Sparse Learning of Multi-Agent Systems from Data

Hao-Tien Chuang*, Dongyang Li†, Shelby Malowney*, Ritwik Trehan†
Mentors: Jea-Hyun Park and Sui Tang ‡

August 2021

Abstract

Multi-agent systems have found wide applications in science and engineering ranging from opinion dynamics to predator-prey systems. A fundamental, challenging problem encountered in these areas is to reveal the link between the collective behaviours and interaction laws. In this report, we leverage recent advancements made within the mathematical modeling community and on sparsity-promoted algorithms in machine learning to consider the data-driven discovery of multi-agent systems with non-local interactions laws dependent on pairwise distances. We adapt the "Sparse Identification of Nonlinear Dynamics" (SINDy) algorithm to approximate interaction laws on systems of interacting agents and demonstrate the effectiveness of the algorithm in various cases.

Keywords – Sparse learning, Interacting particle systems, Data-driven methods

1 Introduction

Multi-agent systems are ubiquitous in science and engineering. The individual interactions among agents can produce a rich variety of collective motions with visually compelling patterns, such as crystallization of particles, clustering of peoples' opinions in social events, and coordinated movements of ants, fish, birds, and cars. However, the interaction laws of agents often remain elusive. It has been a long-standing science problem in various disciplines to reveal the links between the collective behaviour and the individual interaction laws [3, 10, 13, 16].

A common belief in scientific discovery is that complicated collective behaviors are consequences of simple interaction rules, e.g., only depending on pairwise distance. In the mathematical modeling community, there have been tremendous research efforts

*Department of Mathematics, University of California, Los Angeles, CA 90095, USA (**tchuang@ucla.edu**)

†Department of Mathematics, University of California, Santa Barbara, CA 93106, USA (**dongyang_li@ucsb.edu**)

*Department of Mathematics, University of California, Santa Barbara, CA 93106, USA (**smalowney@ucsb.edu**)

†Department of Mathematics and Department of Physics, University of California, Santa Barbara, CA 93106, USA (**ritwik@ucsb.edu**)

‡Department of Mathematics University of California, Santa Barbara, CA 93106, USA (**jhpark1@ucsb.edu suitang@ucsb.edu**)

to model collective dynamics using interaction laws based on pairwise distances. Researchers derive the governing equations for multi-agent systems by combining the fundamental physical law with relatively simple parametric families of elementary interaction functions. The goal is to find certain conditions on the interaction functions such that the underlying dynamical systems are well-posed and the the asymptotic behaviour of the solutions reproduce similar qualitative macroscopic patterns as observed in reality, such as flocking [8], crystallization [12, 17], clustering[4, 11, 14], milling [1, 2, 7].

Recent rapid advancements in the data information technology (especially in digital imaging and high-resolution lightweight GPS devices and particle tracking methods) allow gathering high resolution trajectories of individual agents. This inspired the urgent need for devising efficient data-driven methods to uncover the governing laws. The objective is to turn data into interaction functions that are not just predictive, but also provide physical insight into the nature of the underlying system from which the data was generated. Machine learning algorithms are particularly promising towards achieving this goal. This is clearly supported by their tremendous empirical success in the scientific discovery of many fields, including computer vision, health care, and natural language processing. In particular, sparse regression techniques have drawn a lot of recent attention since many complex systems have simple algebraic representations corresponding to a sparse representation in high dimensional nonlinear functional spaces. By applying them to dynamical systems and partial differential equations, it may determine the fewest terms in the dynamic governing equations required to accurately represent the data. This results in parsimonious models that balance accuracy with model complexity to avoid over-fitting.

In this work, we leverage the recent advancement on the mathematical modeling of multi-agent systems and sparse regression techniques for identifying nonlinear dynamical systems called "SINDy" (Sparse Identification of Nonlinear Dynamics) [6]. We consider the data-driven discovery of interaction laws in a family of homogeneous agent systems that are derived from fundamental physical laws with an unknown interaction function modeling the pairwise distance-based interactions. We also adopt sparsity-promoting techniques to approximate the interaction functions from a large known library of template functions.

2 Problem Settings

We start with a first-order homogeneous agent system consisting of N agents in $\mathbb{R}^d$, interacting according to

$$\dot{\mathbf{x}}_i(t) = \frac{1}{N}\sum_{i'=1}^{N} \phi(\|\mathbf{x}_i(t) - \mathbf{x}_{i'}(t)\|)(\mathbf{x}_i(t) - \mathbf{x}_{i'}(t)), \tag{1}$$

where $i = 1, 2, ..., N$; $\mathbf{x}_i(t) \in \mathbb{R}^d$; $\|\mathbf{x}_i(t) - \mathbf{x}_{i'}(t)\|$ is the Euclidean norm, and ϕ is the interaction law, or kernel, which weighs the vector difference $\mathbf{x}_i(t) - \mathbf{x}_{i'}(t)$.

The form of the governing equation is derived using the Newton's second law where the mass of agents is assumed to be zero. The evolution of agents is governed by minimizing the potential energy function, depending on pairwise distance,

$$U(\mathbf{x}_1, \cdots, \mathbf{x}_N) = \frac{1}{2N} \sum_{i,i'=1,1}^{N,N} \Phi(\|\mathbf{x}_i - \mathbf{x}_{i'}\|)$$

so that the interaction force is the derivative of the potential function U and $\phi(r) = \frac{\Phi'(r)}{r}$. This kind of system has found applications in modeling opinion dynamics in social science or particle dynamics in physics and chemistry [9, 15].

In modeling problems, a key challenge lies in the choice of ϕ to induce the desired collective behaviours, since we have very limited knowledge of the underlying complex system and there is no canonical choice of potential functions for many complex agent systems. Motivated by the increasing availability of high resolution trajectory data sets, we are interested in designing efficient data-driven methods to identify ϕ from data.

