Sparse Identification of Nonlinear Dynamics

Brian Wodetzki

***Abstract*—Non-linear system identification has great potential to solve a variety of problems ranging from identifying chaotic dynamics to charecterizing fluid flows and more. SINDy is a popular algorithm that has proven its ability to identify a multitude of systems. This paper reviews some of the theory behind the optimization method presented in the original SINDy paper and compares it with another promising optimization method called SR3. SR3 proved to be less robust to noise and more susceptible to providing non-optimal solutions than STLS in all cases explored in this paper.**

I. Introduction

System identification has been a popular area for research for decades, beyond papers it is commonly used in a variety of fields from aerospace engineering to civil engineering. One of the first techniques that laid the foundation for system identification was proposed by Moore in 1981 [1] and was originally a solution for model reduction. This technique, known as Principal Component Analysis (PCA) was an improvement on previous methods, developed by individuals such as Kalman, that sought to describe the dynamics of a linear system by characterizing its most dominant modes. The technique developed by Moore required the direct knowledge of a systems dynamics to characterize it by its most dominant modes alone. Building on a number of improvements on Moore's original algorithm, Rowley showed in 2005 [2] that Balanced Proper Orthogonal Decomposition (BPOD) (a method first developed by Willcox and Peraire in 2002 [3]) one can use experimental data to accurately reduce the order of the system by identifying the dominant modes.

Prior to Rowley's paper in 2002, however, there was need to identify linear system modes from data and without a prior knowledge of system dynamics. In an effort to characterize the modes seen in the Hubble space telescope, NASA engineers Juang and Papa developed an algorithm known as the Eigensystem Realization Algorithm (ERA) in 1985 [4]. The method developed is similar in structure to BPOD, however, a prior idea of the model is not required to identify the dominant modes of the system making it one of the first true model identification techniques.

The aforementioned methods, however, only relate to intrinsically linear system. This assumption greatly limits their application as most systems are non-linear in nature. Dynamic Mode Decomposition (DMD) is an algorithm that can identify, and perform model reduction, on a non-linear systems. DMD was first proposed by Schmid in 2009 [5] for fluids problems, but has since found uses in a wide variety of system identification tasks. DMD will find a best fit linear operator to predict the next time step of measurements from a system, whether the system is non-linear or linear. The idea of a linear operator to approximate a non-linear system was first theorized by Koopman in his 1931 paper [6]. Here it is shown that for any non-linear function, one can approximate it using an infinite dimensional linear operator, termed the Koopman operator. In fact, finding the Koopman operator for a given system is still a very active area of research and DMD is regarded as one solution to find the Koopman operator.

Despite the effectiveness of DMD and the promise of the Koopman operator, in the majority of cases both these methods still fail to provide exact solutions to the non-linear dynamics of a system, and only approximate it. Work has been done, however, to characterize the true non-linear dynamics of a system symbolically without the need for a linear approximation. Although work on symbolic non-linear system identification has been done previously work done by Bongard and Lipson [7] and a similar paper written by Schmidt and Lipson [8] proved to be seminal works in this field. The algorithm was based on determining the true symbolic differential equation of a system using genetic programming. This method, and others in the field, had the issue of large equations with redundant terms. Brunton, Proctor, and Kutz saw this as an opportunity and developed an algorithm called Sparse Identification of Nonlinear Dynamical systems (SINDy) [9]. This algorithm uses regression to determine important terms in a dynamical system, then uses optimization to trim terms that do not contribute much to the dynamics. This optimization is repeated until a criteria is met and it is said that the model has reached a best fit. This trimming of redundant terms is why the algorithm is "Sparse". This sparsity or parsimony is the main extension on previous work and it has been shown to work for a number of real world systems. Although there is a number of works extending SINDy, this project aims to replicate the original SINDy algorithm aimed at sparse non-linear system identification.

II. Problem Formulation

Given a non-linear dynamical system, the goal is too identify the full system dynamics given only data from the system. The dynamical system is given below.

