 + 2\,y^3 \tag{11}$$
$$\dot{y} = -2x^3 - 0.1\,y^3 \tag{12}$$

We used three training trajectories with initial conditions [2, 0], [4, 1], and [7, 1]; and two holdout trajectories with initial conditions [3, 2] and [6, 3]. The initial conditions of the first training trajectory are from [1]; all others were selected at random. Trajectories were 28 seconds long with 0.002 second timestep. The initial functional library was all polynomials with degree ≤ 5.

At 70% noise, the method typically recovered the correct sparse functional libraries and accurate coefficients for $\dot{y}$, but missed the minority term x^3 in $\dot{x}$, keeping instead various 5^{th} order terms which had strong linear dependencies ($R^2 \approx 0.88$ to 0.92). See Table 7. The train, validation, and test trajectories were largely accurate (see Figs 15 and 16).

At 100% noise, some models yielded correct functional libraries but results were overall unreliable, as many models included 5^{th} order terms.

4) HOPF NORMAL 2D

We consider a two-dimensional Hopf Normal form (using the identity $z = x^2 + y^2$), as described in [1] with added white noise equivalent to 70%. Results for typical runs are reported.

The true system has ODEs:

$$\dot{x} = 0.2\,x + y - x(x^2 + y^2) = 0.2x + y - x^3 - xy^2 \tag{13}$$
$$\dot{y} = x + 0.2\,y - y(x^2 + y^2) = x + 0.2y - x^2y - y^3 \tag{14}$$

We used three training trajectories with initial conditions [1, 0.75], [0.9, −0.1], and [0.25, 1]; and two holdout trajectories with initial conditions [0.1, −0.75], [0.5, −0.5]. The initial conditions of the first training trajectory are from [1]; all others were selected at random. Trajectories were 16 seconds long with 0.002 second timestep. The initial functional library was all polynomials with degree ≤ 5. The discovered models have two salient features.

First, the toolkit discovered apparently incorrect libraries that were equivalent (via strong linear dependencies) to highly accurate libraries. Examples of relevant linear dependencies are shown in Fig. 8. The equivalent models had

TABLE 7. Harmonic cubic oscillator, 70% noise: Discovered and equivalent models, and absolute coefficient errors for each true functional, $|(\hat{\xi} - \xi)/\xi|$ as percentage ("*inf*" indicates a missed true functional, "$*$" an extra incorrect functional). "Raw" errors are for the discovered equation, "closest" errors are for the transformed equation (cf section II-G).

ODE	Raw Eqn	Closest Eqn	Raw Err %	Closest Err %
Model 0				
$\dot{x}$	$-0.24xy^2 + 2.04y^3$	same	(*inf*, 2, $*$)	(*inf*, 2, $*$)
$\dot{y}$	$-2.03x^3 - 0.12y^3$	same	(6, 20)	(6, 20)
Model 1				
$\dot{x}$	$-0.28xy^2 + 2.18y^3$	same	(*inf*, 9, $*$)	(*inf*, 9, $*$)
$\dot{y}$	$-1.95x^3 - 0.15y^3$	same	(3, 49)	(3, 49)
Model 2				
$\dot{x}$	$1.75y^3 + 0.33x^4y + 0.17y^5$	$-0.01x^3 + 1.98y^3 + 0.33x^4y$	(*inf*, 12, $*$, $*$)	(96, 1, $*$)
$\dot{y}$	$-2.01x^3 - 0.07y^3$	same	(1, 26)	(1, 26)

