7-31-2022

# Data-Driven Equations for Coevolving Network Systems

Nicholas John
ndj3315@rit.edu

# Data-Driven Equations for Coevolving Network Systems

by

Nicholas John

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Applied Mathematics
School of Mathematical Sciences, College of Science

Rochester Institute of Technology
Rochester, NY

Date

July 31, 2022

# Committee Approval:

R·I·T
College of
SCIENCE

---

Nishant Malik, Ph.D. Date

School of Mathematical Sciences

Thesis Advisor

---

Mary Lynn Reed, Ph.D. Date

School of Mathematical Sciences

Committee Member

---

Matthew Hoffman, Ph.D. Date

School of Mathematical Sciences

Committee Member

---

Michael Cromer, Ph.D. Date

School of Mathematical Sciences

Director of Graduate Program

# Abstract

The framework of coevolving networks is a tool to model large-scale interacting dynamical systems undergoing change in connective structure, describing phenomena such as epidemic spreading on social networks with individuals changing their connections to avoid infection. Along with direct computational simulation, coevolving network systems are often formulated in terms of systems of ordinary differential equations in their descriptive statistics. The equation approach reduces the computational burden of analyzing the systems, but deriving equations becomes difficult as the underlying model becomes more complicated and as the desire for accuracy increases. We present an approach to construct equations for coevolving network systems automatically, using data from computational simulations and a formulation of sparse model identification. Using this approach we construct a data-driven system of equations for a coevolving SIS model that reproduces system behavior in both temporal evolution and dependence of steady states on system parameters.

# Acknowledgements

I am thankful for the help of many people over the last two years or so. Some of the standouts are:

Family, friends, teammates, coaches, instructors, and classmates, and everyone else who gave support and encouragement.

My peers in the Complexity Lab who listened to me present on this material, who knows how many times. Some of whom are Jason LaRuez, Kory Schimmelpfennig, Vincent Mei, Adam Giammarese, Kamal Rana, Jenna McDanold, Matthew Madsen, Luke Nearhood, Ivan Jacobs, and Ben Grande. Thank you for putting up with me and for asking thoughtful questions. I look forward to seeing what you all do with your wicked skills and motivation. Don't be strangers!

David Messinger of RIT, for generously meeting with me when I was a younger student looking for research opportunities. My advisor for this thesis is someone you suggested I talk to.

Nate Cahill of RIT, similarly for meeting with me to talk about research opportunities, for giving advice and support for my career, and for teaching me a lot of mathematics.

Michael Cromer of RIT, for giving useful academic advising and directing a worthwhile degree program.

Mary Lynn Reed and Matthew Hoffman of RIT, for generously serving on my thesis committee. I know that both of you have a lot going on, so I strive to make it worth your while.

Guiping Hu and Ivan Jacobs from RIT for asking about the elastic net after two of my presentations. Its inclusion ended up helping!

Andrés Ávila Barrera of Universidad de la Frontera, for providing helpful commentary during weekly meetings last year.

John McCluskey of RIT, for supervising a helpful research project concurrent with this one.

Erik Bollt of Clarkson University, for generously sharing your time and giving insights on my work. I ended up using CVX and developing my own library for model identification by your suggestion, and a lot of improvements followed from there.

Akhtar Khan of RIT, for a useful discussion about optimization that helped me proceed with confidence when implementing routines for numerical differentiation, and of course for teaching me some serious numerical analysis.

Zi-Jia Gong of RIT, for your contributions during the early stages of this work, for continuing to provide helpful commentary, including a discussion that helped refine my differentiation procedure, and for being a mentor and friend. Let's keep studying and collaborating together.

Nishant Malik of RIT, for advising me through two years of research, for introducing me to your field of study, for supporting me unyieldingly, for being a friend, and for turning me into a scientist. It might be overconfidence, but I think I can go from here and produce fantastic, impactful science. Let's continue our discussions and continue to come up with better and better questions.

# Contents

# 1. Introduction

Consider a network with each vertex labeled either $S$ or $I$, like the example in Figure 1. This network is the base component of the mathematical model that this thesis examines.

In this model, the vertices represent people, and their labels indicate infection with a disease. The vertices labeled $I$ represent people who are infected with the disease and capable of spreading the disease to others. We call these vertices *infectious nodes*. The vertices labeled $S$ represent people who are not infected with the disease but susceptible to becoming infected. We call these vertices *susceptible nodes*.