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

In practice, however, data is rarely clean. It is important that any algorithm have the ability to identify dynamics from possibly incomplete or noisy data.

This paper will aim at identifying the underlying dynamics for 3 systems. First, a chaotic system, the Lorenz System. Full state observability will be assumed however there are methods of recovering a projection of the dynamics through partial observations [10]. The dynamics of the system are given below.

Purdue University

$$\begin{aligned}\dot{x} &= \sigma(y-x)\\ \dot{y} &= x(\rho-z)-y\\ \dot{z} &= xy-\beta z\end{aligned} \tag{2}$$

The second system that will be identified reflects the algorithms performance in identifying fluid flows that evolve as a non-linear PDE. A cylinder shedding vortices at a low reynolds number will be used to test this. This data is collected using the 2-D navier-stokes equations integrated through time to produce information on the fluid flow. In this case only the vorticity will be analyzed for simplicity. An example of the flow is presented in figure 1.

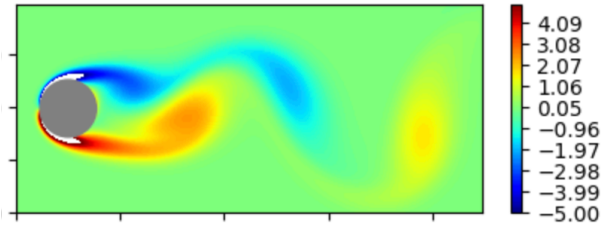


Fig. 1: Heat Map of the Vorticity of a Flow Past A Cylinder

The goal of analyzing this problem will be to reproduce a well known low dimensional approximation of this flow is called the mean field model presented in [11]. This model is purely quadratic but it evolves in a way that resembles cubic non-linearities.

$$\begin{aligned}\dot{x} &= \mu x-\omega y+Axz\\ \dot{y} &= \omega x+\mu y+Ayz\\ \dot{z} &= -\lambda(z-x^2-y^2)\end{aligned} \tag{3}$$

Finally, the third system that will test our methods will be a system that produces complex dynamics through bifurcations. A popular system for simulating dynamics with bifurcations is the logistic map given below.

$$x_{k+1} = \mu x_k(1-x_k) \tag{4}$$

The goal is to develop an algorithm that can accurately identify the dynamics of a wide range of systems give only imperfect data. These three test cases aim to show the effectiveness of the algorithm on multiple different types of systems.

III. Methods

The main method presented in the original SINDy paper [9], and the method that will be used here relies on the concept of regularized regression, that is solving the problem below.

$$\min_x \frac{1}{2}\|Ax-b\|^2 + R(x,\lambda) \tag{5}$$

In general, the goal is to find linear coefficients $x = \Xi$ that optimally project onto the function space encoded in the feature matrix $A = \Theta(\boldsymbol{X})$ to best fit the data $b = \dot{\boldsymbol{X}}$. The function $R(\Xi)$ represents a regularization term that in general will help keep the solution from being overfit and its parameter λ determines how much the regularization effects the solution. The problem in question asks how to find a non-linear function $f(\boldsymbol{X})$ that will fit $\dot{\boldsymbol{X}}$ the best way possible.

$$\min_x \frac{1}{2}\|\Theta(\boldsymbol{X})\Xi - \dot{\boldsymbol{X}}\|^2 + R(\Xi,\lambda) \tag{6}$$

The approach that is taken in the SINDy paper treats $\dot{\boldsymbol{X}}$ as the data and fits a large library of possible functions to the data, this library is encoded with $\Theta(\boldsymbol{X})$ and is dependant on $\boldsymbol{X}$. $\boldsymbol{X}$ is a matrix made of of the trajectories for every state and $\dot{\boldsymbol{X}}$ is a matrix made of the corresponding time derivatives for each state at each point in the trajectory, both are given below.