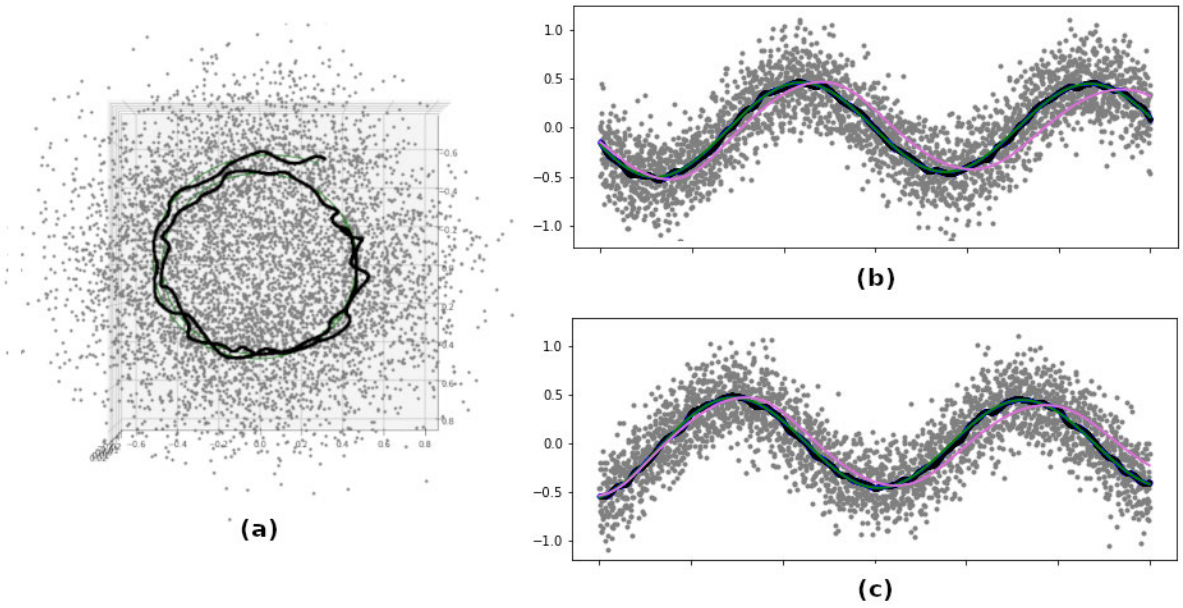


FIGURE 17. Hopf Normal form, 2-D, 70% noise, training set. (a)Time-series in *x*-*y* plane, noisy (grey dots), clean (green) and smoothed with squiggle artifacts (black). (b, c) Typical *x* time-series, with predictions for two validation trajectories. Grey dots are trajectories with added noise. Black lines are smoothed trajectories. Fuschia lines are the trajectories as predicted by discovered models.

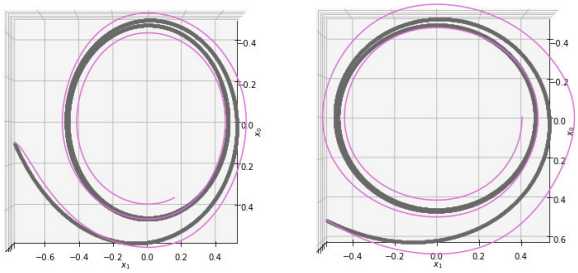


FIGURE 18. Hopf Normal form, 2-D, test set predictions. Predicted test trajectories using a typical discovered model trained on data with 70% noise. Grey dots are test trajectories. Fuschia lines are the predicted trajectories. The trajectories are qualitatively correct, but suffer from a phase offset.

highly similar evolved trajectory behavior, correct functional libraries, and often very accurate coefficient estimates (median 8% error for $\dot{y}$, though median 78% error for $\dot{x}$). Coefficient errors for discovered and closest (via linear dependence) models are given in Table 8. The equivalence of an apparently almost entirely wrong discovered model and a linear transformation very close to the "true" model highlights the importance of the linear dependence assessment method (II-G).

Second, the trajectories evolved by the best models (as judged on validation trajectory FoMs and evolutions) matched the true trajectories (train, validation, and test) well though not perfectly (see Figs 17, 18). This disconnect between correct equation form and correct predictive ability highlights the distinction between the goals of inference (identifying correct functional libraries), and prediction (producing models that behave correctly).