The edges represent connections between people. If two nodes are connected, the disease can

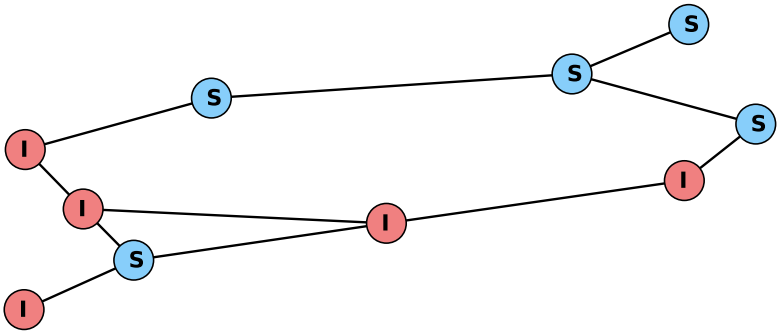


**Figure 1:** An example graph with nodes labeled $I$ or $S$.

spread between the two nodes. An infectious node on one side of an edge can infect a susceptible node on the other side of the edge.

As time progresses, the network changes in three ways. First, the infectious nodes revert to susceptible at some rate $r$, representing recovery from the disease. Second, infectious nodes infect their susceptible neighbors at some rate $p$. Third, susceptible nodes "rewire" their connections to infectious nodes with connections to other susceptible nodes, at some rate $w$ (See Figure 2). To emphasize, both the states of the nodes and the *structure* of the graph change with time. We call such models in which both the states of the nodes and the structure of the network change with time *coevolving networks*, and the literature also refers to them as *adaptive networks* [2].

We call the above model the *coevolving network SIS model*. Gross et al. develop this model in their 2006 study [1] and present a system of ordinary differential equations (ODEs) to approximate the density of infection over time. Equation 1.1 describes this *pair approximation*, in which $i$ and $s$ represent respectively the fractions of infected and susceptible nodes at a given time. $l_{II}$, $l_{SS}$, and $l_{SI}$ refer to the proportions of edges connecting infectious nodes to infectious nodes, susceptible nodes to susceptible nodes, and susceptible nodes to infectious nodes, respectively.

$$\begin{aligned} \frac{d}{dt}i &= pl_{SI} - ri \\ \frac{d}{dt}l_{II} &= pl_{SI}(\frac{l_{SI}}{s} + 1) - 2rl_{II} \\ \frac{d}{dt}l_{SS} &= (r+w)l_{SI} - \frac{2pl_{SI}l_{SS}}{s} \end{aligned} \tag{1.1}$$

The coevolving network SIS model illustrates the value of coevolving networks to mathematical

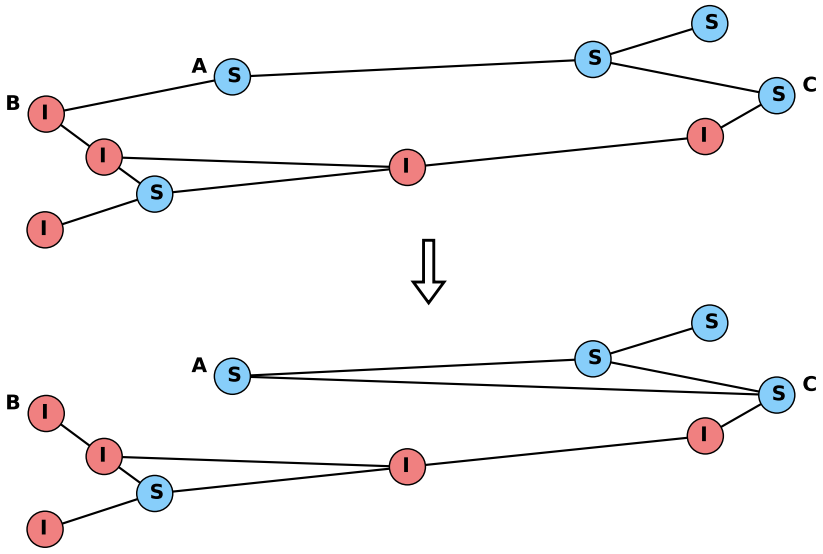


**Figure 2:** An example of rewiring. On the top, the susceptible node $A$ shares an edge with with infectious node $B$. On the bottom, node $A$ replaces its connection to node $B$ with a connection to the susceptible node $C$.

modeling. The network setting for disease spread reflects that a given person realistically only communicates the disease with a small subset of the population. People in California are unlikely to *directly* infect people in New York. Rewiring reflects the tendency for people to avoid becoming infected. If you know that one of your friends is sick, you might avoid that friend for a while to avoid becoming infected yourself.

Gross et al. (2006) is one of a cohort of publications describing coevolving network models and ODE systems to approximate their temporal evolution. Chapter 2 introduces a few of these coevolving network models and demonstrates how approximations like Equation 1.1 require intense mathematical modeling to produce. More accurate approximations than 1.1 and its analogues tend to be much larger equation systems, which are more difficult to produce and analyze. Moreover, approximations are scarce for more complicated coevolving network models.

