

Extracting structured dynamical systems using sparse optimization with very few samples

Hayden Schaeffer^{*1}, Giang Tran^{†2}, Rachel Ward^{‡3} and Linan Zhang^{§1}

¹Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA.

²Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada.

³Department of Mathematics, The University of Texas at Austin, Austin, TX.

Abstract

Learning governing equations allows for deeper understanding of the structure and dynamics of data. We present a random sampling method for learning structured dynamical systems from under-sampled and possibly noisy state-space measurements. The learning problem takes the form of a sparse least-squares fitting over a large set of candidate functions. Based on a Bernstein-like inequality for partly dependent random variables, we provide theoretical guarantees on the recovery rate of the sparse coefficients and the identification of the candidate functions for the corresponding problem. Computational results are demonstrated on datasets generated by the Lorenz 96 equation, the viscous Burgers' equation, and the two-component reaction-diffusion equations (which is challenging due to parameter sensitivities in the model). This formulation has several advantages including ease of use, theoretical guarantees of success, and computational efficiency with respect to ambient dimension and number of candidate functions.

Keywords— High Dimensional Systems, Dynamical Systems, Sparsity, Model Selection, Exact Recovery, Cyclic Permutation, Coherence

1 Introduction

Automated model selection is an important task for extracting useful information from observed data. One focus is to create methods and algorithms which allow for the data-based identification of governing equations that can then be used for more detailed analysis of the underlying structure and dynamics. The overall goal is to develop computational tools for reverse engineering equations from data. Automated model selection has several advantages over manual processing, since algorithmic

*schaeffer@cmu.edu

†giang.tran@uwaterloo.ca

‡rward@math.utexas.edu

§linanz@andrew.cmu.edu

approaches allow for the inclusion of a richer set of potential candidate functions, which thus allow for complex models and can be used to process and fit larger data sets. However, several factors restrict the computational efficiency; for example, as the dimension of the variables grows, the size of the set of possible candidate functions grows rapidly. In this work, we present some computational strategies for extracting governing equation when additional structural information of the data is known.

Data-driven methods for model selection have several recent advances. The authors of [2, 36] developed an approach for extracting physical laws (*i.e.* equations of motion, energy, *etc.*) from experimental data. The method uses a symbolic regression algorithm to fit the derivatives of the time-series data to the derivatives of candidate functions while taking into account accuracy versus simplicity. In [4], the authors proposed a sparse approximation approach for selecting governing equations from data. One of the key ideas in [4] is to use a fixed (but large and possibly redundant) set of candidate functions in order to write the model selection problem as a linear system. The sparse solutions of this system are those that are likely to balance simplicity of the model while preserving accuracy. To find a sparse approximation, [4] uses a sequential least-squares thresholding algorithm which includes a thresholding sub-step (sparsification) and a least-squares sub-problem (fitting). Several sparsity-based methods were developed in order to solve the model selection problem. The authors of [40] proposed an optimization problem for extracting the governing equation from chaotic data with highly corrupted segments (of unknown location and length). Their approach uses the $\ell^{2,1}$ -norm (often referred to as the group sparse or joint sparse penalty), in order to detect the location of the corruption, coupling each of the state-variables. In [40], it was proven that, for chaotic systems, the solution to the optimization problem will locate the noise-free regions and thus extract the correct governing system. In [30], the authors used a dictionary of partial derivatives along with a LASSO-based approach [38] to extract the partial differential equation that governs some (possibly noisy) spatiotemporal dataset. An adaptive ridge-regression version of the method from [4] was proposed in [29] and applied to fit PDE to spatiotemporal data. One computational issue that arises in these approaches is that noise on the state-variables are amplified by numerical differentiation. To lessen the effects of noise, sparse learning can be applied to the integral formulation of the differential equation along with an integrated candidate set as done in [33]. The exact recovery of the governing equation can be guaranteed when there is sufficient randomness in the data, even in the under-sampling limit. In [34], using random initial conditions, an ℓ^1 -penalized optimization problem was shown to recover the underlying differential equation with high probability, and several sampling strategies were discussed. In order to allow for variations in the coefficients, a group-sparse recovery model was proposed in [35]. Using information criteria, [21] proposed a method to choose the “optimal” model learned from the algorithm in [4] as one varies the thresholding parameter.

There have been several recent methods using general sparse approximations techniques for learning governing dynamical systems, including: SINDy with control [5], the SINO method [37], an extension of SINDy to stochastic dynamics [3], sparse identification of a predator-prey system [9], SINDy with rational candidate functions [20], rapid-SINDy [24], the unified sparse dynamics learning (USDL) algorithm which uses a weak formulation with the orthogonal matching pursuit

(OMP) algorithm [22]. Sparsity inducing and/or data-driven algorithms have been applied to other problems in scientific computing, including: sparse spectral methods for evolution equations [31, 19], sparse penalties for obstacle problems [39], sparse-low energy decomposition for conversation laws [13], sparse spatial approximations for PDE [6], sparse weighted interpolation of functions [28], leveraging optimization algorithms in nonlinear PDE [32], sparse approximation for polynomial chaos [23], high-dimensional function approximation using sparsity in lower sets [1], learning PDE through convolutional neural nets [17], modeling dynamics through Gaussian processes [26, 25], and constrained Galerkin methods [16].

1.1 Contributions of this Work.

We present an approach for recovering governing equations from under-sampled measurements using the burst sampling methodology in [34]. In this work, we show that if the components of the governing equations are similar, *i.e.* if each of the equations contains the same active terms (after permutation of the indices), then one can recover the coefficients and identify the active basis terms using fewer random samples than required in [34]. The problem statement and construction of the permuted data and dictionaries are detailed in Section 2. In essence, after permutation, the dictionary matrix is still sufficiently random and maintains the necessary properties for exact and stable sparse recovery. Theoretical guarantees on the recovery rate of the coefficients and identification of the candidate functions (the support set of the coefficients) are provided in Section 3. The proofs rely on a Bernstein-like inequality for partially dependent measurements. The algorithm uses the Douglas-Rachford iteration to solve the ℓ^1 penalized least-squares problem. In Section 4, the algorithm and the data processing are explained¹. In Section 5, several numerical results are presented, including learning the Lorenz 96 system, the viscous Burgers' equation, and a two-component reaction-diffusion system. These examples include third-order monomials, which extend the computational results of [34]. These problems are challenging due to their sensitivities of the dynamics to parameters, for example, shock locations or the structure of patterns. Our approach is able to recover the dynamics with high probability, and in some cases, we can recover the governing dynamical system from one-sample!

2 Problem Statement

Consider an evolution equation $\dot{u} = f(u)$, where $u(t) \in \mathbb{R}^n$ and the initial data is $u(t_0) = u_0$. Assume that f is a polynomial vector-valued equation in u . The evolution equation can be written component-wise as:

$$\begin{cases} \dot{u}_1 = f_1(u_1, \dots, u_n) \\ \dot{u}_2 = f_2(u_1, \dots, u_n) \\ \vdots \\ \dot{u}_n = f_n(u_1, \dots, u_n). \end{cases}$$

¹The code is available on <https://github.com/linanzhang/SparseCyclicRecovery>.

From limited measurements on the state-space u , the objective is to extract the underlying model f . In [34], this problem was investigated for general (sparse) polynomials f using several random sampling strategies, including a burst construction. The data is assumed to be a collection of K -bursts, *i.e.* a short-time trajectory:

$$\{u(t_1; k), u(t_2; k), \dots, u(t_{m-1}; k)\},$$

associated with some initial data $u(t_0; k)$, where $u(\cdot; k)$ denotes the k -th burst, $1 \leq k \leq K$. In addition, we assume that the time derivative associated with each of the measurements in a burst, denoted by $\{\dot{u}(t_0; k), \dot{u}(t_1; k), \dot{u}(t_2; k), \dots, \dot{u}(t_{m-1}; k)\}$, can be accurately approximated. Define the matrix M as the collection of all monomials (stored column-wise),

$$M(k) = \left[\begin{array}{c|c|c|c} M^{(0)}(k) & M^{(1)}(k) & M^{(2)}(k) & \dots \end{array} \right],$$

where the sub-matrices are the collections of the constant, linear, quadratic terms and so on,

$$M^{(0)}(k) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^m,$$

$$M^{(1)}(k) = \begin{pmatrix} u_1(t_0; k) & u_2(t_0; k) & \cdots & u_n(t_0; k) \\ u_1(t_1; k) & u_2(t_1; k) & \cdots & u_n(t_1; k) \\ u_1(t_2; k) & u_2(t_2; k) & \cdots & u_n(t_2; k) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_{m-1}; k) & u_2(t_{m-1}; k) & \cdots & u_n(t_{m-1}; k) \end{pmatrix},$$

and

$$M^{(2)}(k) = \begin{pmatrix} u_1^2(t_0; k) & u_1(t_0; k)u_2(t_0; k) & \cdots & u_n^2(t_0; k) \\ u_1^2(t_1; k) & u_1(t_1; k)u_2(t_1; k) & \cdots & u_n^2(t_1; k) \\ u_1^2(t_2; k) & u_1(t_2; k)u_2(t_2; k) & \cdots & u_n^2(t_2; k) \\ \vdots & \vdots & \ddots & \vdots \\ u_1^2(t_{m-1}; k) & u_1(t_{m-1}; k)u_2(t_{m-1}; k) & \cdots & u_n^2(t_{m-1}; k) \end{pmatrix}.$$

Define the velocity matrix at each of the corresponding measurements:

$$v(k) = \begin{pmatrix} \dot{u}_1(t_0; k) & \dot{u}_2(t_0; k) & \cdots & \dot{u}_n(t_0; k) \\ \dot{u}_1(t_1; k) & \dot{u}_2(t_1; k) & \cdots & \dot{u}_n(t_1; k) \\ \dot{u}_1(t_2; k) & \dot{u}_2(t_2; k) & \cdots & \dot{u}_n(t_2; k) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{u}_1(t_{m-1}; k) & \dot{u}_2(t_{m-1}; k) & \cdots & \dot{u}_n(t_{m-1}; k) \end{pmatrix}.$$

Using this data, one would like to extract the coefficients for each of the components of f , *i.e.* f_j , which is denoted by a column vector c_j . The collection of coefficients can be defined as:

$$C = \begin{pmatrix} | & | & \cdots & | \\ c_1 & c_2 & \cdots & c_n \\ | & | & \cdots & | \end{pmatrix}. \quad (1)$$

Therefore, the problem of identifying the sparse polynomial coefficients associated with the model f is equivalent to finding a sparse matrix C such that $v(k) = M(k)C$ for all bursts k .

In [34], it was shown that finding a sparse matrix C from $v(k) = M(k)C$ with randomly sampled initial data was achievable using an ℓ^1 optimization model. In particular, with probability $1 - \epsilon$, C can be recovered exactly from limited samples as long as the number of samples K satisfies $K \sim s \log(N) \log(\epsilon^{-1})$, where s is the maximum sparsity level among the n columns of C and N is the number of basis functions. In this work, using a coherence bound, we get a **sampling rate that scales like $n^{-1} \log(n)$ when the governing equation has a structural condition** relating each of the components of the model f .

2.1 A Cyclic Condition

When structural conditions on f and u can be assumed *a priori*, one expects the number of initial samples needed for exact recovery to decrease. One common assumption is that the components of the model, f_j , are cyclic (or index-invariant), *i.e.* for all $1 \leq i, j \leq n$, we have:

$$f_j(u_1, u_2, \dots, u_n) = f_i(u_{j-i+1}, u_{j-i+2}, \dots, u_n, u_1, \dots, u_{j-i+n}),$$

where $u_{n+q} = u_q$ and $u_{-q} = u_{n-q}$, for all $0 \leq q \leq (n-1)$. In particular, all components f_j can be obtained by determining just one component, say f_1 , since:

$$f_j(u_1, u_2, \dots, u_n) = f_1(u_j, u_{j+1}, \dots, u_n, u_1, \dots, u_{j-1+n}).$$

The goal is to determine f (by learning f_1), given observations of u and an (accurate) approximation of \dot{u} .

The physical meaning behind the cyclic condition relates to the invariance of a model to location/position. For example, the Lorenz 96 system in $n > 3$ dimensions is given by [18]:

$$\dot{u}_j = -u_{j-2} u_{j-1} + u_{j-1} u_{j+1} - u_j + F, \quad j = 1, 2, \dots, n,$$

for some constant F (independent of j) and with periodic conditions, $u_{-1} = u_{n-1}$, $u_0 = u_n$, and $u_{n+1} = u_1$. Each component of f follows the same structure, and is invariant to the index j . This is also the case with spatiotemporal dynamics which are not directly dependent on the space variable. For a simple example, consider the standard discretization of the heat equation in one

spatial dimensional periodic domain:

$$\dot{u}_j = \frac{1}{h^2} (u_{j-1} - 2u_j + u_{j+1}),$$

with grid size $h > 0$ and periodic conditions, $u_0 = u_n$ and $u_{n+1} = u_1$. The system is invariant to spatial translation, and thus satisfies the cyclic condition.

Extending the results from [34], we show that recovering the governing equations from only one under-sampled measurement is tractable when the model f has this cyclic structure. This is possible since one measurement of $u(t)$ will provide us with n -pieces of information for f_1 (and thus the entire model f). Under this assumption, the problem of determining the coefficient matrix C , defined by Equation (1), reduces to the problem of determining the first column of C , *i.e.* c_1 . For simplicity, we can drop the subscript and look for a coefficient vector $c \in \mathbb{R}^N$ (N is the number of candidate functions) that fits the dynamic data.