5) HOPF NORMAL 3D

The three-dimensional version of the Hopf Normal form (described in [1]) gave the toolkit considerable trouble. This system includes both a transient portion and a steady-state on an attractor where $z =$ constant. At only 20% added noise the models discovered by the toolkit were somewhat poor: They only partially captured correct functionals, with spurious functionals and inaccurate coefficients; they had strong

TABLE 8. Hopf Normal form, 2-D version, 70% noise: Discovered models and their coefficient errors are given in the top half of the table; closest equivalent models (cf section II-G) and their errors are given in the bottom half. The table gives absolute coefficient errors for each true functional, $|(\hat{\xi} - \xi)/\xi|$ as percentage ("*inf*" indicates a missed true functional, "$*$" an extra incorrect functional). "Raw" errors are for the discovered equations, "closest" errors are for the transformed equations.

ODE	Raw Eqn	Raw Err %
(True:)		
$\dot{x}$	$0.2x + y - x^3 - xy^2$	0
$\dot{y}$	$x + 0.2y - x^2y - y^3$	0
Model 0		
$\dot{x}$	$-0.9y - 1.18xy^2 - 1.16y^3 + 4.66y^5$	(*inf*, 190, *inf*, 18, $*$, $*$)
$\dot{y}$	$0.96x$	(4, *inf*, *inf*, *inf*)
Model 1		
$\dot{x}$	$-1.31y + 1.9y^3$	(*inf*, 231, *inf*, *inf*, $*$)
$\dot{y}$	$1.06x + -0.06y - 0.54x^3$	(6, 131, *inf*, *inf*, $*$)
Model 2		
$\dot{x}$	$-1.34y - 0.22xy^2 + 2.18y^3$	(*inf*, 234, *inf*, 78, $*$)
$\dot{y}$	$0.98x - 0.27y^3 - 0.44x^3$	(2, *inf*, *inf*, 73, $*$)
ODE	Closest Eqn	Closest Err %
Model 0		
$\dot{x}$	$0.05x + -0.9y - 0.21x^3 - 1.42xy^2 - 1.16y^3 + 4.66y^5$	(75, 190, 79, 42, $*$, $*$)
$\dot{y}$	$0.96x + 0.21y - 0.88x^2y - 0.97y^3$	(4, 4, 12, 3)
Model 1		
$\dot{x}$	$0.05x - 1.32y - 0.22x^3 - 0.23xy^2 + 1.9y^3$	(75, 232, 78, 77, $*$)
$\dot{y}$	$1.06x + 0.18y - 0.54x^3 - 1.11x^2y - 1.04y^3$	(6, 10, 11, 4)
Model 2		
$\dot{x}$	$0.05x - 1.34y - 0.21x^3 - 0.47xy^2 + 2.18y^3$	(75, 234, 79, 53, $*$)
$\dot{y}$	$0.98x + 0.18y - 0.77x^2y - 1.13y^3 - 0.44x^3$	(2, 10, 23, 13, $*$)

training trajectory predictions but poor validation trajectory predictions; and they failed to predict test trajectories.

REFERENCES

[1] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 15, pp. 3932–3937, 2015.

[2] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge, U.K.: Cambridge Univ. Press, 2019.

[3] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Sci. Adv.*, vol. 3, no. 4, Apr. 2017, Art. no. e1602614.

[4] H. Schaeffer, "Learning partial differential equations via data discovery and sparse optimization," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 473, no. 2197, Jan. 2017, Art. no. 20160446.

[5] L. Zhang and H. Schaeffer, "On the convergence of the SINDy algorithm," *Multiscale Model. Simul.*, vol. 17, no. 3, pp. 948–972, Jan. 2019.

[6] K. Champion, P. Zheng, A. Y. Aravkin, S. L. Brunton, and J. N. Kutz, "A unified sparse optimization framework to learn parsimonious physics-informed models from data," *IEEE Access*, vol. 8, pp. 169259–169271, 2020.

[7] J.-C. Loiseau and S. L. Brunton, "Constrained sparse Galerkin regression," *J. Fluid Mech.*, vol. 838, pp. 42–67, Mar. 2018.