One important observation is that many ad hoc potential functions that have found empirically successful applications in multi-agent systems, such as the Leonard Jones potential and Morse potential, consists of only a few items, making it sparse in the space of possible functions. This inspired the idea of using the sparse regression techniques to look for a sparse representation of ϕ in a large library of candidate functions.

To begin, we collect a time history of the state $\mathbf{X}(t) = \left[\mathbf{x}_1(t), \cdots, \mathbf{x}_N(t)\right]$ and either measure its derivative $\dot{\mathbf{X}}(t)$ or numerically approximate it from $\mathbf{X}(t)$. The trajectory data for each agent is sampled at different time instances $t_1, t_2, \cdots, t_L$ and organized into two matrices, where columns contain each dimension of all agents evolved over time.

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}(t_1) \\ \vdots \\ \mathbf{X}(t_L) \end{bmatrix} \in \mathbb{R}^{LNd} \quad \text{and} \quad \dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{X}}(t_1) \\ \vdots \\ \dot{\mathbf{X}}(t_L) \end{bmatrix} \in \mathbb{R}^{LNd} \tag{2}$$

Next, we construct a library consisting of candidate nonlinear functions of ϕ. For example, say $\Theta = \{1, r, r^2, \cdots, \sin r, \cos r\}$. For any 1D function φ defined in the positive axis of the real line, we introduce a map $\mathbf{f}$ that $\mathbf{f}_\varphi$ defines a interaction force field of the same form with the right hand side of (1) where we replace ϕ with φ. In other words, for $\mathbf{X} \in \mathbb{R}^{dN}$, the i-th component of $\mathbf{f}_\varphi$ is written as

$$[\mathbf{f}_\varphi(\mathbf{X})]_i = \frac{1}{N} \sum_{i'=1}^{N} \varphi(\|\mathbf{x}_i(t) - \mathbf{x}_{i'}(t)\|)(\mathbf{x}_i(t) - \mathbf{x}_{i'}(t)) \in \mathbb{R}^d.$$

A crucial note is that if ϕ has a sparse representation in Θ, then so does the interaction force function $\mathbf{f}_\phi$ in the dictionary $\mathbf{f}_\Theta$. Leveraging this observation, we may set up a sparse regression problem for learning ϕ by solving the system of equations

$$\dot{\mathbf{X}} = \mathbf{f}_\Theta(\mathbf{X})\mathbf{c}, \tag{3}$$

where $\mathbf{c}$ is a sparse vector of coefficients that determine which nonlinearities are active.

Realistically, often only the data $\mathbf{X}$ is available, and $\dot{\mathbf{X}}$ must be approximated numerically, such as using the finite difference method. We will also investigate the case where our data is contaminated with noise, solving the noise perturbed version of (3):

$$\dot{\mathbf{X}} = \mathbf{f}_\Theta(\mathbf{X})\mathbf{c} + \eta\mathbf{Z}, \tag{4}$$

where $\mathbf{Z}$ is a matrix of independently identically distributed Gaussian entries with zero mean, and η is the noise magnitude.

The sparse regression problem studied in this paper is different from [6] where the goal is to looking for the sparse representation for each row of the governing equations. Here we have coupled dynamical systems with non-local interactions governed by ϕ. We promote the sparsity of ϕ which is the same for all rows of governing equations.

3 Methodologies to solve sparse regression problem

Though there are various regression methods to accomplish our proposed solution, this report discusses the following three methods in depth: Least Squares, Sequential Least Squares, and LASSO [5, p. 15, 98, 248].

3.1 Least Squares (LS)

Given the equation (3), the Least Squares method seeks to find the solution $\mathbf{c}$ such that the sum-squared error $\|\dot{\mathbf{X}} - \mathbf{f}_\Theta(\mathbf{X})\mathbf{c}\|_2^2$ is minimized. Considering M sets of initial conditions and L time steps, we minimize the sum-squared error by first letting

$$g(\mathbf{c}) = \sum_{m=1}^{M}\sum_{l=1}^{L} \|\dot{\mathbf{X}}^{(m)}(t_l) - \mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))\mathbf{c}\|_2^2$$

We can then expand out the L_2-norm squared

$$\begin{aligned} g(\mathbf{c}) &= \sum_{m=1}^{M}\sum_{l=1}^{L} (\dot{\mathbf{X}}^{(m)}(t_l) - \mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))\mathbf{c})^T(\dot{\mathbf{X}}^{(m)}(t_l) - \mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))\mathbf{c}) \\ &= \sum_{m=1}^{M}\sum_{l=1}^{L} \dot{\mathbf{X}}^{(m)}(t_l)^T\dot{\mathbf{X}}^{(m)}(t_l) - 2\mathbf{c}^T\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\dot{\mathbf{X}}^{(m)}(t_l) + \mathbf{c}^T\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))\mathbf{c} \end{aligned}$$

Then take the partial derivative of $g(\mathbf{c})$ with respect to $\mathbf{c}$, set the equation equal to 0 for a local minimum of the error, and solve for $\mathbf{c}$.

$$\frac{\partial g(\mathbf{c})}{\partial \mathbf{c}} = \sum_{m=1}^{M}\sum_{l=1}^{L} -2\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\dot{\mathbf{X}}^{(m)}(t_l) + 2\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))\mathbf{c} = 0$$

$$\sum_{m=1}^{M}\sum_{l=1}^{L}\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))\mathbf{c} = \sum_{m=1}^{M}\sum_{l=1}^{L}\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\dot{\mathbf{X}}^{(m)}(t_l)$$

$$\mathbf{c} = \underbrace{\left(\sum_{m=1}^{M}\sum_{l=1}^{L}\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))\right)^{-1}}_{\text{(left)}}\underbrace{\left(\sum_{m=1}^{M}\sum_{l=1}^{L}\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\dot{\mathbf{X}}^{(m)}(t_l)\right)}_{\text{(right)}}$$

In code, this is done by summing $\left(\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))\right)$ and $\left(\mathbf{f}_\Theta(\mathbf{X}^{(m)}(t_l))^T\dot{\mathbf{X}}^{(m)}(t_l)\right)$ separately for each M sets of initial conditions and L time-steps. Finally, $\mathbf{c}$ is found by taking the pseudo-inverse of the "left" and matrix multiplying with "right".