In the case of the coevolving network SIS model, we want to understand the effect of the parameters $p$, $r$, and $w$, as well as the initial configuration of the network, on the prevalence of the disease.

A system of equations describing the progression of the disease in terms of these parameters improves understanding of the underlying system by allowing for analysis without the need to perform additional simulations.

The topic of this thesis is constructing equations like 1.1 using data. We simulate the coevolving network programmatically and record information about the system at discrete time points. Numerical simulations proceed in discrete time, and the parameters $p$, $w$, and $r$ are probabilities of infection and rewiring along each I-S link, and recovery of each infectious node. With time series from a large sample of parameter combinations and initial conditions, we present an approach to automate the construction of ODE systems to describe coevolving networks. The approach fits into the category of methods called *sparse model identification*, which use regression and subsequent truncation of small coefficients to produce symbolic equations.

In addition to developing the analysis of coevolving networks, this thesis is also a case study in sparse model identification. Chapter 3 explains the methods we employ and demarcates our specific modifications to the basic approach to sparse model identification. While the focus of this thesis is application to coevolving network systems, the approach we present generalizes to other systems.

Chapter 2 of this thesis introduces coevolving networks, including a review of previous efforts to model the macroscopic dynamics of coevolving networks using ODE models and a review of methods to construct differential equations from time series data.

Chapter 3 describes our approach to sparse model identification and its application to the problem outlined above. Chapter 4 demonstrates the performance of our solution in the context of the coevolving SIS system. Chapter 4 concludes with a discussion of the results and suggests areas for improvement and further investigation.

The essential three contributions of the thesis are: a method of numerical differentiation suited to the task of sparse model identification, a formulation of sparse model identification that takes advantage of conserved quantities in coevolving networks, and data-driven systems of equations for the coevolving SIS system.

# 2. Background

## 2.1 Coevolving Networks

Introduced in Chapter 1, the coevolving SIS model of epidemic spread illustrates how networks model large systems with interacting components. Epidemics are one example of a *networked dynamical system*; each node in the network takes a value in some set that can change with time. The change of a node is influenced by the values of its neighbors. Moving forward we call the value of a node its *state*. Some models just consider temporal changes in node states. The Glauber dynamics for the Ising model is a famous example from physics, and some of the examples to

follow, such as the SIS model, the voter model, and others have been studied on static, unchanging networks [10].

In addition to the temporal evolution of node states, coevolving networks also incorporate the temporal evolution of the network structure. In other words, the pattern of connection between nodes evolves in time. We sometimes refer to this evolution as a change in the network's *topology*.

Because the dynamics of a node's state depends on the influence of adjacent nodes, change in topology affects change in state. As we see in the below examples, changes in topology are influenced by node states. It is because of this two-way interaction that such systems are called *co*evolving, or adaptive, networks. The review paper by Gross et al. [2] provides an overview of coevolving networks and furnishes a collection of examples. The remainder of this section describes a selection of examples and their interesting properties.

### 2.1.1 The Coevolving Voter Model

Having introduced the coevolving SIS system in Chapter 1, the next example system takes us from epidemics to a new application, the diffusion of opinions in a society. Just like our exposure to diseases, our exposure to ideologies depends on the people with whom we interact. Likewise, such ideologies steer us towards some people and away from others. Introduced by Holme et al. [5] and modified by Durrett et al.[6] (described here), the coevolving voter model formulates this interaction as a process on a social network. The nodes in the network all take one of two values at a given time, either opinion 0 or opinion 1. We call an edge connecting two nodes of different opinions a *discordant edge* [6]. Proceeding in discrete time, each step of the simulation chooses randomly one discordant edge and one of its incident nodes (node $a$, connected to node $b$). With probability $\alpha$, node $a$ rewires to a new, random node holding the same opinion as itself. With probability $1-\alpha$, node $a$ changes its opinion to match node $b$. The simulation can proceed until there are no discordant edges. See figure 3 for an illustration of two possible outcomes: a network dominated by one opinion, or a split into disconnected subnetworks dominated by different opinions. The model considers our tendency to 1) replicate the opinions of our friends and 2) choose friends with similar opinions.