The construction of the optimization problem and computational method are detailed in the subsequent sections. To summarize, we define what cyclic permutations are and explain how to build the associated dictionary matrix. This construction is detailed for the one spatial dimensional case, since general spatial dimensions follow from a vectorization of the n -dimensional problem. Following the second strategy in [34], a subset of the domain is considered (via localization and restriction of the dictionary terms), which leads to a smaller, but still underdetermined, problem. The dictionary is transformed to the tensorized Legendre basis in order to guarantee an incoherence principle on the system. Lastly, the coefficients of the governing equations are learned via an ℓ^1 penalized basis pursuit problem with an inexact (noise-robust) constraint.

2.2 Cyclic Permutations

The (multi-sample) data matrix is computed using a set of cyclic permutations from very few samples. The set of cyclic permutations, \mathcal{C}_n , is a subset of all permutations of $[n] := \{0, 1, \dots, n-1\}$, whose elements are shifted by a fixed amount. There are n cyclic permutations out of the $n!$ possible permutations of $[n]$. In addition, the corresponding $n \times n$ permutation matrices of a cyclic permutation are all circulant. For example, \mathcal{C}_3 contains three permutations of the set $\{0, 1, 2\}$ (out of a total of six possible permutations), *i.e.* $\{0, 1, 2\}$, $\{1, 2, 0\}$, and $\{2, 0, 1\}$ whose permutation matrices are:

$$P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad P_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad P_3 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

The importance of the cyclic permutations is that they preserve the natural ordering between elements, since the left and right neighbors of any element are fixed (with periodic conditions at the first and last elements).

2.3 Dictionary for One Spatial Dimension

Consider the sequence (indexed by k) of discrete measurements $\{u(t_1; k), u(t_2; k), \dots, u(t_{m-1}; k)\}$, obtained through either k -simulations or k -observations. Assume that the data is a discretization of a system with one spatial variable $u(t; k) \in \mathbb{R}^n$ for $t \in \mathbb{R}$ and $k \in \mathbb{N}$. For general spatial dimensions, the variables are multidimensional arrays. In particular, after discretization and vectorization, the spatially dependent function $u(t, x)$ is converted to a 1D array (vector), $u(t, x, y)$ is converted to a 2D array (matrix), and $u(t, x, y, z)$ is converted to a 3D array, *etc.*. As in [34], the total number of temporal samples, denoted by m , is small, and thus we refer to the short-time trajectory as a burst. Each of the bursts is initialized by data sampled from a random distribution.

Given one measurement $u(t_0; k) \in \mathbb{R}^n$, we obtain “multiple” measurements by considering the collection of all cyclic permutations of the data vector $u(t_0; k)$. In particular, we can construct the n -measurement matrix,

$$U(t_0; k) = \begin{pmatrix} u_1(t_0; k) & u_2(t_0; k) & \cdots & u_n(t_0; k) \\ u_2(t_0; k) & u_3(t_0; k) & \cdots & u_1(t_0; k) \\ u_3(t_0; k) & u_4(t_0; k) & \cdots & u_2(t_0; k) \\ \vdots & \vdots & \ddots & \vdots \\ u_n(t_0; k) & u_1(t_0; k) & \cdots & u_{n-1}(t_0; k) \end{pmatrix}. \quad (2)$$

To build the dictionary matrix, we collect all monomials of U . The quadratic matrix, denoted by U^2 , is defined as:

$$U^2(t_0; k) = \begin{pmatrix} u_1^2(t_0; k) & u_1(t_0; k)u_2(t_0; k) & \cdots & u_n^2(t_0; k) \\ u_2^2(t_0; k) & u_2(t_0; k)u_3(t_0; k) & \cdots & u_1^2(t_0; k) \\ u_3^2(t_0; k) & u_3(t_0; k)u_4(t_0; k) & \cdots & u_2^2(t_0; k) \\ \vdots & \vdots & \ddots & \vdots \\ u_n^2(t_0; k) & u_n(t_0; k)u_1(t_0; k) & \cdots & u_{n-1}^2(t_0; k) \end{pmatrix}, \quad (3)$$

and the cubic matrix, denoted by U^3 , is defined as:

$$U^3(t_0; k) = \begin{pmatrix} u_1^3(t_0; k) & u_1^2(t_0; k)u_2(t_0; k) & \cdots & u_1(t_0; k)u_2(t_0; k)u_3(t_0; k) & \cdots & u_n^3(t_0; k) \\ u_2^3(t_0; k) & u_2^2(t_0; k)u_3(t_0; k) & \cdots & u_2(t_0; k)u_3(t_0; k)u_4(t_0; k) & \cdots & u_1^3(t_0; k) \\ u_3^3(t_0; k) & u_3^2(t_0; k)u_4(t_0; k) & \cdots & u_3(t_0; k)u_4(t_0; k)u_5(t_0; k) & \cdots & u_2^3(t_0; k) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ u_n^3(t_0; k) & u_n^2(t_0; k)u_1(t_0; k) & \cdots & u_n(t_0; k)u_1(t_0; k)u_2(t_0; k) & \cdots & u_{n-1}^3(t_0; k) \end{pmatrix}. \quad (4)$$

The process continues this way for any higher-order monomial term. The $n \times N$ dictionary matrix (where $N = \binom{n+p}{p}$ is the number of monomials of degree at most p) is given by:

$$A(t_0; k) = \left[\begin{array}{c|c|c|c|c} 1 & U(t_0; k) & U^2(t_0; k) & U^3(t_0; k) & \cdots \end{array} \right], \quad (5)$$

where one augments the matrix from the right by the additional monomial terms. For simplicity, we will consider the cubic case for all examples and results. Note that when $n = 150$, the number of candidate functions N exceeds half a million for the cubic case and over 22 million for the quartic case.

The velocity for the k th burst is given by:

$$V(t_0; k) = \begin{pmatrix} \dot{u}_1(t_0; k) \\ \dot{u}_2(t_0; k) \\ \dot{u}_3(t_0; k) \\ \vdots \\ \dot{u}_n(t_0; k) \end{pmatrix}.$$

Let c be the vector of coefficients, $c = (c_1, c_2, \dots, c_N)^T$. If we use multiple bursts, say k from $1, \dots, K$ and/or multiple snapshots (*i.e.* $m > 1$), then we concatenate the matrices and vectors row-wise as follows,

$$V = \begin{pmatrix} V(t_0; 1) \\ V(t_1; 1) \\ \vdots \\ V(t_{m-1}; 1) \\ V(t_0; 2) \\ V(t_1; 2) \\ \vdots \\ V(t_{m-1}; 2) \\ \vdots \\ V(t_0; K) \\ V(t_1; K) \\ \vdots \\ V(t_{m-1}; K) \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} A(t_0; 1) \\ A(t_1; 1) \\ \vdots \\ A(t_{m-1}; 1) \\ A(t_0; 2) \\ A(t_1; 2) \\ \vdots \\ A(t_{m-1}; 2) \\ \vdots \\ A(t_0; K) \\ A(t_1; K) \\ \vdots \\ A(t_{m-1}; K) \end{pmatrix}. \quad (6)$$

Thus, the linear inverse problem is to find c such that $V = Ac$. The size of the dictionary matrix A is $mnK \times N$. Therefore, for small K and m , this problem will be underdetermined.

We assume that the number of samples is very small. Thus, the initial data provides a large portion of the information in A . The burst is (mainly) used to obtain an approximation to the velocity, so that V is relatively accurate.

2.4 Dictionary for Higher Spatial Dimension

To generalize to higher spatial dimension, the cyclic permutations must be defined for multiple indices. Given the vectorization of a two-dimensional array, $w = \text{vec}(W)$, where $W = [W_{i,j}]$ for

$1 \leq i, j \leq n$, we must permute w with respect to cyclic permutations of the two-dimensional array W . A permutation of an array preserves the cyclic structural condition if it is a cyclic permutation of both the rows and the columns. In particular, a cyclic permutation of $W \in \mathbb{R}^{n \times n}$ is equivalent to sending each element $W_{i,j}$ to $W_{\gamma(i),\tau(j)}$ for $\gamma, \tau \in \mathcal{C}_n$. In order to combine the n^2 -permuted arrays, each permutation is vectorized and stored row-wise:

$$U(t_0; k) = [\text{vec}(u_{\gamma(i),\tau(j)})(t_0; k)]. \quad (7)$$

As an example, consider $u(t_0; k) \in \mathbb{R}^{3 \times 3}$, where:

$$u(t_0; k) = \begin{pmatrix} u_{1,1}(t_0; k) & u_{1,2}(t_0; k) & u_{1,3}(t_0; k) \\ u_{2,1}(t_0; k) & u_{2,2}(t_0; k) & u_{2,3}(t_0; k) \\ u_{3,1}(t_0; k) & u_{3,2}(t_0; k) & u_{3,3}(t_0; k) \end{pmatrix}.$$

One cyclic permutation of $u(t_0; k)$ is to take rows $\{1, 2, 3\}$ to $\{2, 3, 1\}$ and columns $\{1, 2, 3\}$ to $\{3, 1, 2\}$,

$$\tilde{u}(t_0; k) = \begin{pmatrix} u_{2,3}(t_0; k) & u_{2,1}(t_0; k) & u_{2,2}(t_0; k) \\ u_{3,3}(t_0; k) & u_{3,1}(t_0; k) & u_{3,2}(t_0; k) \\ u_{1,3}(t_0; k) & u_{1,1}(t_0; k) & u_{1,2}(t_0; k) \end{pmatrix}.$$

This construction has the additional benefit of not repeating elements in each row or column. The higher-order monomials and the corresponding dictionary and velocity matrices (A and V , respectively, defined by Equation (6)) are built as before using Equation (7) as the input data. As we increase the spatial dimension, the number of candidate functions grows; for example for $n = 15$, the number of cubic candidate functions in two spatial dimensions is nearly two million!

For a general spatial dimension n , the process above is repeated, where one constructs all permutations of the n -dimensional array $u(t_0; k)$ by applying cyclic permutations to each of the coordinates separately. Each of the permuted n -dimensional arrays are vectorized and collected (row-wise) as is done in Equation (7). The dictionary and velocity matrices are constructed as above.

2.5 Restriction of the Data and Localization of the Dictionary Matrix

The dictionary matrix construction in the previous sections relies on the cyclic permutation of the input. One may restrict the learning algorithm to a subset of the data and also localize the basis to a patch in the domain. This is advantageous, for example, when only a subset of the data is known to be accurate enough to approximate the velocity or when the initial data is only sufficiently random in a subset of the domain.

The process of restricting the data and localizing the basis are technically distinct. The restriction to a subdomain will always be slightly larger than the localization of the basis terms. To illustrate, consider the one-dimensional system with $n > 9$ points. We localize the basis by assuming that the equation for the j th component, say u_j , only depends on monomial terms u_i for $i \in [j - 2, j + 2]$. Therefore, the data matrix $U(t_0; k)$ defined by Equation (2) becomes a five point

approximation:

$$U(t_0; k)|_{5-pnts} = \begin{pmatrix} u_1(t_0; k) & u_2(t_0; k) & u_3(t_0; k) & u_{n-1}(t_0; k) & u_n(t_0; k) \\ u_2(t_0; k) & u_3(t_0; k) & u_4(t_0; k) & u_n(t_0; k) & u_1(t_0; k) \\ u_3(t_0; k) & u_4(t_0; k) & u_5(t_0; k) & u_1(t_0; k) & u_2(t_0; k) \\ & & \vdots & & \\ u_{n-1}(t_0; k) & u_n(t_0; k) & u_1(t_0; k) & u_{n-3}(t_0; k) & u_{n-2}(t_0; k) \\ u_n(t_0; k) & u_1(t_0; k) & u_2(t_0; k) & u_{n-2}(t_0; k) & u_{n-1}(t_0; k) \end{pmatrix}.$$

Note that $U(t_0; k)|_{5-pnts}$ is of size $n \times 5$. The first two and last two rows assume that the data is periodic, since information crosses the boundary between indices n and 1. Next, the restriction of the data onto a subdomain is done by removing all rows that include points outside of the subdomain. For example, the restriction onto the subdomain indexed by $\{3, 4, 5, 6, 7\}$ yields:

$$U(t_0; k)|_{5-pnts,restricted} = \begin{pmatrix} u_3(t_0; k) & u_4(t_0; k) & u_5(t_0; k) & u_1(t_0; k) & u_2(t_0; k) \\ u_4(t_0; k) & u_5(t_0; k) & u_6(t_0; k) & u_2(t_0; k) & u_3(t_0; k) \\ u_5(t_0; k) & u_6(t_0; k) & u_7(t_0; k) & u_3(t_0; k) & u_4(t_0; k) \\ u_6(t_0; k) & u_7(t_0; k) & u_8(t_0; k) & u_4(t_0; k) & u_5(t_0; k) \\ u_7(t_0; k) & u_8(t_0; k) & u_9(t_0; k) & u_5(t_0; k) & u_6(t_0; k) \end{pmatrix}, \quad (8)$$

which reduces the matrix to size 5×5 – the loss of additional rows are required so that all cyclic permutations remain within the domain. It is important to note that the localized and restricted data matrix *no longer requires periodic data as long we are sufficiently away from the boundary*. The localized and restricted dictionary matrix is built by repeating the process in Equations (3)-(5), but using the localized and restricted data matrix described above (see Equation (8)).

Localizing the dictionary elements provide additional benefits. For many dynamical systems, information at a particular spatial point (or index) only interacts with information at its neighboring points (for example, all neighbors within a prescribed distance). Thus, localization may remove unnecessary complexities in the dictionary. The second is that the number of unknowns is severely reduced when considering a subset of the candidate functions. This was observed in [34] where localization reduced the inverse problem to a smaller (but still under-sampled) system and makes the sampling rate nearly independent of the ambient dimension n . Lastly, the accuracy of the approximation to the time derivative controls the error bound in our recovery problem. Thus, if the dynamics are only accurate in a small region, it is better to restrict the learning to that region. More data is usually beneficial; however, adding noisy and inaccurate measurements does not increase the likelihood of recovering the correct governing model.