$$\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}^T(t_1)\\ \vdots\\ \boldsymbol{x}^T(t_m)\end{bmatrix} = \begin{bmatrix}x_1(t_1) & \cdots & x_n(t_1)\\ \vdots & \ddots & \vdots\\ x_1(t_m) & \cdots & x_n(t_m)\end{bmatrix} \tag{7}$$

$$\dot{\boldsymbol{X}} = \begin{bmatrix}\dot{\boldsymbol{x}}^T(t_1)\\ \vdots\\ \dot{\boldsymbol{x}}^T(t_m)\end{bmatrix} = \begin{bmatrix}\dot{x_1}(t_1) & \cdots & \dot{x_n}(t_1)\\ \vdots & \ddots & \vdots\\ \dot{x_1}(t_m) & \cdots & \dot{x_n}(t_m)\end{bmatrix} \tag{8}$$

Above, m represents the number of timesteps for every trajectory and n represents the state dimension of the non-linear equations. Note that this formulation may work with multiple stacked trajectories. Now, the feature matrix $\Phi(\boldsymbol{X})$ will contain functions that the user believes may play a role in the dynamics. Often times this takes the form of multivariate polynomials up to a certain degree, sinusoidal functions, exponentials, any combinations of the latter, and others. The presence of a large number of functions in particular, however, creates its own problems as the true dynamics may be approximated by a combination of multiple functions as in the case of sinusoids (fourier series), polynomials (taylor series), and other function approximations.

This issue above can be mitigated to an extent, however, through clever choice of $R(\Xi)$. In typical regression one of the most popular choices of $R(x)$ is the l_2 norm, $\|\cdot\|_2$. This method of regression is called ridge regression and although it does well at preventing overfitting, it does not fix the issue of misidentifying dynamics by using an excessive amount of functions. This comes from understanding the shape of the level sets of the different norms. Figure 2 shows the level sets of different norms, see that each axis corresponds to the coefficient of every function present in the solution (these images are 2D, so it is useful to picture two functions) one can ascertain the difference between the norms. The l_2 norm, which has level sets the shape of a circle, will prioritize the linear combination of functions that best fit the data. The l_1 norm, however, has level sets that are extended in the direction of each axis and therefore will prioritize solutions that contain a smaller number of function components in the solution. With this logic, the l_0 norm is seen to be an extreme version of the l_1 norm, again finding a solution that minimizes the total number of components in the solution. The l_∞ norm on the other hand prioritizes a mixture of components.

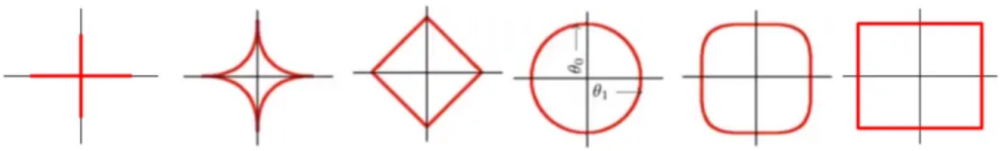


Fig. 2: Visual Representation of Common Norms (left to right): l_0, $l_{0.5}$, l_1, l_2, l_4, l_∞

The norms used in the regularization function for the problem at hand should encourage a small number of total terms terms since many dynamical systems can be explained by equations that have a small number of terms. Equivalently our solution should be sparse or parsimonious. For these reasons l_1 through l_0 norms are desirable. Indeed, l_1 norms are regularly used in regression to encourage parsimonious solutions, so much so that optimization using this regularization has earned its own name, Least Absolute Shrinkage and Selection Operator (LASSO). The name comes from one method of performing this regularized regression called proximal gradient descent which solves eq. 5. This method, presented in algorithm 1, is a two step process and starts by taking a gradient step towards the solution of the linear equation being solved then corrects the solution based on a proximal operator that is dependant on the type of regularization being used.