### 2.1.2 Evolutionary Game Theory with Coevolving Networks

Evolutionary game theory investigates how prevailing strategies emerge in populations of individuals making decisions regarding their interaction with other individuals, where the individuals

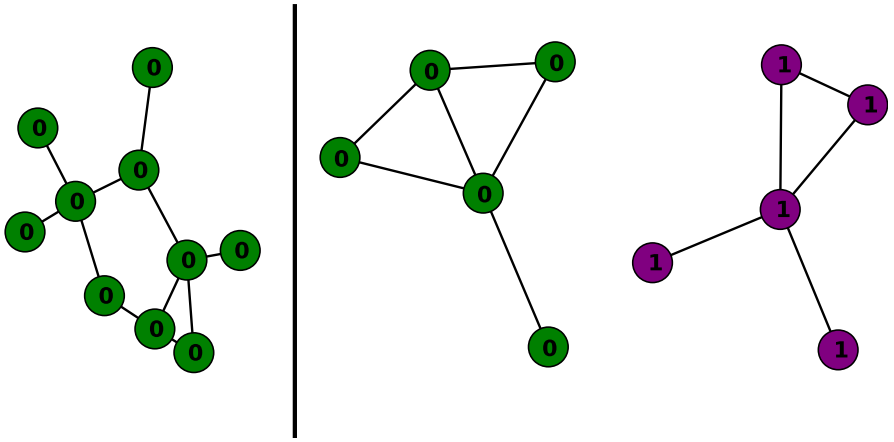


**Figure 3:** Two possible outcomes for the coevolving voter model [6]. Left, the simulation ends, and opinion 0 dominates. Right, the simulation ends with the graph split in two, with each connected component dominated by one opinion.

are referred to as *players*. In standard game theory, players make what is determined to be the "best" decision with the assumption that the adjacent player will also make its own "best" decision. Evolutionary game theory instead affords players flexibility in their decision making. Rather than maximizing their utility in an isolated 1-on-1 game, players strive to succeed in the dynamic ecosystem of other players. For a review of evolutionary game theory see [7].

Fu et al. [8] introduce a model of evolutionary game theory in the context of coevolving networks. In the model they present, each node in the network is a player, and each edge indicates a game played by its two incident nodes. The game of choice in this case is called the prisoner's dilemma game [9], in which both players choose to either *cooperate* or *defect*. The *payoff matrix* gives the outcome for a given player:

| | $C$ | $D$ |
|---|---|---|
| $C$ | 1 | 0 |
| $D$ | $1+u$ | $u$ |

The row indicates the decision of player 1, the column indicates the decision of player 2, and the entry indicates the payoff received by player 1. The constant parameter $u$ falls in the interval $(0, 1)$. If both players cooperate, then both players receive a payoff of 1. If player 1 defects and player 2 cooperates, player 1 receives a payoff of $1 + u$, and player 2 receives a payoff of 0. If both players defect, then both players receive a payoff of $u$. A larger value of $u$ makes defecting more tempting

(consider the limits in which $u$ approaches either 0 or 1).

The state of each node and its inclination to rewire is just a little more complicated than in previous examples. At a given time, every node is labeled either $C$, for cooperator, or $D$, for defector. A node labeled $C$ cooperates in all of the games along its edges, and a node labeled $D$ defects in all of the games along its edges. Like in the voter model, each discrete time step selects an edge connecting nodes of differing type: a $C$ node and a $D$ node. With probability $w$, one of the nodes changes its strategy. To decide which node changes, the total payoff for each node is calculated by summing the payoffs from the games along all of its edges. We can call the total payoff for the $C$ node $P_C$ and likewise $P_D$ for the $D$ node. The probability that the $C$ node becomes a $D$ node is then

$$\phi_{C\to D} = \frac{1}{1+e^{\beta(P_C - P_D)}},$$

where $\beta$ is a constant parameter in the interval $[0,\infty)$. Otherwise the $D$ node becomes a $C$ node. The node with the lower total payoff is more likely to change. Setting the value of $\beta$ at its endpoints leads to two simplified models. If $\beta = 0$, either node will change with equal probability, regardless of total payoff. In the limit of large $\beta$, the node with lower payoff automatically changes.

With probability $1-w$, the node labeled $C$ rewires to a different node, chosen uniformly at random from all nodes in the network, with the exception of immediate neighbors.

Numerical simulations show that with little to no rewiring, all nodes eventually become defectors. Introducing a sufficient level of rewiring, however, allows cooperation to persist in a portion of the population. For a further description, detailing abrupt dynamical transitions and extensions of the model, see the referred publication [8].

#### 2.1.3 Reinforcing Transitivity

Transitivity is a property observed particularly in social networks, in which neighbors of a node tend to be neighbors with each other. Equivalently, transitivity quantifies the prevalence of triangles in a network, as illustrated in Figure 4. For more information on transitivity (and complex networks in general), see the textbook [4] or the review [3] by Newman. One of a few alternate definitions for a network's transitivity found therein is

$$C = \frac{3\times(\text{total number of triangles})}{\text{number of connected triples}}.$$

A connected triple is a connected component consisting of three nodes and two edges (Figure 4). The number 3 appears in the numerator because a triangle consists of three connected triples. The expression thus gives the fraction of connected triples in the network that are closed off as triangles. The value of $C$ ranges between 0 and 1, with a higher value indicating higher transitivity and higher prevalence of triangles. The symbol $C$ refers to another name for this quantity, the "clustering coefficient."