2.6 Bounded Orthogonal Dictionary

The recovery of the coefficient vector c from data V is better conditioned with respect to a dictionary built from bounded orthogonal terms. For simplicity, we will detail this construction for data $u \in \mathbb{R}^n$, *i.e.* one spatial dimension with n -nodes. Consider a subset of the domain, $\mathcal{D} \subset \mathbb{R}^n$,

endowed with a probability measure μ . Suppose that $\{\phi_1, \phi_2, \dots, \phi_N\}$ is a (possibly complex-valued) orthonormal system on \mathcal{D} ,

$$\int_{\mathcal{D}} \phi_j(u) \overline{\phi_k(u)} d\mu(u) = \delta_{j,k} = \begin{cases} 0 & \text{if } j \neq k \\ 1 & \text{if } j = k \end{cases}.$$

The collection $\{\phi_1, \phi_2, \dots, \phi_N\}$ is called a *bounded orthonormal system* with constant $K_b \geq 1$ if:

$$\|\phi_j\|_{\infty} := \sup_{u \in \mathcal{D}} |\phi_j(u)| \leq K_b \quad \text{for all } j \in [N]. \quad (9)$$

Suppose that $u^{(1)}, u^{(2)}, \dots, u^{(m)} \in \mathcal{D}$ are sampling points which are drawn i.i.d. with respect to the orthogonalization measure μ , and consider the sampling matrix:

$$A_{\ell,k} = \phi_k(u^{(\ell)}), \quad \ell \in [m], \quad k \in [N].$$

An important example of a bounded orthonormal system is the Legendre polynomials. In high dimensional systems, we will use the tensorized Legendre basis in place of their corresponding monomials. We denote A_L the dictionary matrix corresponding to the tensorized Legendre basis. For example, if we consider the initial data samples $u(t_0)$ drawn i.i.d. from the uniform distribution $[-1, 1]^n$, then the Legendre polynomials (orthogonalization with respect to $d\mu = \frac{1}{2} dx$) up to degree three are:

$$1, \quad \sqrt{3}u_i, \quad \frac{\sqrt{5}}{2}(3u_i^2 - 1), \quad 3u_i u_j, \quad \frac{\sqrt{7}}{2}(5u_i^3 - 3u_i), \quad \frac{\sqrt{15}}{2}(3u_i^2 - 1)u_j, \quad \sqrt{27}u_i u_j u_k.$$

If a function is s -sparse with respect to the standard quadratic basis, it will be $(s+1)$ -sparse with respect to the Legendre basis, since the quadratic Legendre term, $\frac{\sqrt{5}}{2}(3u_i^2 - 1)$, can add at most a constant to the representation. If a function is s -sparse with respect to the standard cubic basis, it will be $(2s)$ -sparse with respect to the Legendre basis, since the terms $\frac{\sqrt{7}}{2}(5u_i^3 - 3u_i)$ and $\frac{\sqrt{15}}{2}(3u_i^2 - 1)u_j$ each add an additional s terms (in the worst-case scenario). We assume that s is sufficiently small so that, for example, a $(2s)$ -sparse system is still relatively sparse.

For the examples presented here, we focus on dynamical systems with (at most) cubic nonlinearity. The procedure above is not limited to this case. In fact, generalizing this construction to systems which are sparse with respect to alternative bounded orthogonal system is fairly direct. With high probability, a random matrix formed from bounded orthogonal terms will lead to a well-conditioned inverse problem $V = A_L c$ if c is sufficiently sparse (see [7, 11]).

3 Sparse Optimization and Recovery Guarantee

Let A_L be the dictionary in the Legendre basis up to third order. The size of A_L is $mnK \times N$, where N is the number of basis terms. The linear system $V = A_L c$ is underdetermined if we assume that m and K are small and fixed. To “invert” this system, we impose that the vector of coefficients c

is sparse, *i.e.*, c has only a few non-zero elements. This can be written formally as a non-convex optimization problem:

$$\min_c \|c\|_0 \quad \text{subject to } A_L c = V,$$

where $\|c\|_0 = \text{card}(\text{supp}(c))$ is the ℓ^0 penalty which measures the number of non-zero elements of c . In practice, the constraint is not exact since V is computed and contains some errors. The noise-robust problem is:

$$\min_c \|c\|_0 \quad \text{subject to } \|A_L c - V\|_2 \leq \sigma,$$

where $\sigma > 0$ is a noise parameter determined by the user or approximated from the data. The general noise-robust ℓ^0 problem is known to be NP hard [11], and is thus relaxed to the ℓ^1 regularized, noise-robust problem:

$$\min_{c'} \|c'\|_1 \quad \text{subject to } \|A_L c' - V\|_2 \leq \sigma, \quad (\text{L-BP}_\sigma)$$

which we refer to as the Legendre basis pursuit (L-BP) (for a general matrix A this is known as the ℓ^1 basis pursuit). Note that c' is the coefficients in the Legendre basis and c is the coefficients in the standard monomial basis. If the system is sufficiently sparse with respect to the standard monomial basis representation, then it will be sparse with respect to the Legendre basis representation, and thus the formulation is consistent. The parameter σ is independent of the basis we use in the dictionary matrix. In the ideal case, σ is the ℓ^2 error between the computed velocity and true velocity. In practice, it must be estimated from the trajectories.

3.1 Recovery Guarantee and Error Bounds

To guarantee the recovery of sparse solution to the underdetermined linear inverse problem, we use several results from random matrix theory. In general, it is difficult to recover $c \in \mathbb{R}^N$ from $Ac = V$, when $V \in \mathbb{R}^{\tilde{m}}$, $A \in \mathbb{R}^{\tilde{m} \times N}$, and $\tilde{m} \ll N$. In our setting, we know that the system is well-approximated by an s -term polynomial (for small s), and thus the size of the support set of c is relatively small. However, the locations of the nonzero elements (the indices of the support set) are unknown. If the matrix A is incoherent and $\tilde{m} \sim s \log(N)$, then the recovery of the sparse vector c from the ℓ^1 basis pursuit problem is possible. In particular, by leveraging the sparsity of the solution c and the structure of A , compressive sensing is able to overcome the curse of dimensionality by requiring far fewer samples than the ambient dimension of the problem. This approach also yields tractable methods for high-dimensional problems.

We provide a theoretical result on the exact and stable recovery of high-dimensional orthogonal polynomial systems with the cyclic condition via a probabilistic bound on the coherence of the dictionary matrix.

Theorem 3.1. *If A_{j_1} and A_{j_2} are two columns from the cyclic Legendre sampling matrix of order p generated by a vector $u \in \mathbb{R}^n$ with i.i.d. uniformly distributed entries in $[-1, 1]$ and $2p^2 \leq n$, then with probability exceeding $1 - \left(\frac{e}{p} + \frac{e}{2p^2}\right)^{2p} n^{-2p/11}$, the following holds:*

1. $|\langle A_{j_1}, A_{j_2} \rangle| \leq 12p^3 3^p \sqrt{n \log n}$ for all $j_1 \neq j_2 \in \{1, 2, \dots, n\}$,

2. $||A_{j_1}||^2 - n| \leq 12 p^3 3^p \sqrt{n \log n}$ for all $j_1 \in \{1, 2, \dots, n\}$.

Proof. Given a vector $u = (u_1, \dots, u_n) \in \mathbb{R}^n$ with i.i.d. uniformly distributed entries in $[-1, 1]$, let $A \in \mathbb{R}^{n \times N}$, with $N = \binom{n+p}{p}$, be the associated Legendre sampling matrix of order p , that is the matrix formed by transforming the matrix in Equation (5) with $k = 1$ to the Legendre system. In particular, the matrix is defined as:

$$A := \left[\begin{array}{c|c|c|c|c} U_L^0 & U_L^1 & U_L^2 & \cdots & U_L^p \end{array} \right],$$

where U_L^q is a matrix generated from the tensorized Legendre basis of order q for $0 \leq q \leq p$. For examples, $U_L^0 \in \mathbb{R}^n$ is a vector of all ones,

$$U_L^1 := \begin{pmatrix} \sqrt{3} u_1 & \sqrt{3} u_2 & \cdots & \sqrt{3} u_n \\ \sqrt{3} u_2 & \sqrt{3} u_3 & \cdots & \sqrt{3} u_1 \\ \vdots & \vdots & & \vdots \\ \sqrt{3} u_n & \sqrt{3} u_1 & \cdots & \sqrt{3} u_{n-1} \end{pmatrix},$$

and

$$U_L^2 := \begin{pmatrix} \frac{\sqrt{5}}{2} (3u_1^2 - 1) & 3u_1 u_2 & \cdots & \frac{\sqrt{5}}{2} (3u_n^2 - 1) \\ \frac{\sqrt{5}}{2} (3u_2^2 - 1) & 3u_2 u_3 & \cdots & \frac{\sqrt{5}}{2} (3u_1^2 - 1) \\ \vdots & \vdots & & \vdots \\ \frac{\sqrt{5}}{2} (3u_n^2 - 1) & 3u_n u_1 & \cdots & \frac{\sqrt{5}}{2} (3u_{n-1}^2 - 1) \end{pmatrix}.$$

Consider the random variable $Y_{j_1, j_2} = \langle A_{\cdot, j_1}, A_{\cdot, j_2} \rangle$ which is the inner product between the columns j_1 and j_2 of A , where $A = [A_{i, j}]$ for $1 \leq i \leq n$ and $1 \leq j \leq N$. Denote the components of the sum by:

$$Y_i := A_{i, j_1} A_{i, j_2},$$

so that we can write the inner product as:

$$Y_{j_1, j_2} = \langle A_{\cdot, j_1}, A_{\cdot, j_2} \rangle = \sum_{i=1}^n A_{i, j_1} A_{i, j_2} = \sum_{i=1}^n Y_i. \quad (10)$$

The components Y_i have several useful properties. The components of Y_{j_1, j_2} are uncorrelated, in particular, they satisfy $\mathbb{E}[Y_i] = 0$ if $j_1 \neq j_2$ and $\mathbb{E}[Y_i] = 1$ if $j_1 = j_2$, when one normalizes the columns. For fixed j_1 and j_2 , the elements $Y_i = A_{i, j_1} A_{i, j_2}$ follow the same distribution for all $1 \leq i \leq n$. This is a consequence of the cyclic structure of A , since each product $A_{i, j_1} A_{i, j_2}$ has the same functional form applied to different permutations of the data u .

Note that the $L^2(d\mu)$ -normalized Legendre system of order p (*i.e.* the tensor product of univariate Legendre polynomials up to order p) is a bounded orthonormal system with respect to $d\mu = \frac{1}{2} dx$. In particular, each basis term is bounded in $L^\infty([-1, 1]^n)$ by $K_b = 3^{p/2}$ (which can be achieved by $3^{p/2} u_{i_1} \cdots u_{i_p}$ at the boundary of the domain). Therefore, $|Y_i| \leq K_b^2 = 3^p$, $|Y_i - \mathbb{E}[Y_i]| = |Y_i| \leq 3^p$, and $\text{Var}(Y_i) \leq \mathbb{E}(Y_i^2) \leq 9^p$.

Applying Theorem 6.1 from the Appendix, which is a rephrasing of Theorem 2.5 from [14], with $\Delta = 4p^2$, $M = 3^p$, and $\text{Var}(Y_i) \leq 9^p$, yields the following bound:

$$P(|Y_{j_1, j_2} - \mathbb{E}Y_{j_1, j_2}| \geq \tau) \leq 2 \exp\left(-\frac{\tau^2(1 - p^2/n)}{8p^2(9^p n + 3^{p-1} \tau)}\right). \quad (11)$$

By assumption we have $\frac{p^2}{n} \leq \frac{1}{2}$, which happens, for example, when the maximal degree p is small and the ambient dimension n is much larger than p . By setting $\tau = 12p^3 3^p \sqrt{n \log n}$ and using $(1 - \frac{p^2}{n}) \geq \frac{1}{2}$ and $\log n \leq n$, we have:

$$\begin{aligned} P\left(|Y_{j_1, j_2} - \mathbb{E}Y_{j_1, j_2}| \geq 12p^3 3^p \sqrt{n \log n}\right) &\leq 2 \exp\left(-\frac{\tau^2}{16p^2(9^p n + 3^{p-1} \tau)}\right) \\ &\leq 2 \exp\left(-\frac{9p^4 n \log n}{n + 4p^3 \sqrt{n \log n}}\right) \\ &\leq 2 \exp\left(-\frac{9p^4 n \log n}{n + 4p^3 n}\right) \\ &\leq 2 \exp\left(-\frac{9p^4}{1 + 4p^3} \log n\right). \end{aligned} \quad (12)$$

Equation (12) holds for all pairs (j_1, j_2) , therefore taking a union bound over all $N(N-1)/2$ pairs and using the inequality:

$$N = \binom{n+p}{p} \leq \left(\frac{e(n+p)}{p}\right)^p \leq \left(n\left(\frac{e}{p} + \frac{e}{2p^2}\right)\right)^p = n^p \left(\frac{e}{p} + \frac{e}{2p^2}\right)^p,$$

for $p \geq 1$ where $e = \exp(1)$, we have:

$$\begin{aligned} P\left(\exists(j_1, j_2) : |Y_{j_1, j_2} - \mathbb{E}Y_{j_1, j_2}| \geq 12p^3 3^p \sqrt{n \log n}\right) &\leq N^2 \exp\left(-\frac{9p^4}{1 + 4p^3} \log n\right) \\ &\leq n^{2p} \left(\frac{e}{p} + \frac{e}{2p^2}\right)^{2p} \exp\left(-\frac{9p^4}{1 + 4p^3} \log n\right) \\ &\leq \left(\frac{e}{p} + \frac{e}{2p^2}\right)^{2p} \exp\left(\left(2p - \frac{9p^4}{1 + 4p^3}\right) \log n\right) \\ &\leq \left(\frac{e}{p} + \frac{e}{2p^2}\right)^{2p} \exp\left(-\frac{p^4 - 2p}{4p^3 + 1} \log n\right) \\ &\leq \left(\frac{e}{p} + \frac{e}{2p^2}\right)^{2p} n^{-2p/11}, \quad \text{for } p \geq 2. \end{aligned}$$

□

Theorem 3.1 provides an estimate on the coherence of the sampling matrix. We recall the coherence-based sparse recovery result from [12, 10, 11] below.