Algorithm 1 Proximal Gradient Descent

$x^k \leftarrow$ initGuess()
while $i < i_{max}$ **do**
  $z^k \leftarrow x^k - \alpha \nabla_x \|Ax - b\|_2^2$
  $x^{k+1} \leftarrow$ proximalOperator(z^k)
  if $x^{k+1} - x^k > eps$ **then** ▷ x not converged
    $x^{k+1} \leftarrow x^k$
  else if N is odd **then**
    break
  end if
end while

The proximal operator for l_1, shown in eq. 9 and denoted by P, performs an operation that shrinks all values of the coefficients, and sets the smallest coefficient values to 0, this is the selection part of the process.

$$P(x,\lambda) = \begin{cases} 0 & if \quad -\lambda \leq x \leq \lambda \\ x - \lambda & if \quad x > \lambda \\ x + \lambda & if \quad x < -\lambda \end{cases} \tag{9}$$

The method of solving l_1 regularized least squares, however, has become overshadowed by other methods based off of convex optimization as the l_1 regularized regression problem is convex. Irregardless, the structure of proximal optimization is versatile also lends itself to the use of different proximal operators to perform regularized regression, this versatility has led to its general structure being used in many regularized regression problems. One use of the proximal gradient descent is solving the l_0 regularization, presented in eq. 10, which performs thresholding operations on the coefficients. This proximal operator seeks to minimize what is often referred to as the "l_0" norm, however, mathematically this fails to meet the criteria of a norm.

$$P(x,\lambda) = \begin{cases} 0 & if \quad |x| \leq \lambda \\ x & if \quad x > \lambda \end{cases} \tag{10}$$

The operator above has recently gained prominence since Blumenthal [12] showed the effectiveness of using this operator in solving regularization problems, a technique referred to as hard thresholding. This operator, however, poses a non-convex optimization problem which prevents gradient descent algorithms that use this regularization from guaranteeing an optimal solution. It is worth noting that all operators $l_{[0,1)}$ produce non-convex problems, while operators $l_{[1,\infty]}$ produce convex problems. This can also be seen by investigating the level sets of the norms.

A. Sequential Thresholded Least Squares

Recall that the original problem requires a sparse solution of basis function that characterize some dynamics. Therefore some norm in the range of $l_[0,1]$ is desirable because of sparsity promoting features. The use of the l_1 norm to promote sparsity was attempted on system identification problems, however, despite its convexity in many cases it failed to identify the correct dynamics.

In the orginal SINDy paper, a custom algorithm called Sequential Thresholded Least Squares (STLS) was used. This algorithm, presented in algorithm 2, is similar to proximal gradient descent in that the core of the algorithm is a two step process of solving a least squares problem and correcting with a proximal operator.

Algorithm 2 Sequential Thresholded Least Squares

$x^k \leftarrow$ leastSquares(A, b)
while $i < i_{max}$ **do**
  $z^k \leftarrow$ proximalOperator(x^k)
  $idxs \leftarrow$ getNonZeroIdxs(z^k)
  $x^{k+1} \leftarrow$ leastSquares($A[:, idxs], b$)
  if $x^{k+1} - x^k > eps$ **then** ▷ x not converged
    $x^{k+1} \leftarrow x^k$
  else if N is odd **then**
    break
  end if
end while

This algorithm has a few key differences than vanilla proximal gradient descent using the l_0 proximal operator. First, proximal gradient descent uses a gradient descent step, this algorithm solves the problem at hand with least squares. This

is enabled by the second difference. STLS applies the proximal operator first and performs optimization on only the components of the function that are determined by the proximal operator to be relevant, i.e. only the components that are non-zero. These two differences are the primary driving factors that allow this algorithm to outperform proximal gradient descent.

B. SR3

Beyond these two aforementioned methods, there exists a multitude of ways to solve regularized least squares problems that encourage sparsity. One method called Sparse Relaxed Regularized Regression (SR3) presented by Zheng et al. [13] aims to solve the problem by introducing a relaxed coordinate system w, a replacement for the parameters x, to both encourage the regularization while making the problem easier to solve.

$$\min_{x,w} \frac{1}{2}\|Ax-b\|_2^2 + R(w,\lambda) + \frac{\kappa}{2}\|Cx-w\|_2^2 \qquad (11)$$

This can be thought of as pulling the coefficients x closer to a similar vector w by a weight κ where the vector w is effected by the regularization term. This has the effect of posing the problem on the new relaxed coordinates w which results in the solution being reached faster. This is shown in figure 3 where the l_1 problem is being solved in relaxed coordinates.