3.2 Sequential Least Squares (SLS)

Sequential Least Squares or Sequential thresholded least-squares is a modified version of Least Squares. An initial guess on the form of $\mathbf{c}$ is made by taking the Least Squares. Next, the algorithm checks for any coefficients in the coefficient vector that are below a certain threshold λ which is manually set. Functions corresponding to these small coefficients are removed from the approximation by setting the coefficients exactly to zero and the Least Squares approximation is performed again and again with the remaining $\mathbf{f}_\varphi$. After no coefficients are below the threshold, the coefficient vector has been found.

3.3 Least Absolute Shrinkage and Selection Operator (LASSO)

The Least Absolute Shrinkage and Selection Operator is a sparse regression technique that uses an L_1-norm penalized term to balance model complexity with accuracy. In contrast to Least Squares regression, which returns only nonzero coefficients, LASSO can can ensure that the $\mathbf{c}$ vector does not have insignificant constants even with an over-fitted model. Section 4 can be referenced to show concrete example of the difference between Least Squares and LASSO regression. Similar to Least Squares, LASSO finds a sparse coefficient vector $\mathbf{c}$ by minimizing

$$\|\dot{\mathbf{X}} - \mathbf{f}_\Theta(\mathbf{X})\mathbf{c}\|_2^2 + \lambda\|\mathbf{c}\|_1$$

In code, LASSO is done by using the implemented Matlab function "lasso(A,b,'CV',k)", where it solves for x in the equation of the form $Ax = b$ using k-fold cross validation.

4 Numerical Results

Numerical Setup We simulate trajectory data over the observation time $[t_1, t_L]$ with M different initial conditions at L equidistant time samples in the time interval $[t_1, t_L]$. All ODE systems are evolved using ode15s in Matlab with a relative tolerance at 10^{-5} and absolute tolerance at 10^{-6}.

Choice of dictionary In our numerical examples, we shall use dictionaries consists of elementary functions such as (orthogonal) polynomials, rational functions, and trigonometric functions.

Software package The algorithm will be numerically implemented and made available to the public here.

Outline In the following section, we are going to run the three aforementioned regression methods on systems with different interaction laws to gauge their performance and effectiveness in learning ϕ.

1. The user inputs parameters needed to generate the system such as dimension, number of agents, time span, number of timesteps, number of initial conditions, and the interaction law ϕ. A library containing candidate terms φ is also created.
2. With the first-order dynamical system (1), trajectory data $\mathbf{X}$ is generated and used to numerically approximate $\dot{\mathbf{X}}$ for the regression problem.
3. $\mathbf{c}$ is found using regression techniques and approximations for ϕ are created.
4. Trajectories and approximated kernels are plotted and the accuracy of approximated kernels are evaluated by taking the L_2-norm of the error, the difference between ϕ and its approximation.

4.1 Example 1: $\phi(r) = r^2$

First, we set parameters to create the system. We will be working with 10 agents in 2 dimensions, with 100 time-steps between 0 and 10. We will also only consider 1 set of initial conditions.

N	d	$[t_1$	$t_f]$	L	M
10	2	[0	10]	100	1

Table 1: (Example 1) Parameters for the system

For the following 3 examples, 4.1-4.3, we will be using the library of candidate functions below, which includes r to the power of 0 through 5, $\sin(r)$, and $\cos(r)$.