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

[9] J.-C. Loiseau, "Data-driven modeling of the chaotic thermal convection in an annular thermosyphon," *Theor. Comput. Fluid Dyn.*, vol. 34, no. 4, pp. 339–365, Aug. 2020.

[10] Y. Guan, S. L. Brunton, and I. Novosselov, "Sparse nonlinear models of chaotic electroconvection," *Roy. Soc. Open Sci.*, vol. 8, no. 8, Aug. 2021, Art. no. 202367.

[11] N. Deng, B. R. Noack, M. Morzyński, and L. R. Pastur, "Galerkin force model for transient and post-transient dynamics of the fluidic pinball," *J. Fluid Mech.*, vol. 918, pp. 1–36, Jul. 2021.

[12] J. L. Callaham, G. Rigas, J.-C. Loiseau, and S. L. Brunton, "An empirical mean-field model of symmetry-breaking in a turbulent wake," 2021, *arXiv:2105.13990*.

[13] J. L. Callaham, S. L. Brunton, and J.-C. Loiseau, "On the role of nonlinear correlations in reduced-order modeling," 2021, *arXiv:2106.02409*.

[14] J. Zhang and W. Ma, "Data-driven discovery of governing equations for fluid dynamics based on molecular simulation," *J. Fluid Mech.*, vol. 892, pp. 1–18, Jun. 2020.

[15] M. Dam, M. Brøns, J. J. Rasmussen, V. Naulin, and J. S. Hesthaven, "Sparse identification of a predator-prey system from simulation data of a convection model," *Phys. Plasmas*, vol. 24, no. 2, Feb. 2017, Art. no. 022310.

[16] A. A. Kaptanoglu, K. D. Morgan, C. J. Hansen, and S. L. Brunton, "Physics-constrained, low-dimensional models for magnetohydrodynamics: First-principles and data-driven approaches," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 104, no. 1, Jul. 2021, Art. no. 015206.

[17] S. Beetham and J. Capecelatro, "Formulating turbulence closures using sparse regression with embedded form invariance," *Phys. Rev. Fluids*, vol. 5, no. 8, Aug. 2020, Art. no. 084611.

[18] S. Beetham, R. O. Fox, and J. Capecelatro, "Sparse identification of multiphase turbulence closures for coupled fluid–particle flows," *J. Fluid Mech.*, vol. 914, pp. 1–23, May 2021.

[19] M. Schmelzer, R. P. Dwight, and P. Cinnella, "Discovery of algebraic Reynolds-stress models using sparse symbolic regression," *Flow, Turbulence Combustion*, vol. 104, nos. 2–3, pp. 579–603, Mar. 2020.

[20] M. Sorokina, S. Sygletos, and S. Turitsyn, "Sparse identification for nonlinear optical communication systems: SINO method," *Opt. Exp.*, vol. 24, no. 26, pp. 30433–30443, Dec. 2016.

[21] S. Thaler, L. Paehler, and N. A. Adams, "Sparse identification of truncation errors," *J. Comput. Phys.*, vol. 397, Nov. 2019, Art. no. 108851.

[22] K. Kaheman, E. Kaiser, B. Strom, J. N. Kutz, and S. L. Brunton, "Learning discrepancy models from experimental data," 2019, *arXiv:1909.08574*.

[23] B. M. de Silva, D. M. Higdon, S. L. Brunton, and J. N. Kutz, "Discovery of physics from data: Universal laws and discrepancies," 2019, *arXiv:1906.07906*.

[24] D. E. Shea, S. L. Brunton, and J. N. Kutz, "SINDy-BVP: Sparse identification of nonlinear dynamics for boundary value problems," *Phys. Rev. Res.*, vol. 3, no. 2, Jun. 2021, Art. no. 023255.