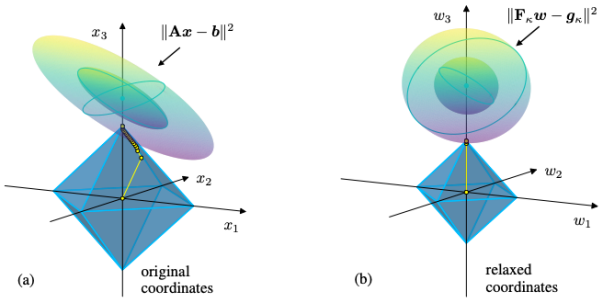


Fig. 3: Effect of Relaxed Coordinates w On An l_1 Regularized Least Squares Problem

Although in the original SR3 paper a proximal gradient descent algorithm is proposed, a slight deviation is made in the true implementation [14] where similar to the STLS algorithm, a least squares problem is solved at each iteration instead of a gradient step towards the solution. This is presented in algorithm 3.

In this algorithm, the returned variable is w as these are the relaxed and regularized coefficients of the functions. C is a map from x to w and is often the identity. κ the weight between x and w is often 1. This algorithm has also proven effective in solving some system identification problems, but it also has an advantage over STLS. SR3 was made to handle proximal operators associated with all $l_{[0,1]}$ norms, and even higher norms such as but not limited to the l_2 norm.

C. Limitations

It is important to note, however, that all of these methods have key problems that still make these problems difficult to solve. First, no measurement is perfect and a certain amount of noise in either the measurement of $\boldsymbol{X}$ and/or $\dot{\boldsymbol{X}}$ will cause this algorithm to fail. Typically, only a noisy $\boldsymbol{X}$ is measured the derivative must be found which can be difficult, however, a multitude of methods do exist to do this [15].

On top of this, one must know what functions to put in the library Θ. Although sparse regularization is used, often times with a large library incorrect solutions are found. Another issue is knowing the state dimension and which states are relevant for the system dynamics.

Algorithm 3 SR3

$x^k \leftarrow$ initGuess()
$w^k \leftarrow$ initGuess()
$A_{\text{stack}} \leftarrow \text{stack}(A, \kappa C)$
while $i < i_{max}$ **do**
  $b_{\text{stack}} \leftarrow \text{stack}(b, w)$
  $x^{k+1} \leftarrow \text{leastSquares}(A_{\text{stack}}, b_{\text{stack}})$
  $z^{k+1} \leftarrow Cx^{k+1}$
  $w^{k+1} \leftarrow \text{proximalOperator}(z^{k+1})$
  if $w^{k+1} - w^k > eps$ **then** ▷ w not converged
    $w^{k+1} \leftarrow w^k$
    $x^{k+1} \leftarrow x^k$
  else if N is odd **then**
    break
  end if
end while