Theorem 3.2 (Related to Theorem 5.15 from [11]). *Let A be an $m \times N$ matrix with ℓ_2 -normalized columns. If:*

$$\max_{j \neq k} |\langle A_j, A_k \rangle| < \frac{1}{2s-1}, \quad (13)$$

then for any s -sparse vector $c \in \mathbb{C}^N$ satisfying $v = Ac + e$ with $\|e\|_2 \leq \sigma$, a minimizer $c^\#$ of $L\text{-BP}_\sigma$ approximates the vector c with the error bound:

$$\|c - c^\#\|_1 \leq d s \sigma,$$

where $d > 0$ is a universal constant.

Using Theorem 3.2, we can show the exact recovery for the case where A is a cyclic Legendre sampling matrix of order p .

Theorem 3.3. *Let $A \in \mathbb{R}^{n \times N}$ be the Legendre sampling matrix of order p generated by a vector $u \in \mathbb{R}^n$ with i.i.d. uniformly distributed entries in $[-1, 1]$, then with probability exceeding $1 - \left(\frac{\epsilon}{p} + \frac{\epsilon}{2p^2}\right)^{2p} n^{-2p/11}$, an s -sparse vector $c \in \mathbb{C}^N$ satisfying $v = Ac + e$ with $\|e\|_2 \leq \sigma$ can be recovered by $c^\#$, the solution of $L\text{-BP}_\sigma$, with the error bound:*

$$\|c - c^\#\|_1 \leq d s \sigma,$$

for some universal constant $d > 0$ as long as:

$$\frac{n}{\log n} \geq 144 p^6 9^p s^2.$$

In addition, if A is generated from K samples $u(k) \in \mathbb{R}^n$ with i.i.d. uniformly distributed entries in $[-1, 1]$ for $1 \leq k \leq K$ and:

$$K \geq \frac{144 p^6 9^p s^2 \log n}{n},$$

then with probability exceeding $1 - \left(\frac{\epsilon}{p} + \frac{\epsilon}{2p^2}\right)^{2p} n^{-2p/11}$, an s -sparse vector $c \in \mathbb{C}^N$ satisfying $v = Ac + e$ with $\|e\|_2 \leq \sigma$ can be recovered by $c^\#$, the solution of $L\text{-BP}_\sigma$, with the error bound:

$$\|c - c^\#\|_1 \leq d s \sigma.$$

Proof. The normalized matrix can be written as:

$$\bar{A} = AD,$$

where D is a diagonal matrix with $D_j = \|A_{\cdot,j}\|_2^2$, i.e. the diagonal contains the squared norm of the columns of A . Then \bar{A} has ℓ_2 -normalized columns, and Equation (13) is satisfied with probability exceeding $1 - \left(\frac{\epsilon}{p} + \frac{\epsilon}{2p^2}\right)^{2p} n^{-2p/11}$ as long as:

$$\frac{n}{\log n} \geq 144 p^6 9^p s^2.$$

Thus by Theorem 3.2, we have the corresponding ℓ^1 error bound. The extension of this result to multiple samples $u(k) = (u(k)_1, \dots, u(k)_n)$ for $1 \leq k \leq K$, follows directly as long as:

$$K \geq \frac{144 p^6 9^p s^2 \log n}{n}.$$

□

The results in Theorem 3.3 are important on their own. In particular, the theorem shows that for large enough dimension, one can recovery cyclic polynomial systems from only a few samples.

Returning to the problem of model selection for structured dynamical systems, we can apply Theorem 3.3 to obtain the following recovery guarantee.

Theorem 3.4. *Let $\{u(t_0; k), \dots, u(t_{m-1}; k)\}$ and $\{\dot{u}(t_0; k), \dots, \dot{u}(t_{m-1}; k)\}$ be the state-space and velocity measurements, respectively, for $1 \leq k \leq K$ bursts of the n -dimensional evolution equation $\dot{u} = f(u)$. Assume that the components, f_j , satisfy the cyclic structural condition and that they have at most s non-zero coefficients with respect to the Legendre basis. Assume that for each k , the initial data $u(t_0; k)$ is randomly sampled from the uniform distribution in $[-1, 1]^n$ (thus each component of the initial vector are i.i.d.). Also, assume that the total number of bursts, K , satisfies:*

$$K \geq \frac{144 p^6 9^p s^2 \log n}{n}. \quad (14)$$

Then with probability exceeding $1 - \left(\frac{e}{p} + \frac{e}{2p^2}\right)^{2p} n^{-2p/11}$, the vector c can be recovered exactly by the unique solution to Problem (L-BP $_{\sigma}$). In addition, under the same assumptions as above, if

the time derivative is approximated within η -accuracy in the scaled ℓ^2 norm, i.e. if $\tilde{u}(t_0; k)$ is the approximation to the time derivative and:

$$\sqrt{\frac{1}{K} \sum_{k=1}^K \left| \tilde{u}(t_0; k) - \dot{u}(t_0; k) \right|^2} \leq \eta,$$

then by setting $\sigma = \sqrt{K} \eta$ and using the submatrix of A_L consisting of only the initial data, any particular vector c is approximated by a minimizer $c^{\#}$ of Problem (L-BP $_{\sigma}$) with the following error bound:

$$\|c - c^{\#}\|_1 \leq d s \sigma, \quad (15)$$

where d is a universal constant.

Theorem 3.4 provides an ℓ^1 error bound between the learned coefficients and the true sparse

coefficients. If the nonzero elements of $c^\#$ are sufficiently large with respect to the error bound, then the support set containing the s -largest coefficients coincides with the true support set.

Proposition 3.5. *Assume that the conditions of Theorem 3.4 hold. Let S be the support set of the true coefficients c , i.e. $S := \text{supp}(c)$, and let $S^\#$ be the support set of the s -largest (in magnitude) of $c^\#$, a minimizer of Problem (L-BP $_\sigma$). If*

$$\sigma < \frac{\min_{j \in S} |c_j|}{2 d s}, \quad (16)$$

where d is the same universal constant as in Equation (15), then $S^\# = S$.

Proof. This proposition is a consequence of the recovery bound in Equation (15):

$$\|c^\# - c\|_1 \leq d s \sigma.$$

By assumption, σ satisfies Equation (16), then the maximum difference between the true and approximate coefficients is:

$$\max_j |c_j - c_j^\#| \leq \|c - c^\#\|_1 \leq d s \sigma < \frac{1}{2} \min_{j \in S} |c_j|.$$

Thus, for any $j \in S$, we have $|c_j^\#| > \frac{1}{2} \min_{j \in S} |c_j|$, and for any $j \in S^c$, we have $|c_j^\#| \leq \frac{1}{2} \min_{j \in S} |c_j|$. Therefore, $S^\#$ corresponds to the support set of $|c_j^\#| > \frac{1}{2} \min_{j \in S} |c_j|$, which is identically S . \square

Proposition 3.5 provides validation for post-processing the coefficients of Problem (L-BP $_\sigma$), in particular, if the noise is small enough, we could remove all but the s largest (in magnitude) coefficients in $c^\#$.

It is worth noting that it is possible to recover the system from one time-step. This is more probable as the dimension n of the problem grows. The sampling bound improves as n grows, since for large n , we have $n \gg s^2 \log n$. Thus, for large enough n , one random sample is sufficient. Furthermore, if $s^2 \ll n$, we can recover the system from only one time step and from only a subset $\tilde{n} < n$ of the coordinate equation, where $\tilde{n} \sim s^2$. Therefore, one just needs to have \tilde{n} accurate estimations of velocities.

Theorem 3.4 also highlights an important aspect of the scaling. Without any additional assumptions, one is limited to lower-order polynomials, since the numbers of samples required may be too large (since K_b grows rapidly). However, with additional assumptions, for example the cyclic structural condition, the recovery becomes nearly dimension-free, which as a side-effect, allows for higher-order polynomials more easily.

Note that if the initial data follows another random distribution, then one can construct the corresponding orthogonal polynomial basis. For example, we could assume that the initial data has i.i.d. components sampled from the Chebyshev measure on $[-1, 1]^n$ or an interpolating measure between the uniform measure and the Chebyshev measure [27].

4 Numerical Method

The constrained optimization problem (L-BP $_{\sigma}$) can be solved using the Douglas-Rachford algorithm [15, 8]. To do so, we first define the auxiliary variable w with the constraints:

$$(w, c) \in \mathcal{K} := \{(w, c) \mid w = Ac\} \quad \text{and} \quad w \in B_{\sigma}(V) := \{w \mid \|w - V\|_2 \leq \sigma\},$$

Equation (L-BP $_{\sigma}$) can be rewritten as an unconstrained minimization problem:

$$\min_{(w,c)} F_1(w, c) + F_2(w, c), \tag{17}$$

where the auxiliary functions F_1 and F_2 are defined as:

$$F_1(w, c) := \|c\|_1 + \mathbb{I}_{B_{\sigma}(V)}(w), \quad \text{and} \quad F_2(w, c) := \mathbb{I}_{\mathcal{K}}(w, c).$$

Here $\mathbb{I}_{\mathcal{S}}$ denotes the indicator function over a set \mathcal{S} , *i.e.*,

$$\mathbb{I}_{\mathcal{S}}(w) := \begin{cases} 0, & \text{if } w \in \mathcal{S}, \\ \infty, & \text{if } w \notin \mathcal{S}. \end{cases}$$

The utility of writing the optimization problem in this form is that both auxiliary functions have a simple and explicit proximal operators, which will be used in the iterative method. The proximal operator for a function $F(x)$ is defined as:

$$\text{prox}_{\gamma F}(x) := \underset{y}{\text{argmin}} \left\{ \frac{1}{2} \|x - y\|^2 + \gamma F(y) \right\},$$

where $\gamma > 0$ (to be specified later). The proximal operator of $F_1(w, c)$ is:

$$\begin{aligned} \text{prox}_{\gamma F_1}(w, c) &= \underset{(y,d)}{\text{argmin}} \left\{ \frac{1}{2} \|w - y\|^2 + \frac{1}{2} \|c - d\|^2 + \gamma \|d\|_1 + \gamma \mathbb{I}_{B_{\sigma}(V)}(w) \right\} \\ &= \left(\underset{y}{\text{argmin}} \left\{ \frac{1}{2} \|w - y\|^2 + \mathbb{I}_{B_{\sigma}(V)}(w) \right\}, \underset{d}{\text{argmin}} \left\{ \frac{1}{2} \|c - d\|^2 + \gamma \|d\|_1 \right\} \right) \\ &= \left(\text{proj}_{B_{\sigma}(V)}(w), S_{\gamma}(c) \right), \end{aligned}$$

where the projection onto the ball can be computed by:

$$\text{proj}_{B_{\sigma}(V)}(w) := \begin{cases} w, & \text{if } w \in B_{\sigma}(V), \\ V + \sigma \frac{w - V}{\|w - V\|_2}, & \text{if } w \notin B_{\sigma}(V). \end{cases}$$

and the soft-thresholding function S with parameter γ is defined (component-wise) as:

$$[S_\gamma(c)]_j = \begin{cases} c_j - \gamma \frac{c_j}{|c_j|}, & \text{if } |c_j| > \gamma, \\ 0, & \text{if } |c_j| \leq \gamma. \end{cases}$$

Similarly, the proximal operator for F_2 is:

$$\begin{aligned} \text{prox}_{\gamma F_2}(w, c) &= \underset{(y,d)}{\text{argmin}} \left\{ \frac{1}{2} \|w - y\|^2 + \frac{1}{2} \|c - d\|^2 + \mathbb{I}_{\mathcal{K}}(w, c) \right\} \\ &= (A(I + A^T A)^{-1}(c + A^T w), (I + A^T A)^{-1}(c + A^T w)). \end{aligned}$$

To implement the proximal operator for F_2 , we pre-compute the Cholesky factorization $(I + A^T A) = LL^T$ and use forward and back substitution to compute the inverse at each iteration. This lowers the computational cost of each of the iterations. The iteration step for the Douglas-Rachford method is:

$$\begin{aligned} (\tilde{w}^{k+1}, \tilde{c}^{k+1}) &= \left(1 - \frac{\mu}{2}\right) (\tilde{w}^k, \tilde{c}^k) + \frac{\mu}{2} \text{rprox}_{\gamma F_2} \left(\text{rprox}_{\gamma F_1} (\tilde{w}^k, \tilde{c}^k) \right), \\ (w^{k+1}, c^{k+1}) &= \text{prox}_{\gamma F_1}(\tilde{w}^{k+1}, \tilde{c}^{k+1}), \end{aligned} \tag{18}$$

where $\text{rprox}_{\gamma F_i}(x) := 2\text{prox}_{\gamma F_i}(x) - x$ for $i = 1, 2$. The second step of Equation (18) can be computed at the last iteration and does not need to be included within the main iterative loop. The approximation (w^k, c^k) converges to the minimizer of Problem (17) for any $\gamma > 0$ and $\mu \in [0, 2]$.