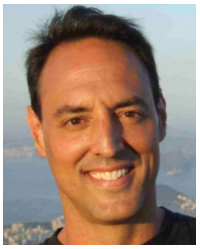
[25] K. P. Champion, S. L. Brunton, and J. N. Kutz, "Discovery of nonlinear multiscale systems: Sampling strategies and embeddings," *SIAM J. Appl. Dyn. Syst.*, vol. 18, no. 1, pp. 312–333, Jan. 2019.

[26] J. J. Bramburger and J. N. Kutz, "Poincaré maps for multiscale physics discovery and nonlinear floquet theory," *Phys. D, Nonlinear Phenomena*, vol. 408, Jul. 2020, Art. no. 132479.

[27] J. J. Bramburger, J. N. Kutz, and S. L. Brunton, "Data-driven stabilization of periodic orbits," *IEEE Access*, vol. 9, pp. 43504–43521, 2021.

[28] P. Gelß, S. Klus, J. Eisert, and C. Schütte, "Multidimensional approximation of nonlinear dynamical systems," *J. Comput. Nonlinear Dyn.*, vol. 14, no. 6, Jun. 2019.
[29] L. Boninsegna, F. Nüske, and C. Clementi, "Sparse learning of stochastic dynamical equations," *J. Chem. Phys.*, vol. 148, no. 24, Jun. 2018, Art. no. 241723.
[30] J. L. Callaham, J.-C. Loiseau, G. Rigas, and S. L. Brunton, "Nonlinear stochastic modelling with Langevin regression," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 477, no. 2250, Jun. 2021, Art. no. 20210092.
[31] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proc. Nat. Acad. Sci. USA*, vol. 116, no. 45, pp. 22445–22451, Nov. 2019.
[32] M. Kalia, S. L. Brunton, H. G. E. Meijer, C. Brune, and J. N. Kutz, "Learning normal form autoencoders for data-driven discovery of universal,parameter-dependent governing equations," 2021, *arXiv:2106.05102*.
[33] H. Schaeffer and S. G. McCalla, "Sparse model selection via integral terms," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 96, no. 2, Aug. 2017, Art. no. 023302.
[34] P. A. K. Reinbold, D. R. Gurevich, and R. O. Grigoriev, "Using noisy or incomplete data to discover models of spatiotemporal dynamics," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 101, no. 1, Jan. 2020, Art. no. 010203.
[35] D. R. Gurevich, P. A. K. Reinbold, and R. O. Grigoriev, "Robust and optimal sparse regression for nonlinear PDE models," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 29, no. 10, Oct. 2019, Art. no. 103113.
[36] E. P. Alves and F. Fiuza, "Data-driven discovery of reduced plasma physics models from fully-kinetic simulations," 2020, *arXiv:2011.01927*.
[37] P. A. K. Reinbold, L. M. Kageorge, M. F. Schatz, and R. O. Grigoriev, "Robust learning from noisy, incomplete, high-dimensional experimental data via physically constrained symbolic regression," *Nature Commun.*, vol. 12, no. 1, pp. 1–8, Dec. 2021.
[38] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, "PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data," *J. Open Source Softw.*, vol. 5, no. 49, p. 2104, May 2020.
[39] A. Kaptanoglu, B. de Silva, U. Fasel, K. Kaheman, A. Goldschmidt, J. Callaham, C. Delahunt, Z. Nicolaou, K. Champion, J.-C. Loiseau, J. Kutz, and S. Brunton, "PySINDy: A comprehensive Python package for robust sparse system identification," *J. Open Source Softw.*, vol. 7, no. 69, p. 3994, Jan. 2022, doi: 10.21105/joss.03994.
[40] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics with control (SINDYc)," *IFAC Nolcos*, vol. 49, no. 18, pp. 710–715, 2016.
[41] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," 2017, *arXiv:1711.05501*.
[42] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Inferring biological networks by sparse identification of nonlinear dynamics," *IEEE Trans. Mol., Biol. Multi-Scale Commun.*, vol. 2, no. 1, pp. 52–63, Jun. 2016.
[43] K. Kaheman, J. N. Kutz, and S. L. Brunton, "SINDy-PI: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 476, no. 2242, Oct. 2020, Art. no. 20200279.
[44] G. Tran and R. Ward, "Exact recovery of chaotic systems from highly corrupted data," 2016, *arXiv:1607.01067*.