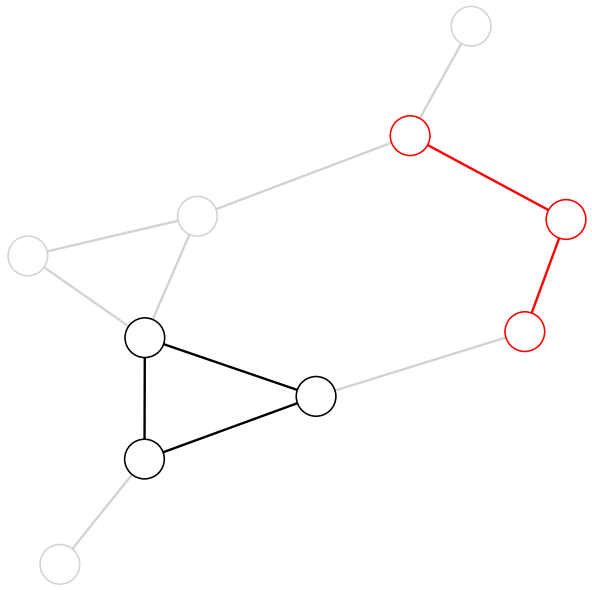

**Figure 4:** A triangle (black) and a connected triple (red) within a larger network. Transitivity quantifies the prevalence of these triangles in a network. Can you spot the other triangle?

Of interest to us is the impact of transitivity on dynamical processes. If the nodes are clumped together with lots of triangles, what is the impact on spreading processes and other dynamical systems? This question is addressed in the context of coevolving networks by adjusting some of the system rules with attention to transitivity. Malik et al. [12] observe that coevolving network models tend to remove triangles over time, since when nodes rewire, they find a node with the same state without regard for its location in the network. This tendency makes the networks less resemblant of actual social networks. Motivated by this observation, they propose a modification of the coevolving SIS model that encourages transitivity.

Recall that at rate $w$, susceptible nodes remove links with infectious nodes and rewire to a new susceptible node, choosing randomly from the members of the population that are not already

neighbors. In the modified model, upon the occurence of a rewiring, with probability $\eta$ the susceptible node rewires to a susceptible *neighbor of a neighbor*. The result is the formation of a new triangle. Otherwise, with probability $1-\eta$, the susceptible node rewires to any random susceptible node from the population. This new parameter $\eta \in (0,1)$ adjusts the amount of transitivity reinforcement. Computational experiments indicate that higher $\eta$ leads to fewer infections over the long term. This result tentatively suggests that actual societies can slow infectious disease outbreaks by closing their triangles. Related studies reinforce transitivity in the coevolving voter model as well [11, 12, 13].

### 2.1.4 ODE Approximations for Coevolving Networks

Having become aquainted with a few examples of coevolving networks and their modeling scope, we turn our attention to some of the tools to investigate them. The first and foremost tool is, surely, computational simulation. All of the models are stated in terms of rules to update the system in discrete time steps. These rules translate to computer code for stochastic simulations. Simulations can reveal the effects of varying parameter values and initial conditions on the trajectories and long-term evolution of coarse variables, such as the fraction of infectious nodes in the SIS model. The alternate approach in common use is to construct systems of ordinary differential equations that approximate the temporal evolution of coarse variables, or summary statistics, describing the state of the network system. This system of equations serves as an alternate picture of the coevolving network system.

Why make equations? After all, these equations are only approximations, and their predictions can deviate from the results of computational simulations. For one thing, the equations can usually be numerically integrated to obtain trajectories and long-term behavior much more quickly than the computational simulations. When investigating the behavior as (potentially several) parameters vary, this computational advantage adds up. Furthermore, equations contain information on their own, codifying the effect of each variable and parameter on the rate of change of each variable. The computational simulations and the ODE systems are both mathematical models; both describe the interaction of state and topology, and both have limitations in their resemblance to the real world. When viewed as a modeling framework, coevolving networks ought to be actualized in the way that best serves an application. As coevolving networks become more popular as modeling tools, perhaps they will find use in settings that necessitate quick, on-the-fly evaluation, where equations are more appropriate.

In this thesis, the type of ODE system that we are most interested in is called a *pair approximation*.