An outline of the numerical method is provided Algorithm 1. The data, $u(t; k) \in \mathbb{R}^n$, is given at two consecutive time-steps $t = t_0$ and $t = t_1$, and each component of $u(t_0; k)$ is i.i.d. uniform. The number of samples must satisfy Equation (14). First, the data must be arranged into the data matrix U using the cyclic permutation construction, as detailed in Sections 2.3, 2.4, and 2.5. Then, the data matrix is transformed so that each element is ranged in the interval $[-1, 1]$. Using the transformed data matrix, the Legendre dictionary matrix A_L is computed using the basis described in Section 2.6 and is normalized so that each column has unit ℓ^2 -norm. The coefficients with respect to the normalized Legendre dictionary is computed by solving Problem (L-BP $_\sigma$) via the Douglas-Rachford method. The last step is to map the coefficients with respect to the normalized Legendre dictionary to the standard monomial basis. As an optional step, the problem $Ac = V$ with respect to the monomial dictionary can be re-solved by restricting it to the support set computed from the main algorithm. In particular, let c be the output from Algorithm 1 and $S = \text{supp}(c)$, then the solution can be refined by solving the reduced system $A|_S \tilde{c} = V$ (see also Proposition 3.5).

5 Computational Results

The method and algorithm are validated on a high-dimensional ODE as well as two finite dimensional evolution equations that arise as the discretization of nonlinear PDEs with synthetic data. In each case, the initial data is perturbed by a small amount of uniform noise. For the 2D examples, it is assumed that there exists a block of size $n \times n$ of the data which is nearly uniformly distributed

Algorithm 1 Learning Sparse Dynamics

Data: Given: $u(t; k) \in \mathbb{R}^n$ for $t = t_0$ and $t = t_1$. The number of bursts k is either equal to 1 or very small. The number of nodes n does not need to be large.

Result: Coefficients of the governing equation $c \in \mathbb{R}^N$.

Step 1: Construct data matrix U as in Sections 2.3, 2.4, and 2.5.

Step 2 (optional): Add Gaussian noise to U , *i.e.* $U \mapsto U + \eta$, where $\eta \sim \mathcal{N}(0, \text{var})$.

Step 3: Construct the velocity vector V from using U from the previous step.

Step 4: Transform $U \mapsto aU + b$ so that each elements is valued in $[-1, 1]$.

Step 5: Construct the dictionary matrix A_L using U from Step 4; see Section 2.6.

Step 6: Normalize each column of A_L to have unit ℓ^2 -norm.

Step 7: Apply the Douglas-Rachford algorithm to solve Problem (L-BP $_{\sigma}$).

Input: Set $\sigma > 0$. Compute the Cholesky decomposition of $(I + A_L^T A_L)$. Initialize \tilde{w}^0 and \tilde{c}^0 .

while the sequence $\{\tilde{c}^k\}$ does not converge **do**

 | Update \tilde{w}^{k+1} and \tilde{c}^{k+1} based on Equation (18).

end

Output: $c_L := c^k$

Step 8: Map the coefficients c_L obtained from Step 5 to the coefficients with respect to the standard monomial basis on the original U as constructed in Step 1.

Step 9 (optional): The coefficients can be “debiased” by solving the system $A|_S \tilde{c} = V$, where $A|_S$ is the submatrix of A consisting of columns of A indexed by $S := \text{supp}(c)$ (from Step 6; see also Proposition 3.5).

in $[-1, 1]^{n \times n}$ (possibly up to translation and rescaling). Similarly for the high-dimensional ODE case, one can restrict to a subset of the components. Therefore, the input data to Problem (L-BP $_{\sigma}$) is restricted to the block (see Figure 1; the restriction is described in Section 2.5). It is important to note that the data restricted onto the blocks are not necessarily uniformly random; they may contain some slope. However, we assume that the dominate statistics are close to the uniform measure. In each of the examples, we apply the Douglas-Rachford algorithm described in Section 4, with the parameter $\sigma > 0$ determined beforehand.

5.1 The Lorenz 96 Equation

For the first example, we consider the Lorenz 96 equation:

$$\dot{u}_j = -u_{j-2} u_{j-1} + u_{j-1} u_{j+1} - u_j + F, \quad j = 1, 2, \dots, n,$$

for $j = 1, \dots, n$ with periodic conditions $u_{-1} = u_{n-1}$, $u_0 = u_n$, and $u_{n+1} = u_1$. We simulate the data using the forward Euler method with $n = 128$ and $F = 8$. The simulation is performed with a finer time-step $dt = 5 \times 10^{-5}$, but we only record the solution at the two time-stamps, the initial time $t_0 = 0$ and the final time $t_1 = 10^{-2}$. Let

$$u(t) = \left(u_1(t), u_2(t), \dots, u_n(t) \right)^T \in \mathbb{R}^n,$$

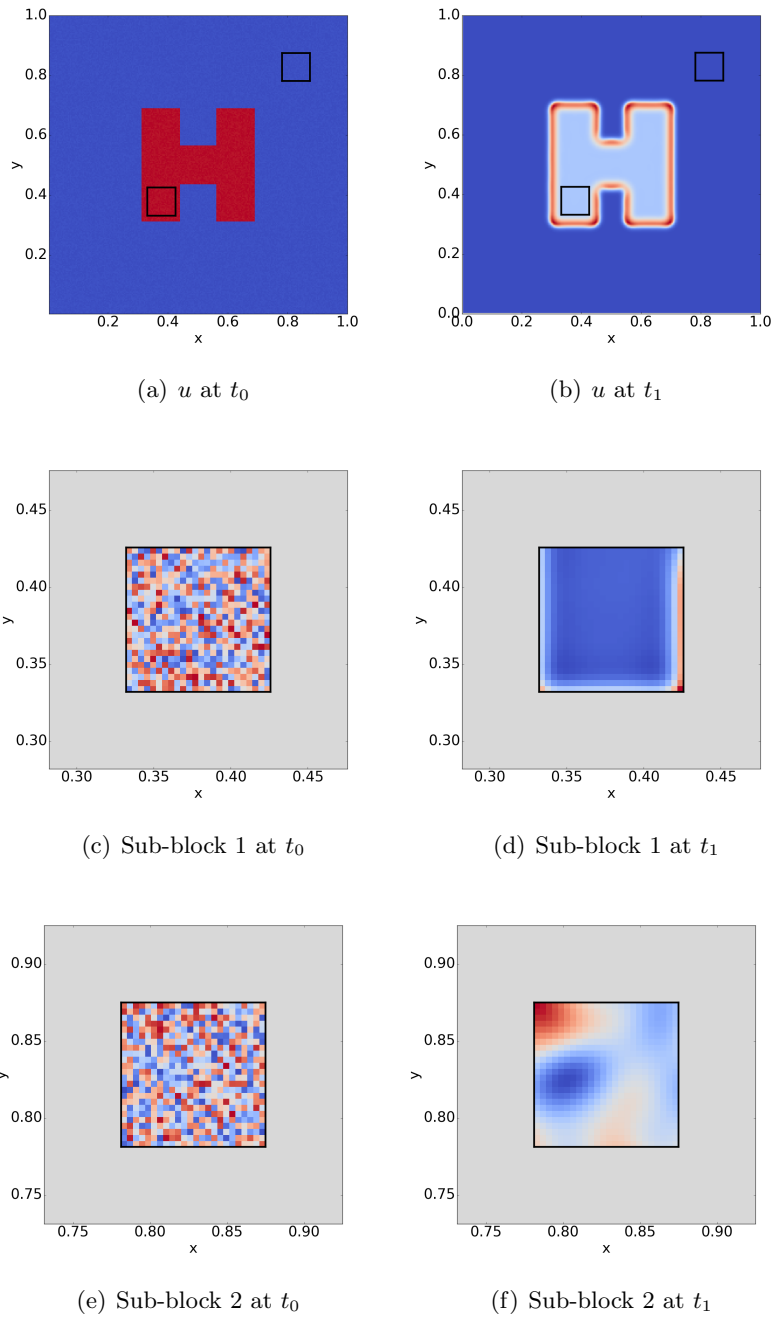


Figure 1: This figure includes a visual description of what the algorithm sees as its input. The first column corresponds to the system at t_0 and the second column corresponds to the system at t_1 . The first row is the full state, which is not known to the user; the two highlighted blocks are what is actually given. In the second row, the first block and its evolution are shown and in the third row the second block and its evolution is shown.

and set the initial data to be $u(0) = \nu$, where ν is sampled from the uniform distribution in $[-1, 1]^n$. Assume that the input data is corrupted by additive Gaussian noise, and the resulting measurements are denoted by \tilde{u} , i.e.,

$$\tilde{u} = u + \eta, \quad \eta \sim \mathcal{N}(0, \text{var}).$$

To construct the velocity vector V , we use the following approximation of \dot{u} :

$$\dot{u}_i(t_0) := \frac{\tilde{u}_i(t_1) - \tilde{u}_i(t_0)}{dt}, \quad i = 1, 2, \dots, n.$$

In this example, we vary the variance of the additive noise and the size of the dictionary, and compare the accuracy of the recovery under different noise levels and dictionary sizes. The results are provided in Section 5.4.

5.2 Viscous Burgers' Equation

Consider a 2D variant of the viscous Burgers' Equation:

$$u_t = \alpha \Delta u + u u_x + u u_y,$$

where Δ is the Laplacian operator and is defined by $\Delta u = u_{xx} + u_{yy}$, and $\alpha > 0$ is the viscosity. The equation is spatially invariant and well-posed, and thus there exists a discretization that yields a finite dimensional system that satisfies the cyclic structural condition. In particular, we simulate the data using the finite dimensional semi-discrete system:

$$\dot{u}_{i,j} = \alpha \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} + \frac{(u_{i+1,j})^2 - (u_{i-1,j})^2}{4h} + \frac{(u_{i,j+1})^2 - (u_{i,j-1})^2}{4h},$$

for $i, j = 1, 2, \dots, n$, where h is the grid spacing of the spacial domain, and $n = 1/h$. For α large enough (relative to h), this semi-discrete system is convergent. Note that this nonlinear evolution equation is 9-sparse with respect to the standard monomial basis in terms of $u_{i,j}$. We simulate the data using the discrete system above with a 128×128 grid, i.e. $h = 1/128$, and $\alpha = 10^{-2}$. This choice of α allows for both nonlinear and diffusive phenomena over the time-scale that we are sampling. The simulation is performed with a finer time-step $dt = 5 \times 10^{-8}$, but the solution is only recorded at the two time-stamps, the initial time $t_0 = 0$ and the final time $t_1 = 10^{-5}$.

The results appear in Figure 2. The initial data is plotted in Figures 2(a)-2(b), and is given by:

$$u_0(x, y) = 50 \sin(8\pi(x - 0.5)) \exp\left(-\frac{(x - 0.5)^2 + (y - 0.5)^2}{0.05}\right) + \nu,$$

where ν is sampled from the uniform distribution in $[-1, 1]^{128 \times 128}$. To construct the velocity vector

V , we use the following approximation of \dot{u} :

$$\dot{u}_{i,j}(t_0) := \frac{u_i(t_1) - u_i(t_0)}{dt}, \quad i, j = 1, 2, \dots, n. \quad (19)$$

The input to the algorithm is a block of size 7×7 . For display purpose, we mark in Figure 2(a) the location of the block which is used as the input. The learned equation is given by:

$$\begin{aligned} \dot{u}_{i,j} = & -655.9404u_{i,j} + 163.3892u_{i+1,j} + 163.5089u_{i-1,j} + 163.4859u_{i,j+1} + 163.5551u_{i,j-1} \\ & + 31.9211(u_{i+1,j})^2 - 31.7654(u_{i-1,j})^2 + 31.7716(u_{i,j+1})^2 - 31.8849(u_{i,j-1})^2, \end{aligned} \quad (20)$$

compared to the exact equation:

$$\begin{aligned} \dot{u}_{i,j} = & -655.36u_{i,j} + 163.84u_{i+1,j} + 163.84u_{i-1,j} + 163.84u_{i,j+1} + 163.84u_{i,j-1} \\ & + 32(u_{i+1,j})^2 - 32(u_{i-1,j})^2 + 32(u_{i,j+1})^2 - 32(u_{i,j-1})^2. \end{aligned} \quad (21)$$

The correct 9-terms are selected from the 351 possible candidate functions. To compare between the learned and true evolutions, we simulate the two systems up to the time of the shock formation, which is well beyond the interval of learning. Note that the qualitative difference between the two shocks is small.

5.3 Two Component Cubic Reaction-Diffusion Systems

Consider the 2D Gray-Scott Equation, which models a reaction-diffusion system:

$$\begin{aligned} u_t &= r_u \Delta u - uv^2 + f(1 - u), \\ v_t &= r_v \Delta v + uv^2 - (f + k)v, \end{aligned}$$

where r_u and r_v are the diffusion rates of u and v , respectively, f is the processing rate of u , and k represents the rate of conversion of v . We simulate the data using the finite dimensional semi-discrete system:

$$\begin{aligned} \dot{u}_{i,j} &= r_u \Delta_{h,9} u_{i,j} - u_{i,j} (v_{i,j})^2 + f(1 - u_{i,j}), \\ \dot{v}_{i,j} &= r_v \Delta_{h,9} v_{i,j} + u_{i,j} (v_{i,j})^2 - (f + k)v_{i,j}, \end{aligned}$$

for $i, j = 1, 2, \dots, n$, where h is the grid spacing of the spacial domain, $n = 1/h$, and $\Delta_{h,9}$ denotes the nine-point discrete Laplacian operator which is defined by:

$$\begin{aligned} \Delta_{h,9} u_{i,j} &= \frac{2}{3h^2} (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 5u_{i,j}) \\ &+ \frac{1}{6h^2} (u_{i+1,j+1} + u_{i-1,j+1} + u_{i-1,j+1} + u_{i-1,j-1}). \end{aligned}$$

Note that this nonlinear evolution equation is 12-sparse with respect to the standard monomial basis in terms of $u_{i,j}$ and is 11-sparse in terms of $v_{i,j}$.