$$\left[1, r, r^2, ..., r^5, \cos(r), \sin(r)\right]$$

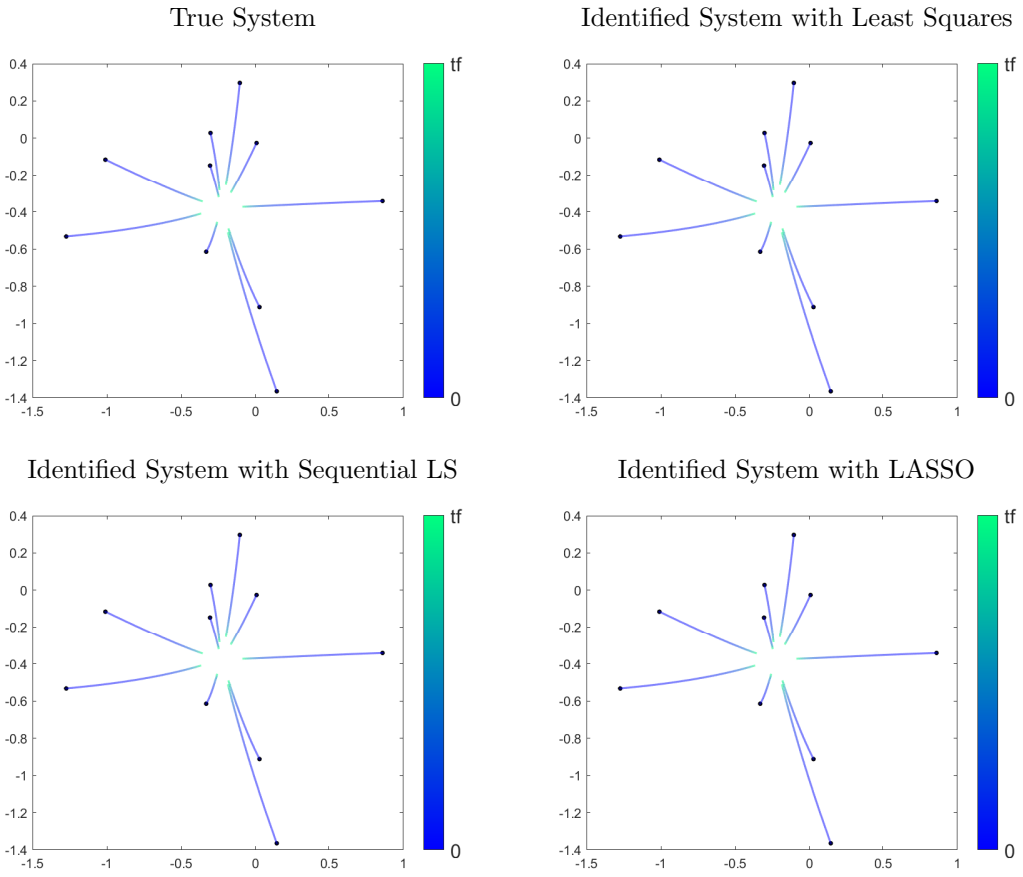

Figure 1: Trajectory visualization for each method. The agents start at an initial position, represented by black dots, and follow a path evolved over time, represented by the changing color. Initially, r is larger so $\phi(r)$ weighs the velocities of the particles heavily towards each other. The agents slow down over time as pairwise distances close in.

Approximation Method	Approximated $\phi(r)$	L^2 error
LS	4.6e-05$-$2.6e-05r+0.99998r^2 + ...	1.2884e-06
SLS	r^2	0
LASSO	r^2	9.1078e-15

Table 2: Approximated $\phi(r) = r^2$ and L^2 errors

Through Least Squares, all 8 candidate functions are required when approximating $\phi(r)$. However, the error is still low since the coefficients for all other candidate functions besides $r.^2$ are accurately pinpointed to be close to 0. Sequential Least Squares and LASSO introduces sparsity to entirely eliminate these low values and both almost exactly approximates $\phi(r)$. Regardless of the complexity of the approximation, all methods are comparably accurate from a qualitative analysis of the trajectory and kernel plots.

4.2 Example 2: $\phi(r) = 2r^2 + 3r^5$

For the second example, we let $\phi(r) = 2r^2 + 3r^5$ which is contains multiple terms still within the library. This is done to distinguish the effectiveness between Sequential Least Squares and LASSO, which are both sparse regression techniques.

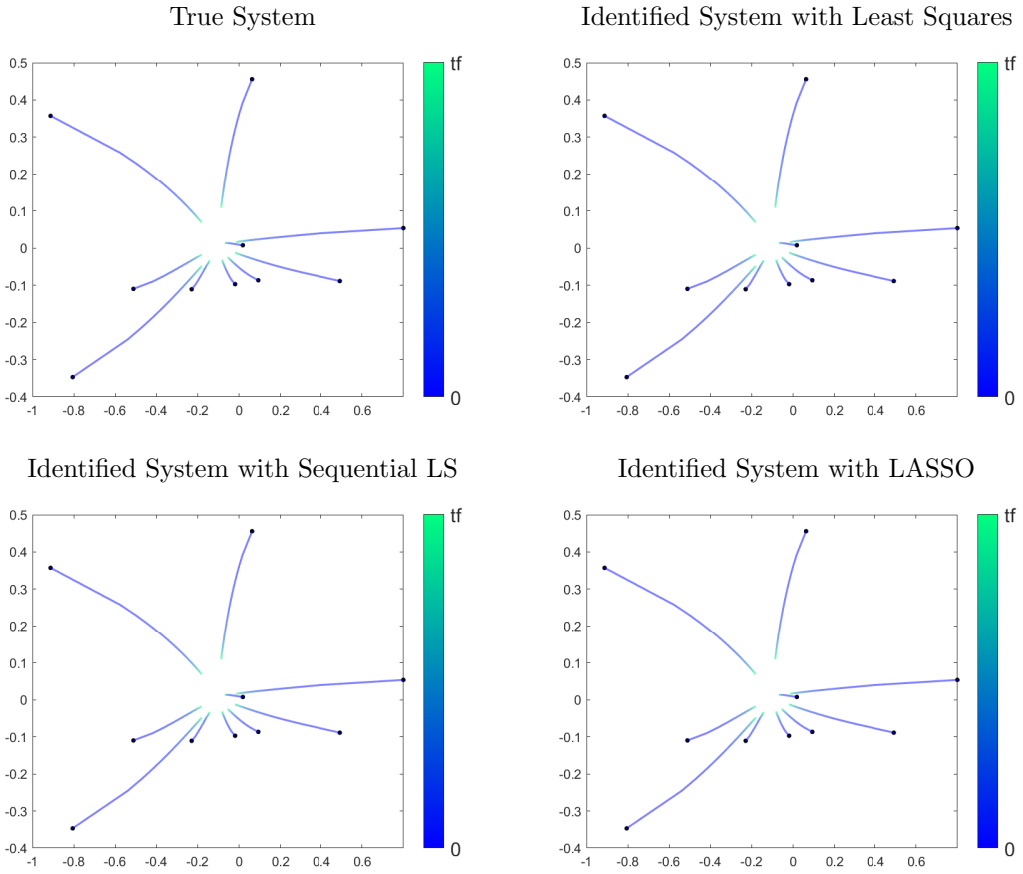

Figure 2: Trajectory visualization for each method with $\phi(r) = 2r^2 + 3r^5$

Approximation Method	Approximated $\phi(r)$	L^2 error
LS	$0.024 + \ldots + 1.9878r^2 + \ldots + 2.9997r^5 + \ldots$	9.8887e-06
SLS	$2r^2 + 3r^5$	5.0537e-13
LASSO	$2r^2$+9.7145e-15$r^4 + 3r^5$	7.0334e-13

Table 3: Approximated $\phi(r) = 2r^2$+$3r^5$ and L^2 errors. The Least Squares approximation is mostly omitted due to using all the terms.