Let us look again at the pair approximation (Equation 1.1) for the coevolving SIS system, introduced in [1]. This time we dispense with the parentheses in order to look at each term separately.

$$\begin{aligned}\frac{d}{dt}i &= pl_{SI} - ri \\ \frac{d}{dt}l_{II} &= pl_{SI} - 2rl_{II} + \frac{pl_{SI}l_{SI}}{s} \\ \frac{d}{dt}l_{SS} &= rl_{SI} + wl_{SI} - \frac{2pl_{SI}l_{SS}}{s}\end{aligned}$$

The variable $i$ is the fraction of nodes that are infectious at a given time. Because the number of nodes does not change with time, the variable $s$, the fraction of susceptible nodes, is $1 - i$. The variable $l_{II}$ is defined as the number of edges connecting two infectious nodes divided by the number of nodes in the network. $l_{SS}$ is defined analogously. Because the number of edges does not change with time, $l_{SI}$ is $M/N - l_{II} - l_{SS}$. $M$ is the number of edges, and $N$ is the number of nodes.

In the first equation, the term $pl_{SI}$ accounts for new infections taking place along edges between susceptible and infectious nodes. At each time step, the infectious node infects the susceptible node with probability $p$, and the edge of type $SI$ becomes an edge of type $II$. Scaling with the number of $SI$ links in the network gives a good indication of new infections, but it does have a shortcoming. The term does not consider that some susceptible nodes are neighbors with more than one infectious node, so this first equation will consequently overestimate the occurence of new infections. The term $-ri$ accounts for recovery reducing the number of infectious nodes. At each time step, an infectious node becomes a susceptible node with probability $r$.

In the second equation, the term $-2rl_{II}$ considers that when a node on either side of an edge of type $II$ recovers, the edge becomes an edge of type $SI$. The factor of 2 reflects that either node could cause this change. The term $pl_{SI}$ accounts for the formation of new edges of type $II$ when an infection occurs over an edge of type $SI$. This term is inexact for the same reason it is inexact in the first equation. When the susceptible node becomes infectious, it could create more than one edge of type $II$, depending on who its other neighbors are. This imbalance is the reason for the third term, which is less transparent. The term $\frac{pl_{SI}l_{SI}}{s}$ is an approximation of another term that considers connected triples [14], which can be written as $pt_{ISI}$. See Figure 5 for an illustration of a connected triple. This new notation $t_{ISI}$ is the number of connected triples with a susceptible node between two infectious nodes, divided by the number of nodes in the network. Because variables representing connected triples like $t_{ISI}$ are not part of the system of equations, they need to be approximated in terms of the variables that are in the system of equations. For this reason

Gross et al. [1] use the *moment closure approximation* of [15]

$$t_{abc} \approx l_{ab} \frac{l_{bc}}{b}. \tag{2.1}$$

The approximation starts with the prevalence of edges of type $ab$. If the node of type $b$ at the end of this $ab$ edge is connected to a node of type $c$, then we have a triple of type $abc$. If we select an arbitrary node of type $b$ from the network, we can expect it to be connected to $\frac{l_{bc}}{s}$ nodes of type $c$, on average. Thus the product $l_{ab}\frac{l_{bc}}{b}$ approximates $t_{abc}$.

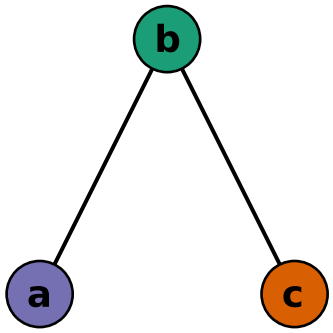


**Figure 5:** A connected triple of type $abc$. The moment closure approximation [1, 15] $t_{abc} \approx l_{ab}\frac{l_{bc}}{b}$ allows for connected triples to appear in pair approximations.

Plugging in this approximation, $pt_{ISI}$ becomes $\frac{pl_{SI}l_{SI}}{s}$. The terms in the third equation have similar motivation.

Similar systems of equations have been derived for other models, including dynamics on static networks [10] and other coevolving networks including the coevolving voter model [6] and the prisoner's dilemma system [16]. Demirel et al. [17] present and juxtapose strategies to construct pair approximations and their moment closures.

In Chapter 4 we compare the performance of this pair approximation with computational simulations and the data-driven equations developed in this thesis. For now, however, we can see that this pair approximation has a few potential sources for error, and they result from ignoring intricacies of the network structure. Using an approximation to represent connected triples in terms of

edge types and node types suggests the possibility of including such higher-order subnetworks as variables in the system of equations. The most popular approach to do so is the method of *approximate master equations*.