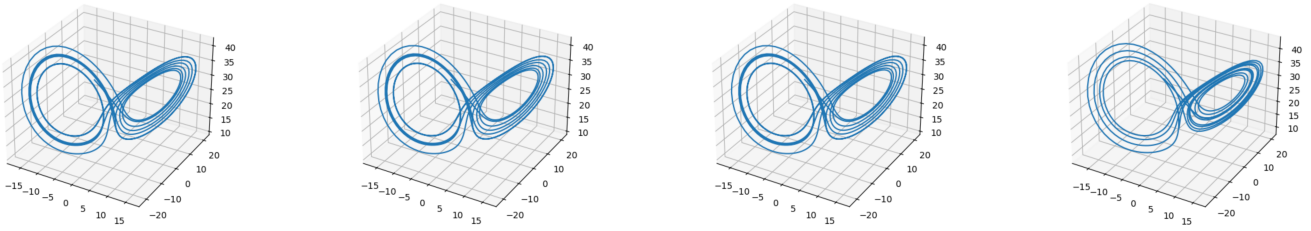
IV. Results

STLS and SR3 with both the l_0 and the l_1 proximal operator were ran for each of the test cases listed above, proximal gradient descent was not used as it proved to be ineffective for all but the most simple tasks. To find the best weights for SR3 a simple cross validation test was done, however, tuning by hand was the primary method for SR3 and STLS.

A. Lorenz System

For the Lorenz System all optimization methods were able to identify the system with some degree of accuracy. STLS identifies the system perfectly, along with SR3 with l_0 regularization, however, SR3 with l_1 has some errors as seen in figure 4

To test the effectiveness of the system with noise a zero mean gaussian white noise of standard deviations ranging up to 10 were applied to both the state measurements and the derivative measurements of the system. The norm of the error on the individual states are shown in figure 5. Despite the performance of SR3 with l_0 regularization in the case with no noise, when noise is added to the system the error is larger than all other attempted methods. It is also important to note that both STLS and SR3 with l_0 identify the exactly correct system initially, while SR3 with l_1 has error even with no noise. When noise is applied the error with SR3 with l_1 seems to increase at the same rate as STLS. This may seem like a good alternative to STLS, however, about 4% of the time SR3 with l_1 fails to produce a valid solution. Although STLS and



(a) True System (b) STLS Identified System (c) l_0 Regularized SR3 Identified System (d) l_1 Regularized SR3 Identified System

Fig. 4: True vs. Identified Lorenz Systems

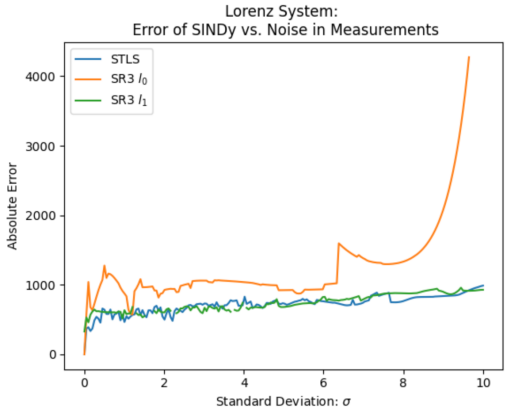


Fig. 5: Effect of Noise on SINDy Algorithms

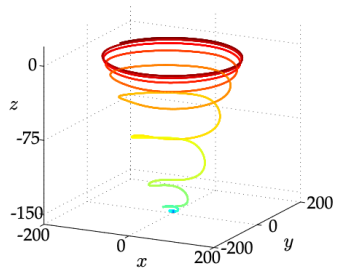


Fig. 6: True Mean Field Dynamics [16]

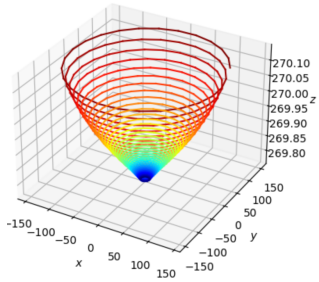


Fig. 7: Approximated Mean Field Dynamics

SR3 with l_1 have similar performances with noise, this fact makes STLS the better method.

B. Flow Past a Cylinder

The mean field model in eq. 3 is is one of the best 3 dimensional approximations of the flow field that exist. To predict this mean field model, it is also required to have a 3-dimensional system. To do this dimensionality reduction is needed. One of the most widely used method of dimensional reduction is Proper Orthogonal Decomposition (POD). The POD of the given the matrix $\boldsymbol{X}$ is given below.

$$X = U\Sigma V^* = ZV^* \tag{12}$$

In the decomposition above the POD modes are given by the rows of V^*. The modes are said to be constant throughout time and describe the most prominent features in the flow field. Z on the other hand are referred to as the modal coefficients. These coefficients determine how the POD modes evolve over time. To reduce the dimensionality, it is common practice to truncate the POD modes and the modal coefficients at the ith highest singular value, defined by Σ. Since the goal is approximating the mean field equations, $i = 3$. This results in a good approximation as this still captures 96.4% of the systems total energy.