Again, for Least Squares, all 8 candidate functions are required in the approximation. LASSO is again sparser than Least Squares, approximating $\phi(r)$ correctly, with the

exception of the low r^4 coefficient. For SLS, this small r^4 coefficient is not present. Since Sequential Least Squares was defined with a threshold value larger than all the trivial coefficients in the Least squares result, the r^4 coefficient was subsequently removed from the approximation.

4.3 Example 3: $\phi(r) = \cos(\frac{\pi r}{2}), \quad 0 < r < 1$

For the third example, we wanted to choose our function $\phi(r)$ to be outside the library. This is to simulate the case in which a governing law beyond the expectation of what we may include within the library is encountered. With the chosen $\phi(r)$, particles' velocities become increasingly more weighted as their pairwise distances decrease, up to 1 as r approaches 0. If the distance between two particles are over 1, they do not interact.

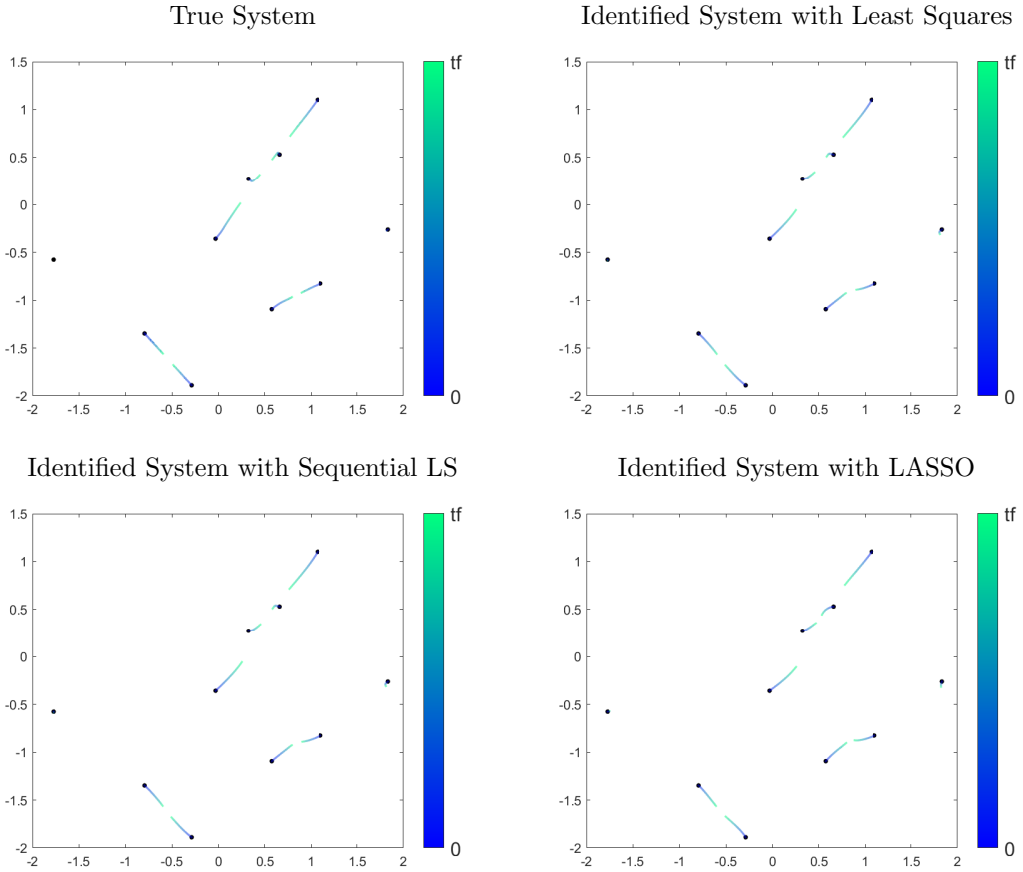

Figure 3: Trajectory visualization for each method with $\phi(r) = \cos(\frac{\pi r}{2}), \quad 0 < r < 1$