[45] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor, "Model selection for dynamical systems via sparse regression and information criteria," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 473, no. 2204, 2017, Art. no. 20170009.
[46] S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz, "Chaos as an intermittently forced linear system," *Nature Commun.*, vol. 8, no. 1, pp. 1–9, Dec. 2017.
[47] W. Su, M. Bogdan, and E. Candes, "False discoveries occur early on the lasso path," 2015, *arXiv:1511.01957*.
[48] S. M. Hirsh, D. A. Barajas-Solano, and J. N. Kutz, "Sparsifying priors for Bayesian uncertainty quantification in model discovery," 2021, *arXiv:2107.02107*.
[49] S. Zhang and G. Lin, "Robust data-driven discovery of governing physical laws with error bars," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 474, Sep. 2018, Art. no. 20180305.
[50] M. Quade, M. Abel, J. Nathan Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for rapid model recovery," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 28, no. 6, Jun. 2018, Art. no. 063116.
[51] P. J. Rousseeuw, "Least median of squares regression," *J. Amer. Statist. Assoc.*, vol. 79, no. 388, pp. 871–880, 1984.
[52] F. Lejarza and M. Baldea, "DySMHO: Data-driven discovery of governing equations for dynamical systems via moving horizon optimization," 2021, *arXiv:2108.00069*.
[53] S. H. Rudy, S. L. Brunton, and J. N. Kutz, "Smoothing and parameter estimation by soft-adherence to governing equations," *J. Comput. Phys.*, vol. 398, Dec. 2019, Art. no. 108860.
[54] K. Ahnert and M. Abel, "Numerical differentiation of experimental data: Local versus global methods," *Comput. Phys. Commun.*, vol. 177, no. 10, pp. 764–774, Nov. 2007.
[55] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proc. Nat. Acad. Sci. USA*, vol. 116, no. 45, pp. 22445–22451, Nov. 2019.
[56] B. C. Daniels and I. Nemenman, "Automated adaptive inference of phenomenological dynamical models," *Nature Commun.*, vol. 6, no. 1, pp. 1–8, Nov. 2015.
[57] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 474, no. 2219, Nov. 2018, Art. no. 20180335.
[58] C. B. Delahunt and J. N. Kutz. (2021). *Codebase for High Noise SINDy Toolkit*. [Online]. Available: github.com/charlesDelahunt/sindyToolkitForHighNoise
[59] 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," 2021, *arXiv:2111.10992*.
[60] P. Virtanen *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, Mar. 2020.
[61] S. Pan, N. Arnold-Medabalimi, and K. Duraisamy, "Sparsity-promoting algorithms for the discovery of informative Koopman-invariant subspaces," *J. Fluid Mech.*, vol. 917, pp. 1–49, Jun. 2021.

CHARLES B. DELAHUNT (Member, IEEE) received the Ph.D. degree in electrical engineering, in 2018, and recently finished a Postdoctoral Researcher with the Department of Applied Mathematics, University of Washington, Seattle, WA, USA, with work focused on machine learning. He is currently a senior research scientist at Global Health Labs, Bellevue, WA, USA, applying machine learning to health care challenges in low- and middle-income countries.



J. NATHAN KUTZ (Senior Member, IEEE) received the B.S. degrees in physics and mathematics from the University of Washington, Seattle, WA, USA, and the Ph.D. degree in applied mathematics from Northwestern University, Evanston, IL, USA, in 1994. At the University of Washington, he is currently a Professor of applied mathematics and electrical engineering; an Adjunct Professor of physics and mechanical engineering, and electrical engineering; a Senior Data Science Fellow with the eScience Institute; and the Director of the NSF AI Institute for Dynamic Systems.