We first present the implementation details for constructing Problem (L-BP $_{\sigma}$) in this setting (a system of PDEs). Given the initial data $u(t_0; k), v(t_0; k) \in \mathbb{R}^{n \times n}$, construct the data matrix

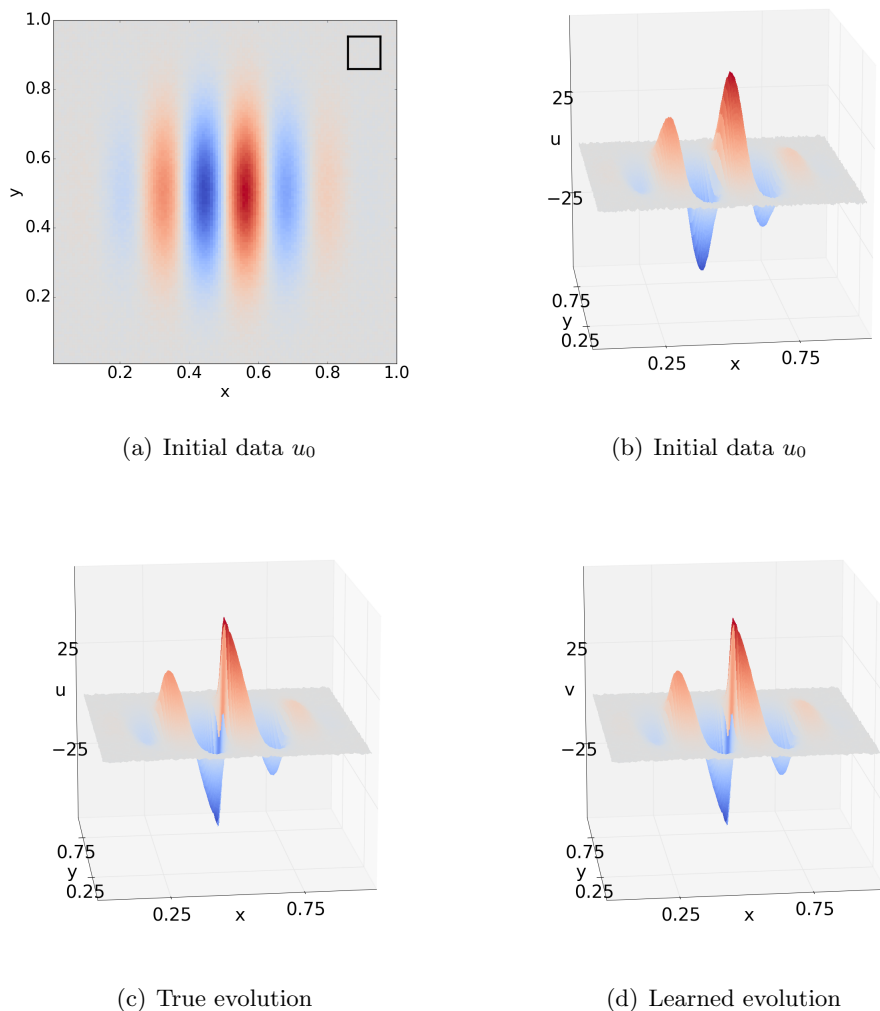


Figure 2: **The Burgers's Equation:** (a) The initial data u_0 in a planar view; the sub-block in the boxed region is used as the input to the algorithm. (b) The initial data u_0 in a 3D view. (c) The true evolution at $T = 10^{-3}$ using Equation (21). (d) The learned evolution at $T = 10^{-3}$ using Equation (20).

$W(t_0; k)$ as follows:

$$W(t_0; k) = \begin{pmatrix} u_{1,1}(t_0; k) & u_{1,2}(t_0; k) & \cdots & u_{n,n}(t_0; k) & v_{1,1}(t_0; k) & v_{1,2}(t_0; k) & \cdots & v_{n,n}(t_0; k) \\ u_{1,2}(t_0; k) & u_{1,3}(t_0; k) & \cdots & u_{n,1}(t_0; k) & v_{1,2}(t_0; k) & v_{1,3}(t_0; k) & \cdots & v_{n,1}(t_0; k) \\ u_{1,3}(t_0; k) & u_{1,4}(t_0; k) & \cdots & u_{n,2}(t_0; k) & v_{1,3}(t_0; k) & v_{1,4}(t_0; k) & \cdots & v_{n,2}(t_0; k) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{n,n}(t_0; k) & u_{n,1}(t_0; k) & \cdots & u_{n-1,n-1}(t_0; k) & v_{n,n}(t_0; k) & v_{n,1}(t_0; k) & \cdots & v_{n-1,n-1}(t_0; k) \end{pmatrix}.$$

Localization and restriction of $W(t_0; k)$ are performed with respect to both u and v independently. For example, with $n > 7$, the restriction onto the indices $(i, j) \in \{3, 4, 5\}^2$ is given by:

$$W(t_0; k)|_{9-pnts,restricted} = \left[U(t_0; k)|_{9-pnts,restricted} \quad | \quad V(t_0; k)|_{9-pnts,restricted} \right], \quad (22)$$

where $U(t_0; k)|_{9-pnts,restricted}$ is given by:

$$\begin{pmatrix} u_{3,3}(t_0; k) & u_{3,4}(t_0; k) & u_{3,2}(t_0; k) & u_{4,3}(t_0; k) & u_{4,4}(t_0; k) & u_{4,2}(t_0; k) & u_{2,3}(t_0; k) & u_{2,4}(t_0; k) & u_{2,2}(t_0; k) \\ u_{3,4}(t_0; k) & u_{3,2}(t_0; k) & u_{3,3}(t_0; k) & u_{4,4}(t_0; k) & u_{4,2}(t_0; k) & u_{4,3}(t_0; k) & u_{2,4}(t_0; k) & u_{2,2}(t_0; k) & u_{2,3}(t_0; k) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{5,5}(t_0; k) & u_{5,6}(t_0; k) & u_{5,4}(t_0; k) & u_{6,5}(t_0; k) & u_{6,6}(t_0; k) & u_{6,4}(t_0; k) & u_{4,5}(t_0; k) & u_{4,6}(t_0; k) & u_{4,4}(t_0; k) \end{pmatrix},$$

and $V(t_0; k)|_{9-pnts,restricted}$ is defined in the same way using the information of $v(t_0; k)$. Thus, we have reduced the size of the data matrix from $n \times (2n)$ to 9×18 . The localized and restricted dictionary matrix is then built by repeating the process in Equations (3)-(5), but using the localized and restricted data matrix described above (see Equation (22)). The velocity vectors, V_u for $\dot{u}_{i,j}$ and V_v for $\dot{v}_{i,j}$, are constructed as in Equation (6), and $\dot{u}_{i,j}$ and $\dot{v}_{i,j}$ are approximated using Equation (19). Let A_L be the (localized and restricted) dictionary in the Legendre basis. With the given system of PDEs, we then need to solve two basis pursuit problems:

$$\min_{c'_u} \|c'_u\|_1 \quad \text{subject to} \quad \|A_L c'_u - V_u\|_2 \leq \sigma,$$

and,

$$\min_{c'_v} \|c'_v\|_1 \quad \text{subject to} \quad \|A_L c'_v - V_v\|_2 \leq \sigma,$$

where c'_u and c'_v are the coefficients for the governing equations for $\dot{u}_{i,j}$ and $\dot{v}_{i,j}$, respectively, in the Legendre basis. Note that A_L is the same between each of the basis pursuit problems above since each equation depends on both u and v , but the outputs (c'_u and c'_v) are cyclic independently. Its worth noting that this example extends beyond the theoretical results, since the entire governing equation is not cyclic, but it is cyclic in the components (u, v) .

We simulate the data using the discrete system above with a 128×128 grid, *i.e.* $h = 1/128$, and parameters $r_u = 0.3$, $r_v = 0.15$. We consider three different parameter sets for the Gray-

Scott model, by varying the values of f and k . The simulation is performed with a finer time-step $dt = 10^{-6}$, but the solution is only recorded at two time-stamps, the initial time $t_0 = 0$ and the final time $t_1 = 10^{-5}$.

The initial data is shown in Figure 3, and is given by:

$$u_0(x, y) = 1 + 0.2\nu, \quad v_0(x, y) = \mathbb{I}_H(x, y) + 0.02\nu,$$

where ν is sampled from the uniform distribution in $[-1, 1]^{128 \times 128}$, and $H \subset [0, 1]^2$ represented the H-shaped region in Figure 3(d). The input to the algorithm is a block of u and the corresponding block of v , each of size 7×7 . For display purposes, we mark the block's location in each of Figures 3(a) and 3(d).

For the first example, we use the parameters $f = 0.055$ and $k = 0.063$, which creates a “coral”

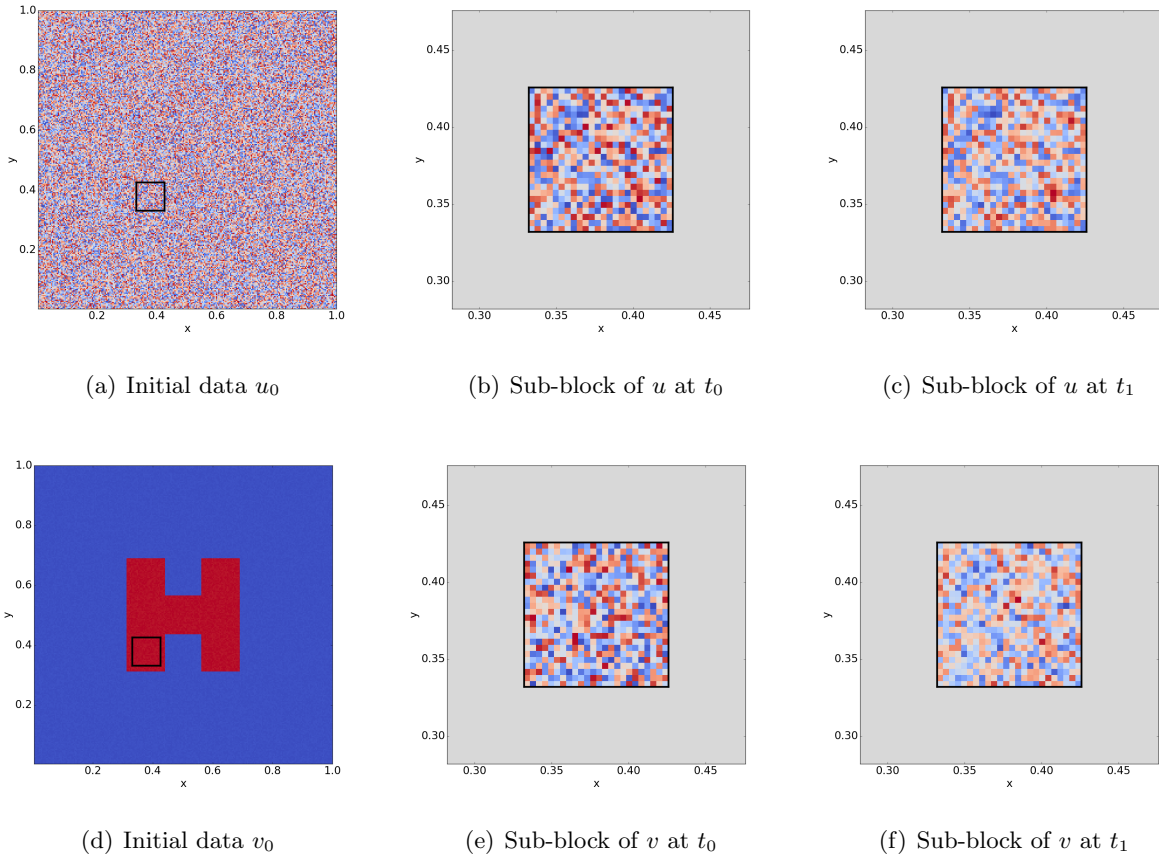


Figure 3: **Initial Data for the Gray-Scott Equation:** (a)(d) The initial data u_0 and v_0 ; the sub-blocks in the boxed regions are used as the input to the algorithm. (b)(c) The sub-block of u at time-stamps t_0 and t_1 , whose measurements are used to compute $\dot{u}_{i,j}$. (e)(f) The sub-block of v at time-stamps t_0 and t_1 , whose measurements are used to compute $\dot{v}_{i,j}$.

pattern. The visual results are given in Figure 4. The learned equations are:

$$\begin{aligned} u_t &= 0.30000\Delta u - 1.00000uv^2 - 1.05500u + 0.05501, \\ v_t &= 0.15000\Delta v + 1.00000uv^2 - 0.61801v - 0.00001, \end{aligned} \tag{23}$$

compared to the exact equations:

$$\begin{aligned} u_t &= 0.3\Delta u - uv^2 - 1.055u + 0.055, \\ v_t &= 0.15\Delta v + uv^2 - 0.618v. \end{aligned} \tag{24}$$

To compare between the learned and true evolutions, we simulate the two systems up to time-stamp $T = 5000$, well past the interval of learning. It is worth noting that two evolutions are close (see Section 5.4 for errors).