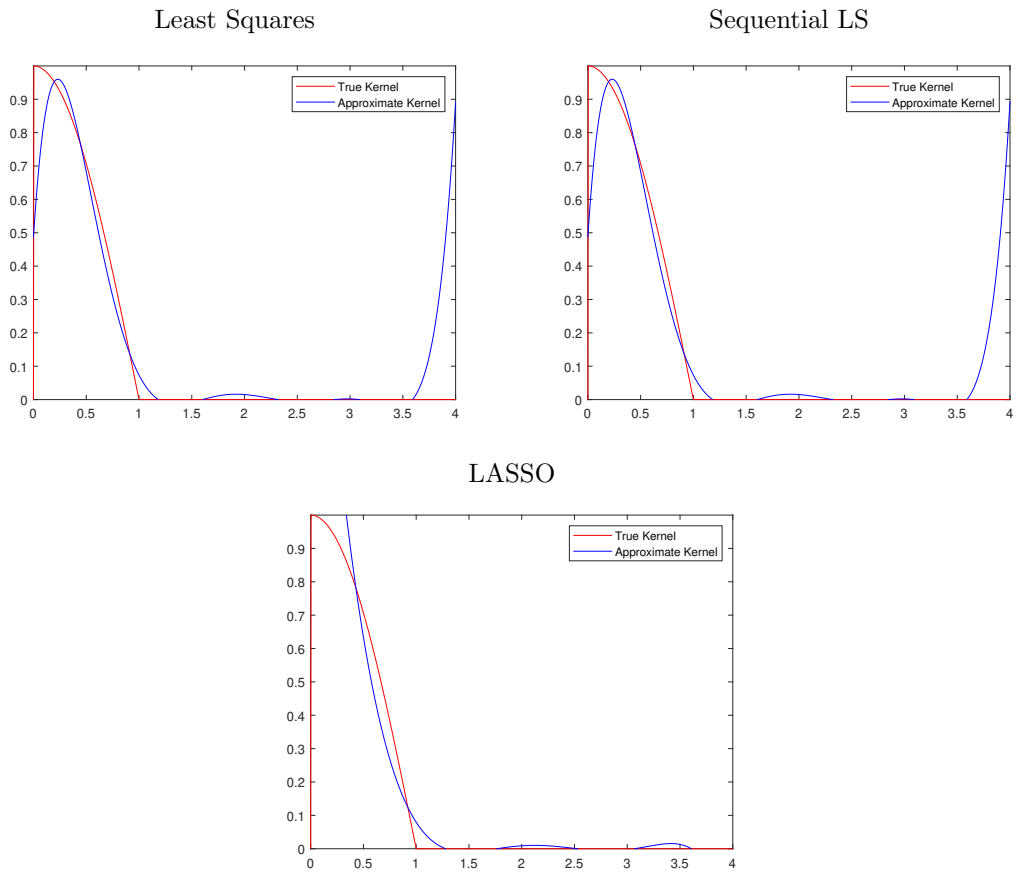

Figure 4: Comparisons between the true kernel function and approximated kernel functions for each method.

Candidate functions	Least Squares	Sequential LS	LASSO
1	-222.3525	-222.3525	9.0435
r	-120.6094	-120.6094	-10.7351
r^2	95.5764	95.5764	0
r^3	39.2875	39.2875	0
r^4	-19.6963	-19.6963	0.37069
r^5	1.8993	1.8993	-0.059665
$\sin(r)$	125.456	125.456	6.1245
$\cos(r)$	222.8397	222.8397	-6.8361

Table 4: Coefficients corresponding to candidate functions in the library generated from each method.

Approximation Method	L^2 error
LS	1.54054
SLS	1.54054
LASSO	2.88242

Table 5: L^2 errors for each method.

Least Squares and Sequential Least Squares both share the same coefficient vector. Sequential Least Squares is not able to make a more accurate model than the Least Squares because all coefficients are above the threshold value. LASSO, however, is able to make a sparser model with the trade-off of less accuracy.

4.4 Example 4: $\phi(r) = e^r$

The final case is considered for approximating continuous functions that are not in the given library of candidate functions. In addition, e^r has an interesting form in which we can use to validate the generated coefficient vectors. Recall the series expansion,

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

To replicate the series and capture the coefficients as best as possible, we choose the library of functions

$$\begin{bmatrix} 1 & r & r^2 & r^3 & \dots & r^{10} \end{bmatrix}$$

with the expectation that the coefficients will match up with each of the coefficients in the series expansion of e^x.

Candidate functions	Least Squares	Sequential LS	LASSO
1	1	1.0458	0.97589
r	1	0.63894	1.204
r^2	0.49998	1.2664	0
r^3	0.16675	-0.44167	0.67471
r^4	0.041531	0.21908	-0.19326
r^5	0.0084625	0	0.053236
r^6	0.0013257	0	0
r^7	0.00020614	0	0
r^8	3.2031e-05	0	0
r^9	-9.7072e-07	0	0
r^{10}	9.2955e-07	0	0

Table 6: As expected, the Least Squares method outputs a non-sparse vector, meaning all the candidate terms were used. Out of the three methods, Least Squares most accurately approximated each of the coefficients up to the first 8 terms in the series and begins to drop off in accuracy afterward.

4.5 Adding noise

In reality, collected data is often contaminated with noise. To add noise, we create a matrix $\mathbf{Z}$ of independently identically distributed Gaussian entries with zero mean and noise magnitude η as shown in (4). With the inclusion of noise on the trajectory data, LASSO remains generally more robust than LS and SLS, but there still exists a noticeable increase in error for all three methods.