Marceau et al. (2010) [14] develop approximate master equations for the coevolving SIS system. The variables used in the equations are $S_{km}$ and $I_{km}$ for $0 \leq m \leq k$, and $0 \leq k \leq k_{\max}$, totaling $(k_{\max}+1)(k_{\max}+2)$ equations [10]. $S_{km}$ is the fraction of nodes in the network that are susceptible, have $k$ neighbors ("degree $k$", in the language of graph theory and network science), and have $m$ infectious neighbors. See figure 6 for an illustration. $I_{km}$ is the fraction of nodes in the network that are infectious, have $k$ neighbors, and have $m$ infectious neighbors. The approximation is cut off at some positive integer $k_{\max}$ which the degree of the vast majority of nodes is expected to stay below.

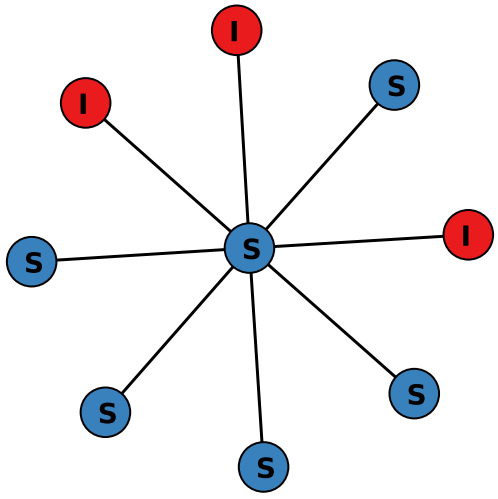


**Figure 6:** A node of type $S_{km}$, $k = 8$, $m = 3$ (center).

The system of equations includes a term for each of the ways in which one type of node can become another type of node, resulting from infection, recovery, or rewiring. The equations in [14] take the form

$$\frac{dS_{km}}{dt} = rI_{km} - pmS_{km} + r[(m+1)S_{k(m+1)} - mS_{km}] + p\frac{S_{SI}}{S_S}[(k-m+1)S_{k(m-1)} - (k-m)S_{km}]$$
$$+w[(m+1)S_{k(m+1)} - mS_{km}] + w\frac{S_I}{S}[S_{(k-1)m} - S_{km}]$$
$$\frac{dI_{km}}{dt} = -rI_{km} + pmS_{km} + r[(m+1)I_{k(m+1)} - mI_{km}] + p(1+\frac{S_{II}}{S_I})[(k-m+1)I_{k(m-1)} - (k-m)I_{km}]$$
$$+w[(k-m+1)I_{(k+1)m} - (k-m)I_{km}].$$

A derivation of these equations is beyond the scope of this review; even some of the notation is left unexplained here. Regardless, we can see the jump in complexity from the pair approximation, both in the number of equations and the complexity of the equations themselves. The reward is an increase in accuracy resulting from more detailed modeling of the coevolving network system. Similar systems of equations have been derived for dynamics on static networks [10], the coevolving voter model [6], the prisoner's dilemma system [16], and transitivity-reinforced systems [13]. While the objective of this thesis is to create pair approximations from data, a project for the future could be to do the same for approximate master equations.

## 2.2 Sparse Model Identification

Sparse model identification is an approach to use time series data to create a mathematical model in the form of a system of differential equations. Sparse model identification is the basic procedure through which we create pair approximations for the coevolving SIS system. Wang et al. introduced sparse model identification in [31].

Given an n-dimensional time series $\mathbf{x}(t_i)$, the objective is to obtain a system of differential equations

$$\begin{aligned}\frac{dx_1}{dt} &= f_1(\mathbf{x})\\ \frac{dx_2}{dt} &= f_2(\mathbf{x})\\ &\vdots\\ \frac{dx_n}{dt} &= f_n(\mathbf{x})\end{aligned}$$

corresponding with $\mathbf{x}(t_i)$ and hopefully corresponding accurately with other time series generated by the process that generates $\mathbf{x}(t_i)$.

Using a procedure for numerical differentiation, produce an accompanying time series $\frac{d}{dt}\mathbf{x}(t_i)$ that estimates the derivative of $\mathbf{x}(t_i)$. We discuss such procedures in Chapter 3.

Sparse model identification produces equations of the form

$$f_i = C_1 g_1 + \cdots + C_s g_s,$$

where each function $f_i$ is a linear combination of some collection, or library [19], of functions $g_1 \ldots g_s$. The functions in this library are chosen by the investigator. The coefficients $C_1 \ldots C_s$ result from a regression procedure, fitting the numerical derivatives $\frac{d}{dt}\mathbf{x}(t_i)$ with the library functions evaluated on the time series $\mathbf{x}(t_i)$. Using a regression that induces some of the coefficients $C_j$ to be zero, such as LASSO [44] or sequential threshold least squares [19], reduces the number of terms in the equations to a desired level. By not just scaling but also selecting the terms of the equation, sparse model identification attempts to reveal the fundamental interactions driving the system.