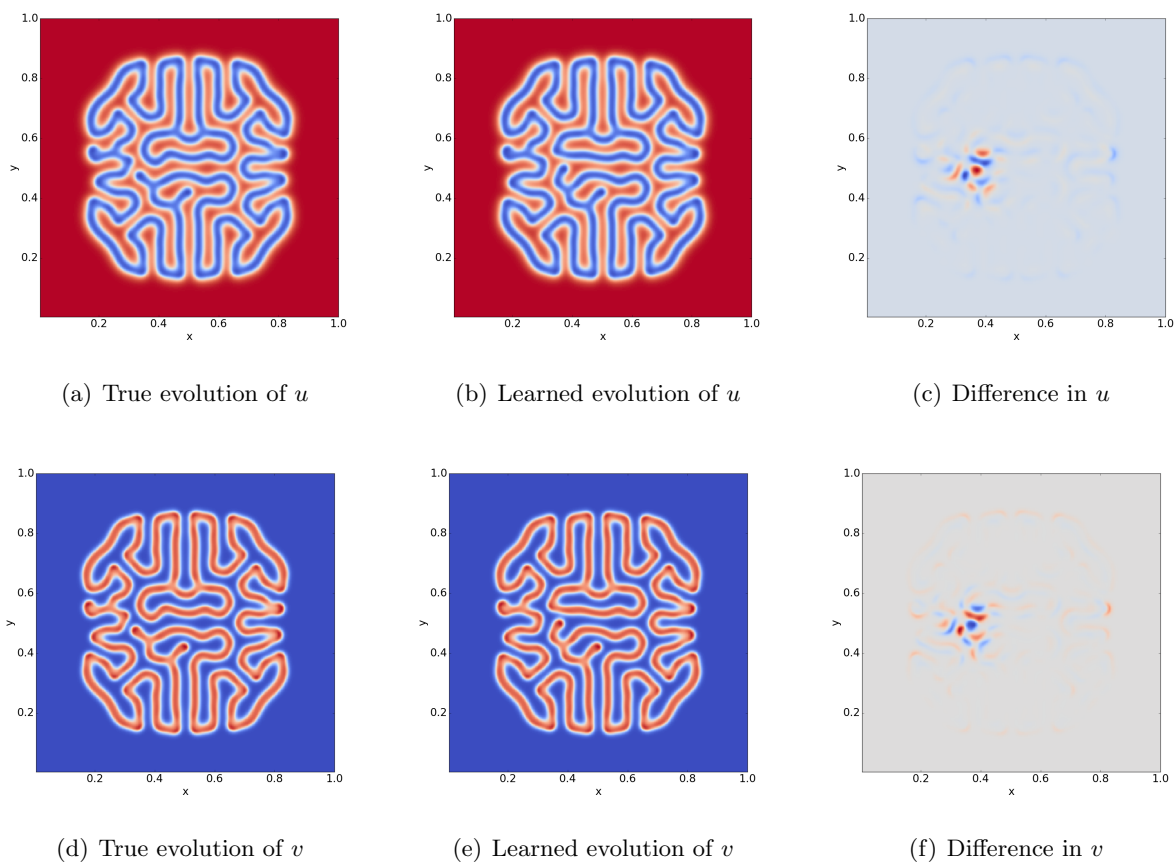


Figure 4: **The Gray-Scott Equation, Example 1:** (a)(d) The true evolution at $T = 5000$ using Equation (24). (b)(e) The learned evolution at $T = 5000$ using Equation (23). (c)(f) The difference between the true evolution and the learned evolution. Note that the patterns are very similar except at a small region near the center.

In the second example, we use the parameters $f = 0.026$ and $k = 0.053$, which yields a hexagonal pattern. The visual results are given in Figure 5. The learned equations is:

$$\begin{aligned} u_t &= 0.30000\Delta u - 1.00000uv^2 - 1.02600u + 0.02601, \\ v_t &= 0.15000\Delta v + 1.00001uv^2 - 0.57901v - 0.00001, \end{aligned} \quad (25)$$

compared to the exact equations:

$$\begin{aligned} u_t &= 0.3\Delta u - uv^2 - 1.026u + 0.026, \\ v_t &= 0.15\Delta v + uv^2 - 0.579v. \end{aligned} \quad (26)$$

As before, to compare between the learned and true evolutions, we simulate the two systems up to time-stamp $T = 2500$, beyond the learning interval. Small errors in the coefficient lead to some error in the pattern formulation; however, visually the simulations are similar.

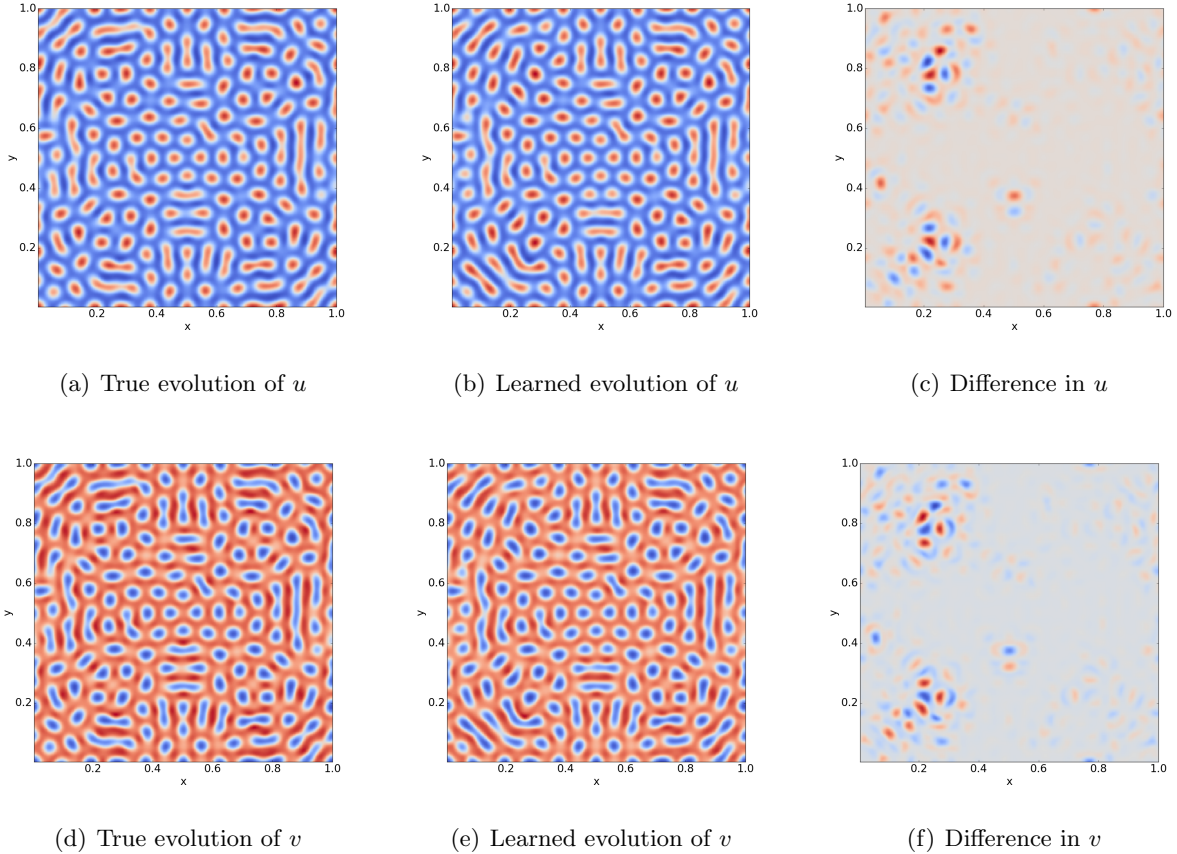


Figure 5: **The Gray-Scott Equation, Example 2** (a)(d) The true evolution at $T = 2500$ using Equation (26). (b)(e) The learned evolution at $T = 2500$ using Equation (25). (c)(f) The difference between the true evolution and the learned evolution. The overall qualitative structures are similar.

The last example uses the parameters $f = 0.018$ and $k = 0.051$, which leads to “U” shaped patterns. The visual results are given in Figure 6. The learned equations are:

$$\begin{aligned} u_t &= 0.30000\Delta u - 1.00000uv^2 - 1.01800u + 0.01801, \\ v_t &= 0.15000\Delta v + 1.00001uv^2 - 0.56902v, \end{aligned} \tag{27}$$

compared to the exact equations:

$$\begin{aligned} u_t &= 0.3\Delta u - uv^2 - 1.018u + 0.018, \\ v_t &= 0.15\Delta v + uv^2 - 0.569v. \end{aligned} \tag{28}$$

As before, we compare the learned and true evolutions, by simulating the two systems up to time-stamp $T = 1000$, well beyond the learning interval. The location of the “U” shaped regions are correct; however, there is some error in their magnitude.

5.4 Discussion

In all of the examples found in Sections 5.1-5.3, the linear systems are under-determined. Nevertheless, the model selected and parameters learned via Problem (L-BP $_{\sigma}$) yield relatively accurate results. The parameters used in the computational experiments in Sections 5.1-5.3 are summarized in Tables 1 and 2, and the corresponding errors are displayed in Tables 3 and 4.

In Tables 3 and 4(a), we measure the relative error in the learned model by comparing the coefficients:

$$E_{c, \text{LBP}} = \frac{\|c_{\text{exact}} - c'\|_{\ell^2}}{\|c_{\text{exact}}\|_{\ell^2}},$$

where c_{exact} is the exact coefficient vector corresponding to the underlying system and c' is the solution of Problem (L-BP $_{\sigma}$). The relative errors in the coefficients are within the theoretical bounds. Thus from limited measurements, we are able to extract the governing equations with high-accuracy.

In Table 4(b), we display the relative error between the learned solution and the exact solution:

$$E_u = \frac{\|u_{\text{exact}}(T) - u(T)\|_{\ell^2}}{\|u_{\text{exact}}(T)\|_{\ell^2}},$$

where $u_{\text{exact}}(T)$ is the true evolution at the final time T , and $u(T)$ is the evolution at the final time T with governing equation determined by c . The final time T is outside of the interval used to learn the coefficients. In both Burgers’ and Gray-Scott’s equation, the relative error is within expectation. Note that small errors in the coefficients accumulate rapidly in these evolution equations, since the locations of sharp transitions in the solution will be sensitive to the coefficients.

In Table 3, we display the relative errors $E_{c, \text{LBP}}$ for the Lorenz 96 Equation with different noise levels and dictionary sizes. In all the examples, a large noise level and a small sampling rate lead to a higher relative error. As the sampling rate increases and the noise level decreases as well as

the relative error. However, it is worth noting that the support sets are correctly identified.

Based on Theorem 3.4, for sufficient large block-sizes, it is possible to learn the correct coefficients with only one burst and one time-step. In Table 5, we display the relative errors $E_{c, \text{LBP}}$ for the Burgers' Equation and the Gray-Scott Equation using one burst and varying the block-sizes. The block-sizes are chosen so that the linear systems remain under-determined (see second columns in Table 5). In both examples, starting with a small block size leads to a high relative error, but as the block-size increases the relative error decreases.

For comparison, we calculate the least-square solution:

$$c_{\text{ls}} = \underset{c}{\operatorname{argmin}} \|Ac - V\|_2,$$

where A is the dictionary in the monomial basis and V is the velocity matrix. The relative error

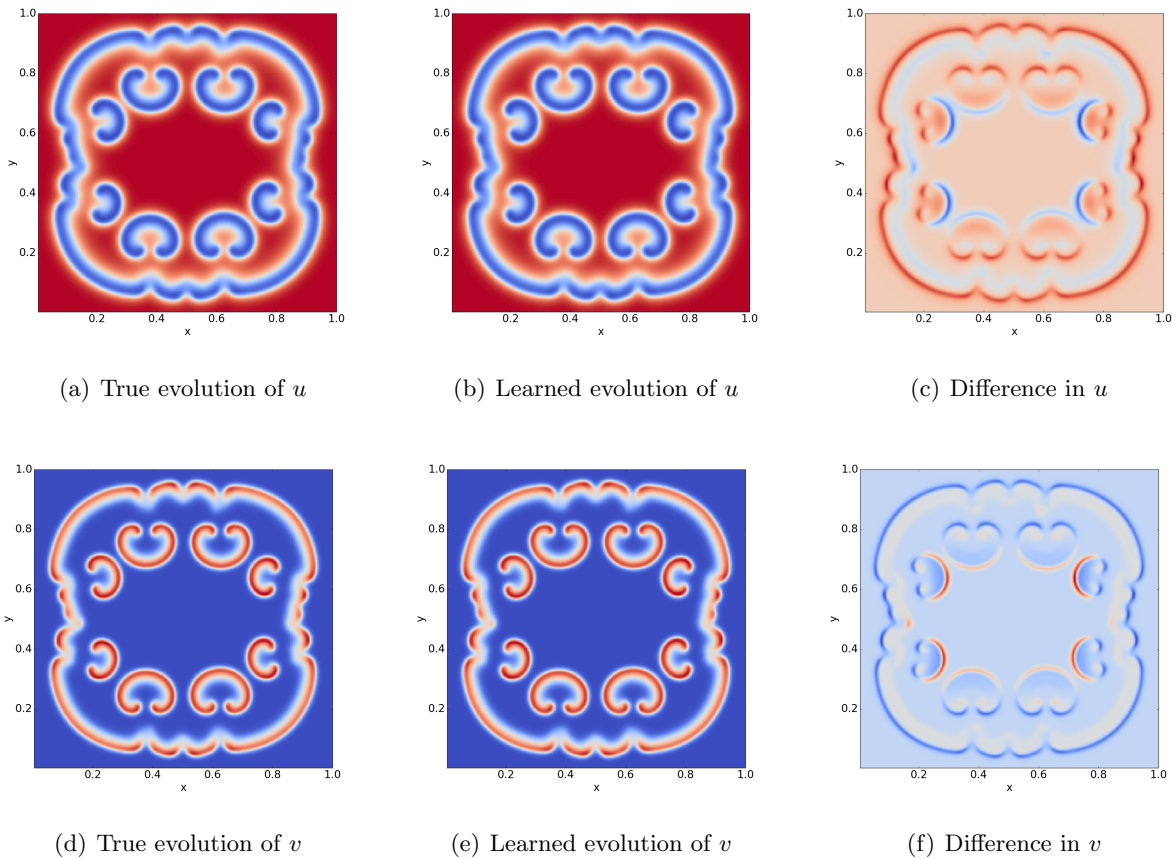


Figure 6: **The Gray-Scott Equation, Example 3:** (a)(d) The true evolution at $T = 1000$ using Equation (28). (b)(e) The learned evolution at $T = 1000$ using Equation (27). (c)(f) The difference between the true evolution and the learned evolution. The location of the regions are nearly identical. The errors are due to a difference in magnitude.

associated with the least-squares solution is denoted by $E_{c,LS}$. In Table 6, we display $E_{c,LS}$ for the Burgers' Equation and the Gray-Scott Equation corresponding to the same examples found in Table 5. The least-squares solution produces large errors since the resulting coefficient vector is dense (overfitted), leading to meaningless results.