4.5.1 $\phi(r) = r^2$ with added noise of magnitude $\eta = 0.01$

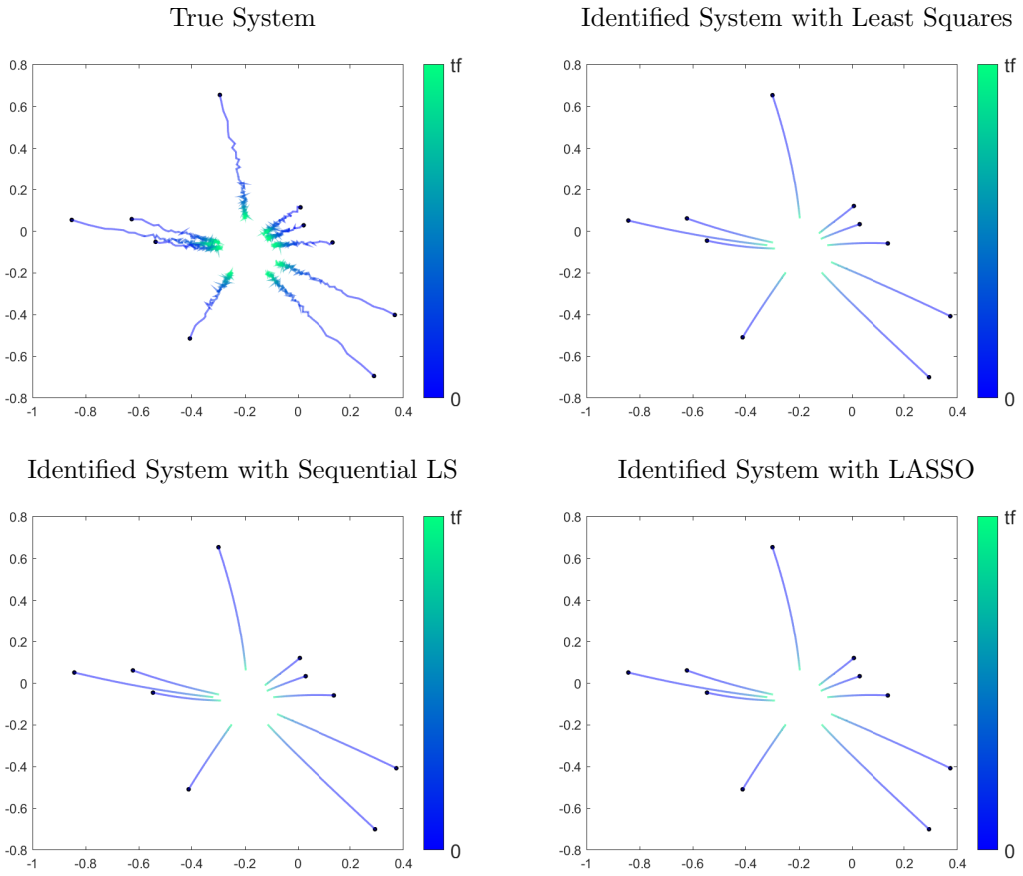

Figure 5: Noise is added to both $\mathbf{X}$ and $\dot{\mathbf{X}}$ as shown in the True System.

Approximation Method	L^2 error
LS	39.6497
SLS	39.6497
LASSO	0.5540

Table 7: L^2 errors for each method.

4.5.2 $\phi(r) = \cos(\frac{\pi r}{2}), \quad 0 < r < 1$ with added noise of magnitude $\eta = 0.01$

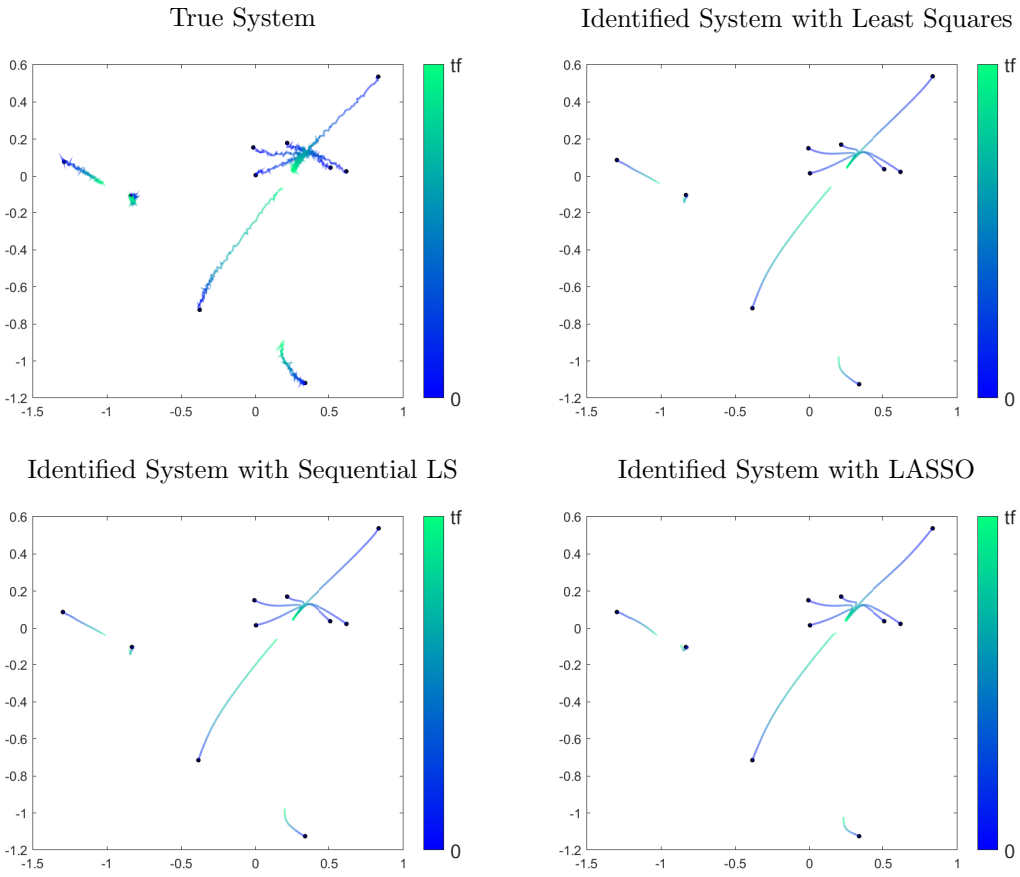

Figure 6: Noise is added to both $\mathbf{X}$ and $\dot{\mathbf{X}}$ as shown in the True System. There is a noticeable discrepancy among the trajectories between LASSO and LS/SLS for the second leftmost agent. It is unclear which is correct due to noise on the True System.

Approximation Method	L^2 error
LS	14.6955
SLS	14.6955
LASSO	4.7336

Table 8: L^2 errors for each method.