Using the truncated POD modes and modal coefficients, one can use SINDy to determine how the modal coefficients evolve over time.

$$\dot{Z} = f(Z) \tag{13}$$

Using STLS an approximation of the mean field dynamics can be recovered as is shown in figures 6 and 7. Like the true mean field model STLS discovers mostly quadratic terms. STLS, however, was not able to fully recover the true dynamics and does identify terms that should not exist and even a few higher order terms. The primary reason for this is that the modes identified by POD are different from the modes presented in the original paper [16]. Indeed, with these different POD modes, different dynamics arise, dynamics that the typical mean field model do not capture.

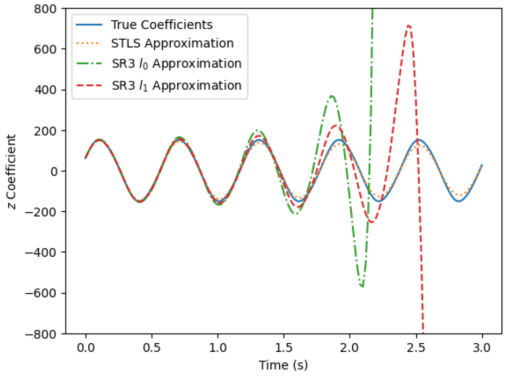
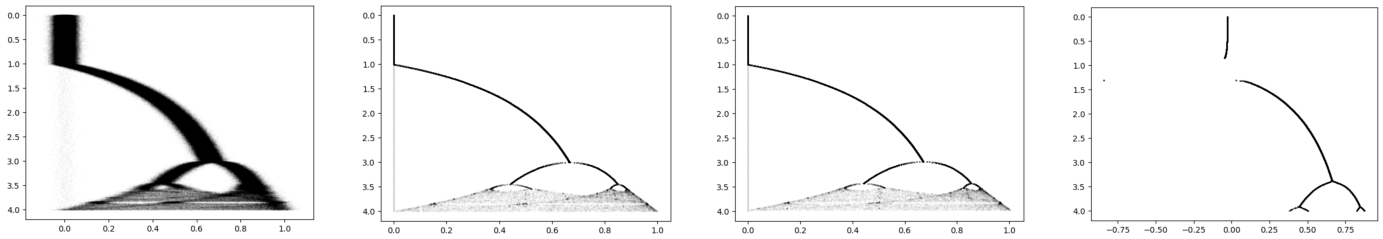


Fig. 8: Comparison of Optimization Methods on Predicting a Modal Coefficient's Evolution



(a) Noisy System (b) STLS Identified System (c) l_0 Regularized SR3 Identified System (d) l_1 Regularized SR3 Identified System

Fig. 9: Data vs. Identified Logarithmic Map

When comparing STLS to SR3 with either l_0 or l_1 the difference is astronomical as the latter can only produce usable solutions in very specific circumstances as seen by figure 8.

Note how the true dynamics of the modal coefficients in figure 8 are sinusoidal, and that the dynamics identified by STLS seems to be a decreasing sinusoid. In fact the mostly quadratic dynamics identified by STLS are not the true dynamics, and instead the true dynamics are nearly linear. STLS will not converge on this solution, however, unless all functions in the library are linear, otherwise the algorithm will overfit. As for the other optimization methods, in almost all cases their predictions of the dynamics are inaccurate.