Table 1: Parameters used in the computational qualitative experiments in Section 5.1.

(a) Parameters in matrix constructions.

	The Lorenz 96 Equation		
	Example 1	Example 2	Example 3
Block size	25	25	45
Number of bursts	2	4	4
Localization of the dictionary	10	10	10
Basis	3rd order Legendre	3rd order Legendre	3rd order Legendre
Size of the dictionary $n \times N$	50×2040	100×2024	204×2024

(b) Parameters in Problem (L-BP $_{\sigma}$).

		The Lorenz 96 Equation		
		Example 1	Example 2	Example 3
σ	var=0.2%	0.3515	0.53575	0.4075
	var=0.1%	0.3607	0.5074	0.7143
	var=0.05%	0.3380	0.5002	0.6888

Table 2: Parameters used in the computational qualitative experiments in Sections 5.2-5.3.

(a) Parameters in matrix constructions.

	The Burgers' Equation	The Gray-Scott Equation
Block size	7×7	7×7
Number of bursts	4	3
Localization of the dictionary	5×5	3×3
Basis	2nd order Legendre	3rd order Legendre
Size of the dictionary $n \times N$	196×351	147×1330

(b) Parameters in Problem (L-BP $_{\sigma}$). For the Gray-Scott Examples, the top value is the u -component and the bottom value is the v -component.

	The Burgers' Equation	The Gray-Scott Equation		
		Example 1	Example 2	Example 3
σ	26.3609	5.5707×10^{-5}	6.3055×10^{-5}	6.3321×10^{-5}
		5.2655×10^{-5}	6.0194×10^{-5}	6.0579×10^{-5}

Table 3: Errors associated with the computational experiments in Section 5.1.

		The Lorenz 96 Equation		
		Example 1	Example 2	Example 3
Size of the dictionary $n \times N$		50×2040	100×2024	204×2024
Sampling rate $n/N \times 100\%$		2.47%	4.94%	10.08%
$E_{c, \text{LBP}}$	var=0.2%	0.0276	0.0201	0.0185
	var=0.1%	0.0265	0.0197	0.0175
	var=0.05%	0.0254	0.0170	0.0165

Table 4: Errors associated with the computational experiments in Sections 5.2-5.3. For the Gray-Scott examples, the top value is the u -component and the bottom value is the v -component.

(a) Relative error, $E_{c, \text{LBP}}$

Burgers' Equation	Gray-Scott Equation		
	Example 1	Example 2	Example 3
0.0102	1.3775×10^{-5}	1.6550×10^{-5}	1.6382×10^{-5}
	1.6284×10^{-5}	1.5525×10^{-5}	1.5362×10^{-5}

(b) Relative error, E_u

Burgers' Equation	Gray-Scott Equation		
	Example 1	Example 2	Example 3
0.0038	0.0353	0.0856	0.0132
	0.1416	0.1221	0.0453

Table 5: Relative error $E_{c, \text{LBP}}$ with one burst and varying block-sizes.

(a) Burgers' Equation

Block size	Size of the dictionary	$E_{c, \text{LBP}}$
7×7	49×351	0.3285
9×9	81×351	0.0212
11×11	121×351	0.0187
15×15	225×351	0.0100

(b) The Gray-Scott Equation.

Block size	Size of the dictionary	$E_{c, \text{LBP}}$	
		u -component	v -component
15×15	225×1330	7.1278×10^{-1}	2.6170×10^{-1}
21×21	441×1330	2.0613×10^{-4}	2.4336×10^{-4}
27×27	729×1330	1.4427×10^{-5}	2.4541×10^{-5}

Table 6: Relative error $E_{c, \text{LS}}$ with one burst and varying block sizes.

(a) Burgers' Equation

Block size	Size of the dictionary	$E_{c, \text{LS}}$
7×7	49×351	1.3633
9×9	81×351	1.7804
11×11	121×351	2.8562

(b) The Gray-Scott Equation.

Block size	Size of the dictionary	$E_{c, \text{LS}}$	
		u -component	v -component
15×15	225×1330	5.8483	7.7621
21×21	441×1330	6.7622	7.5338
27×27	729×1330	5.9542	2.9482

6 Conclusion and Future Directions

In this work, we presented an approach for extracting the governing equation from under-sampled measurements when the system has structured dynamics. We showed that permuting i.i.d randomly sampled bursts and restructuring the associated dictionary yields an i.i.d. random sampling of a bounded orthogonal system, thus using a Bernstein-like inequality with a coherence condition, we show that the recovery is exact and stable. In addition, when the noise is sufficiently low, then the support of the coefficients can be recovered exactly, *i.e.* the terms in the governing equation can be exactly identified. The computational examples also highlight ways to extend the learning approach to larger systems and multi-component systems (where the cyclic structural condition must be carefully applied).

The structural assumption is valid for many dynamic processes, for example, when the data comes from a spatially-invariant dynamic system. In the algorithm and results, we made the assumption that one can sample a sub-block of the data to reasonable accuracy, in order to calculate derivatives accurately and so that the dictionary remains a bounded orthogonal system with respect to the given sampling measure. This is a weak assumption since the size of the sub-block is small. Thus, given noisy data, the sub-block could be the result of a pre-processing routine that has denoised and down-sampled the data (with the removal of outliers). The details on the pre-processing requirements is left for future investigations.

In future work, we would like to relax the requirements from cyclic structures to more general structures, while maintaining the very low-sampling requirement. An open question in the burst framework is how to quantify the trade-off between the burst size and the number of trajectories. This was not discussed in the numerical results but is of intrinsic importance to learning governing equations. In particular, if one has freedom to sample the initial condition but is limited by the number of samples along each trajectory or if one has freedom to sample along trajectories but is limited by the number of trajectories then it would be helpful to have theoretical estimates on the recovery rates. Lastly, the parameter $\sigma > 0$ used in the constraint must be estimated from the

data. It may be possible to learn σ for a given dataset.

Acknowledgments

H.S. and L.Z. acknowledge the support of AFOSR, FA9550-17-1-0125 and the support of NSF CAREER grant #1752116. G.T. acknowledges the support of NSERC grant #06135. R.W. acknowledges the support of NSF CAREER grant #1255631.

Appendix

Since the data is not independent, we cannot directly apply random matrix theory utilizing the standard Bernstein inequality. There are large deviations theorems for sums of “locally dependent” centered random variables. We recall and rephrase Theorem 2.5 from [14] for our purposes.

Theorem 6.1 (Theorem 2.5 from [14]). *Suppose that $Y = \sum_{i=1}^n Y_i$ and that all Y_i have the same distribution with $Y_i - \mathbb{E}Y_i \leq M$ for some $M > 0$. Suppose further that $S = n \text{Var}(Y_i)$, and $\Delta = \Delta_0 + 1$, where Δ_0 is the maximal degree of the dependency graph Γ for $\{Y_i\}$. Then*

$$P(Y - \mathbb{E}Y \geq \tau) \leq \exp\left(-\frac{\tau^2(1 - \Delta/(4n))}{2\Delta(S + M\tau/3)}\right). \quad (29)$$

Note that when $\Delta = 1$, this reduces essentially to the standard Bernstein inequality for i.i.d. random variables. For our problem, the dependency graph Γ has an edge between two vertices Y_i and Y_j if they have an index in common.

References

- [1] Ben Adcock, Simone Brugiapaglia, and Clayton G Webster. Polynomial approximation of high-dimensional functions via compressed sensing. *arXiv preprint arXiv:1703.06987*, 2017.
- [2] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.
- [3] Lorenzo Boninsegna, Feliks Nüske, and Cecilia Clementi. Sparse learning of stochastic dynamical equations. *The Journal of Chemical Physics*, 148(24):241723, 2018.
- [4] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.
- [5] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Sparse identification of nonlinear dynamics with control (SINDYc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

- [6] Russel E. Caflisch, Stanley J. Osher, Hayden Schaeffer, and Giang Tran. PDEs with compressed solutions. *Communications in Mathematical Sciences*, 13(58):2155–2176, 2015.
- [7] Emmanuel J. Candes and Yaniv Plan. A probabilistic and RIPless theory of compressed sensing. *IEEE Transactions on Information Theory*, 57(11):7235–7254, 2011.
- [8] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.
- [9] Magnus Dam, Morten Brøns, Jens Juul Rasmussen, Volker Naulin, and Jan S Hesthaven. Sparse identification of a predator-prey system from simulation data of a convection model. *Physics of Plasmas*, 24(2):022310, 2017.
- [10] David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.
- [11] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1(3). Birkhäuser Basel, 2013.
- [12] Rémi Gribonval and Morten Nielsen. Sparse representations in unions of bases. *IEEE transactions on Information theory*, 49(12):3320–3325, 2003.
- [13] Thomas Y. Hou, Qin Li, and Hayden Schaeffer. Sparse+ low-energy decomposition for viscous conservation laws. *Journal of Computational Physics*, 288:150–166, 2015.
- [14] Svante Janson. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24(3):234–248, 2004.
- [15] Pierre-Louis Lions and Bertrand Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.
- [16] Jean-Christophe Loiseau and Steven L Brunton. Constrained sparse Galerkin regression. *Journal of Fluid Mechanics*, 838:42–67, 2018.
- [17] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from data. *arXiv preprint arXiv:1710.09668*, 2017.
- [18] Edward N. Lorenz. Predictability: A problem partly solved. In *Proceedings of Seminar on Predictability*, volume 1(1), 1996.
- [19] Alan Mackey, Hayden Schaeffer, and Stanley Osher. On the compressive spectral method. *Multiscale Modeling & Simulation*, 12(4):1800–1827, 2014.
- [20] Niall M. Mangan, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2(1):52–63, 2016.

- [21] Niall M Mangan, J Nathan Kutz, Steven L Brunton, and Joshua L Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proc. R. Soc. A*, 473(2204):20170009, 2017.
- [22] Yannis Pantazis and Ioannis Tsamardinos. A unified approach for sparse dynamical system inference from temporal measurements. *arXiv preprint arXiv:1710.00718*, 2017.
- [23] Ji Peng, Jerrad Hampton, and Alireza Doostan. On polynomial chaos expansion via gradient-enhanced ℓ^1 -minimization. *Journal of Computational Physics*, 310:440–458, 2016.
- [24] Markus Quade, Markus Abel, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for rapid model recovery. *arXiv preprint arXiv:1803.00894*, 2018.
- [25] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.
- [26] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Numerical Gaussian processes for time-dependent and non-linear partial differential equations. *arXiv preprint arXiv:1703.10230*, 2017.
- [27] Holger Rauhut and Rachel Ward. Sparse Legendre expansions via ℓ_1 -minimization. *Journal of Approximation Theory*, 164(5):517–533, 2012.
- [28] Holger Rauhut and Rachel Ward. Interpolation via weighted ℓ_1 minimization. *Applied and Computational Harmonic Analysis*, 40(2):321–351, 2016.
- [29] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3:e1602614, 2017.
- [30] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. In *Proceedings of the Royal Society of London A*, volume 473(2197), page 20160446. The Royal Society, 2017.
- [31] Hayden Schaeffer, Russel Caflisch, Cory D Hauck, and Stanley Osher. Sparse dynamics for partial differential equations. *Proceedings of the National Academy of Sciences*, 110(17):6634–6639, 2013.
- [32] Hayden Schaeffer and Thomas Y. Hou. An accelerated method for nonlinear elliptic PDE. *Journal of Scientific Computing*, 69(2):556–580, 2016.
- [33] Hayden Schaeffer and Scott G. McCalla. Sparse model selection via integral terms. *Physical Review E*, 96(2):023302, 2017.
- [34] Hayden Schaeffer, Giang Tran, and Rachel Ward. Extracting sparse high-dimensional dynamics from limited data. *arXiv preprint arXiv:1707.08528*, 2017.

- [35] Hayden Schaeffer, Giang Tran, and Rachel Ward. Learning dynamical systems and bifurcation via group sparsity. *arXiv preprint arXiv:1709.01558*, 2017.
- [36] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.
- [37] Mariia Sorokina, Stylianos Sygletos, and Sergei Turitsyn. Sparse identification for nonlinear optical communication systems: Sino method. *Optics Express*, 24(26):30433–30443, 2016.
- [38] Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
- [39] Giang Tran, Hayden Schaeffer, William M. Feldman, and Stanley J. Osher. An l^1 penalty method for general obstacle problems. *SIAM Journal on Applied Mathematics*, 75(4):1424–1444, 2015.
- [40] Giang Tran and Rachel Ward. Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling & Simulation*, 15(3):1108–1129, 2017.