5 Discussion

To summarize, we first considered a homogeneous agent system represented by a system of first-order ODEs. Agents in the system interact with one another through an unknown interaction law ϕ dependent on pairwise distances. This model has applications from opinion dynamics in social science to particle dynamics in physics. Given the parameters needed to create the system and an interaction law, we generate trajectory data $\mathbf{X}$ and its derivative $\dot{\mathbf{X}}$, which will then be used to simulate data collected from a real-world scenario. With the assumption that only these two data matrices are known, we may set up a sparse regression problem to learn ϕ from a library consisting of candidate nonlinear functions φ.

For the regression problem, we consider three regression techniques: LS, SLS, and LASSO. For all methods, we could calculate comparably accurate approximations of the interaction law, but some methods worked better than others depending on ϕ. With the examples examined in this paper, we found that for the three methods, given the same problem, SLS is always sparser than LS, unless the thresholding parameter λ is greater than all entries in the coefficient vector. Additionally, LASSO is often sparser than LS because of the sparsity penalty term, but sometimes at the cost of accuracy. Sparsity is necessary since many complex systems have simple algebraic representations, meaning only few important terms make up the interaction law and govern the dynamics of the agent system. Ideally, we seek to find a balance between both accuracy and sparsity.

One important observation was that there was more trouble with the piece-wise interaction law in Section 4.3. For the kernel plots, the Least Squares and SLS methods had complex and poor approximations that could not capture the change in function behavior between intervals with the given library of candidate functions. Similarly, the LASSO approximation was not simple, but did better in approximating the horizontal line for $r > 1$ than the other two methods. Something to consider for the future is to detect discontinuities or non-differentiable points and return multiple coefficient vectors applied on corresponding intervals.

Several other ways to move forward with this research may be to include a larger selection of functions within the library, implementing other sparse regression techniques, or to use PDEs instead of systems of ODEs. Though not presented in this work, our research has also continued in this subject for the heterogeneous case by adding a second type of agent with separate interaction laws, which may be useful in predator-prey interactions or particles of different properties.

6 Acknowledgments

We would like to thank Dr. Sui Tang and Dr. Jea-Hyun Park for being our mentors through this experience and providing valuable insight and feedback. Our sincere gratitude also goes to Dr. Maribel Bueno for organizing this research program including its variety of events and conferences. Also, we extend our thanks to Dr. Steven Brunton

and his collaborators for their resourceful PNAS paper to reference. This research group acknowledges the University of California, Santa Barbara as its host institution from which our mentors and supervisors affiliate with. Finally, research was made possible for Hao-Tien Chuang and Shelby Malowney through the National Science Foundation grant DMS-1850663.

References

[1] Nicole Abaid and Maurizio Porfiri. Fish in a ring: spatio-temporal pattern formation in one-dimensional animal groups. *Journal of The Royal Society Interface*, 7(51):1441–1453, 2010.

[2] Giacomo Albi, D Balagué, José A Carrillo, and J32150701305 von Brecht. Stability analysis of flock and mill rings for second order models in swarming. *SIAM Journal on Applied Mathematics*, 74(3):794–818, 2014.

[3] Michele Ballerini, Nicola Cabibbo, Raphael Candelier, Andrea Cavagna, Evaristo Cisbani, Irene Giardina, Vivien Lecomte, Alberto Orlandi, Giorgio Parisi, Andrea Procaccini, et al. Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *Proceedings of the national academy of sciences*, 105(4):1232–1237, 2008.

[4] Vincent D Blondel, Julien M Hendrickx, and John N Tsitsiklis. On krause's multi-agent consensus model with state-dependent connectivity. *IEEE transactions on Automatic Control*, 54(11):2586–2597, 2009.

[5] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[6] 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.

[7] Yao-Li Chuang, Maria R D'orsogna, Daniel Marthaler, Andrea L Bertozzi, and Lincoln S Chayes. State transitions and the continuum limit for a 2d interacting, self-propelled particle system. *Physica D: Nonlinear Phenomena*, 232(1):33–47, 2007.

[8] Felipe Cucker and Steve Smale. Emergent behavior in flocks. *IEEE Transactions on automatic control*, 52(5):852–862, 2007.

[9] Massimo Fornasier, Jan Haškovec, and Jan Vybíral. Particle systems and kinetic equations modeling interacting agents in high dimension. *Multiscale Modeling & Simulation*, 9(4):1727–1764, 2011.

[10] Yael Katz, Kolbjørn Tunstrøm, Christos C Ioannou, Cristián Huepe, and Iain D Couzin. Inferring the structure and dynamics of interactions in schooling fish. *Proceedings of the National Academy of Sciences*, 108(46):18720–18725, 2011.

[11] Ulrich Krause et al. A discrete nonlinear and non-autonomous model of consensus formation. *Communications in difference equations*, 2000:227–236, 2000.

[12] Mathieu Lewin and Xavier Blanc. The crystallization conjecture: a review. *EMS Surveys in Mathematical Sciences*, 2(2):255–306, 2015.

[13] Ryan Lukeman, Yue-Xian Li, and Leah Edelstein-Keshet. Inferring individual rules from collective behavior. *Proceedings of the National Academy of Sciences*, 107(28):12576–12580, 2010.

[14] Sebastien Motsch and Eitan Tadmor. Heterophilious dynamics enhances consensus. *SIAM review*, 56(4):577–621, 2014.

[15] Stacy Patterson and Bassam Bamieh. Interaction-driven opinion dynamics in online social networks. In *Proceedings of the First Workshop on Social Media Analytics*, pages 98–105, 2010.

[16] David JT Sumpter. *Collective animal behavior*. Princeton University Press, 2010.

[17] Elena Vedmedenko. *Competing interactions and pattern formation in nanoworld*. John Wiley & Sons, 2007.