C. Logistic Map

To test if the SINDy method works for discrete time systems with bifurcations and noise, the algorithms were applied to the logistic map seen in eq. 4. During the analysis noise was only applied to the target values, or the $\boldsymbol{X}_{k+1}$ values. Applying nearly any amount of noise to $\boldsymbol{X}$ will cause all algorithms to fail to identify the system. The original paper claims that the data was generated by a white noise forcing term in the dynamics, this is misleading as noise must be added after the fact. A high amount of process noise will cause the system itself to diverge as the attractors are not powerful enough to overcome even small amounts of noise.

Multiple trajectories were used to produce this data, with 1000 total corresponding to 100 initial conditions per value of μ where there were 10 values of mu. With zero mean gaussian white noise of standard deviation $\sigma = 0.025$ applied to $\boldsymbol{X}_{k+1}$ some algorithms were able to reconstruct the true system. As seen in figure 9 STLS and SR3 with l_0 were both able to reconstruct the system almost exactly. It is important to note though, that in the identified coefficients of the SR3 with l_0 identified system there where values that were not equal to 1 that described the dynamics of μ, this means the algorithm was not able to determine the static nature of μ throughout the timeseries. This was not the case with STLS. SR3 with l_1 performed particularly bad in this example and failed to identify the proper dynamics and only approached the correct dynamics when allowed to overfit by selecting a very low weight on regularization. The same behavior as SR3 with l_0 was present with this method as well, a failure to identify the constant nature of μ. However, in this case it was even more pronounced as the l_1 proximal operator shrunk the coefficients representing the dynamics of the μ value from where they should be.

V. Conclusion

SINDy as a whole is a promising algorithm with clear ability to determine systems with a wide variety of dynamics, including chaotic dynamics. SINDy, however, has a number of shortcomings that prevent it from being a silver bullet. SINDy requires knowledge of what variables are important enough to be states and an idea of the functions that underlie the dynamics. Another requirement is that only a low amount of noise acceptable in the measurements of the system, however, there are methods of mitigating the effect of noise on the SINDy algorithm. Many optimization methods for can be used for SINDy, however, in the three examples shown in this report STLS outperformed SR3, a promising method, in all cases. STLS seems to have a resilience to small amounts of noise and an affinity for narrowing down the correct dynamics. SR3 with l_0 and with l_1 regularization proved to be less robust than STLS when it came to noise resistance. The methods of SR3 also seemed prone to incorrect local minimum that prohibited the algorithm from converging on the correct solution.

VI. Future Work

One major improvement that is worth mentioning in this paper is incorporating a mixture of SR3 methods with the STLS algorithm. One major advantage STLS has over SR3 is that once function coefficients are deemed to be unnecessary to the solution, STLS stops optimizing over them and keeps them at zero. This is not the case with SR3 as a full least squares solution is found at every iteration causing coefficients that were zero to once again become non-zero. This does not need to be the case and a hybrid between the two methods may cause a marked increase in performance. NOTE: All code is given in the github repository located at [17]

References

[1] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, 1981.

[2] C. W. ROWLEY. Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, 15(03):997–1013, 2005.

[3] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 40(11):2323–2330, 2002.

[4] Jer-Nan Juang and Richard S. Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics*, 8(5):620–627, 1985.

[5] Peter Schmid and Jörn Sesterhenn. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656, 11 2008.

[6] B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[7] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.

[8] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[9] 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.

[10] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eurika Kaiser, and J. Nathan Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8(1), May 2017.

[11] BERND R. NOACK, KONSTANTIN AFANASIEV, MAREK MORZYŃSKI, GILEAD TADMOR, and FRANK THIELE. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497:335–363, 2003.

[12] Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing, 2008.

[13] Peng Zheng, Travis Askham, Steven L. Brunton, J. Nathan Kutz, and Aleksandr Y. Aravkin. A unified framework for sparse relaxed regularized regression: Sr3. *IEEE Access*, 7:1404–1423, 2019.

[14] Sparse relaxed regularized regression open source implementation. https://uw-amo.github.io/AMO_Site/software/sr3/.

[15] Python derivative. https://pypi.org/project/derivative/.

[16] 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, March 2016.

[17] Github repository. https://github.com/Bwodetzki/SINDy.