References [37] and [38] overview sparse model identification and discuss best practices. In Chapter 3 we describe the modified approach to sparse model identification used in this thesis. The approach described above has a number of extensions and modifications. Yang et al. investigate an extension to time-varying systems in [32]. Boninsegna et al. propose an extension to stochastic differential equations [35]. Zhang et al. [36] present convergence results for the formulation of [19]. Schaeffer et al. [34] avoid numerical differentiation by optimizing coefficients in the integral form of the equations to identify. Similarly, Messenger et al. [40] use the weak form of the equations in order to optimize through a Galerkin procedure. Mangan et al. [33] discuss hyperparameter tuning using information criteria. Kaheman et al. [39] use optimization via automatic differentiation to both identify a model and remove noise simultaneously.

It is worth noting that parameter estimation from data, including for differential equations, is an established subject in the field of inverse problems [28], and that sparse model identification is a form and application of parameter estimation. As a notable early example, Crutchfield et al. use data to make differential equations in the 1987 paper [29]. Kukreja et al. use LASSO to create sparse and interpretable NARMAX models, addressing a similar problem for difference equations rather than differential equations [30].

There is some previous work that resembles the application of this thesis. While this thesis presents results for global-scale dynamics (variables like $i$, $l_{II}$, and $l_{SS}$), Bakhtiarnia et al. [42] nest sparse model identification with a genetic algorithm to build equations for the micro-scale dynamics, at

the node-level, of networked dynamical systems from simulation data. Daniels et al. [41] use a procedure similar to sparse model identification to construct equations of global-scale dynamics from data generated through simulation of high-dimensional biochemical processes. Bramburger et al. [43] use sparse model identification to build equations for the slow-timescale dynamics of time series possessing dynamics occurring also on a fast, periodic timescale.

# 3. Methods

Similar to previous developments of sparse model identification (see Chapter 2), we produce our equations as linear combinations of basis functions, selected in a sparse regression of training trajectories and their numerically estimated derivatives. The method employed here offers several adjustments to suit the needs of the application to coevolving networks, including a new differentiation method that is well suited to the task of sparse model identification, and regression that leverages conserved quantities of coevolving networks (number of nodes and edges) to improve the model identification performance. The chapter ends with an example to demonstrate the utility of these modifications to model identification in general.

## 3.1 Data Collection

In general, the data collection process goes as follows. Obtain a computer simulation of a coevolving network system, in order to run a collection of simulations sampling the range spanned by each system parameter. At each step of the simulation, record the values of coarse variables to be described in the final equations. To make pair approximations, the coarse variables are the densities of each compartment and link type. In the case of the SIS model of [1], the densities of compartments are labeled $i$ and $s$. The variable $i$ is the ratio of infectious nodes to the total number of nodes, and $s$ is the same ratio but for susceptible nodes. Because the number of nodes in the network is conserved, only one of $i$ and $s$ needs to be recorded, and the other can be calculated afterward. The densities of link types are labeled $l_{II}$, $l_{SS}$, and $l_{SI}$. The variable $l_{II}$ is the number of links in the network connecting two infectious nodes, divided by the number of nodes. Because the number of links in the network is conserved, only two of the three edge types must be recorded during simulation. The parameters to the system, $p$, $r$, and $w$, are sampled from the range of values they can take. Another parameter that can be sampled is the mean degree, $< k >= \frac{2M}{N}$. The maximum number of links in a network with $N$ nodes is $\binom{N}{2}$, but because the simulations become increasingly computationally demanding with higher mean degree, the value of $< k >$ should be sampled in a predefined interval. Figure 7 plots a time series of measurements of $i$, $l_{II}$, and $l_{SS}$ collected from a simulation.

For the experiments and results shown in Chapter 4, we implement a numerical simulation of the coevolving SIS system with the C++ programming language. Each simulation accepts values for the number of nodes $N$ and the number of edges $M$ in the network, the system parameters $p$, $r$, and $w$, and an initial number of infectious nodes. For the initial network configuration, we sample an Erdős-Rényi random graph [18, 4] by successively placing the $M$ edges between randomly-selected pairs of non-adjacent nodes. Initial network configuration does affect the trajectories of the coarse variables [10, 11, 14] (in this case $i$, $l_{II}$, and $l_{SS}$), and it is possible to sample additional initial network configurations, but for simplicity we use the Erdős-Rényi model for all simulations. We also limit the variety of network configurations by using $N = 5,000$ nodes and $M = 5,000$, $M = 17,500$, and $M = 10,000$ edges (resulting in mean degree $< k >= 2$, $< k >= \frac{7}{2}$, and $< k >= 20$ respectively) for the simulations used in our three examples. Ideally we would create data-driven equations that apply to network systems with any mean degree, and we anticipate this simplification to be generalized in future work. The initial number of infectious nodes $N * i_0$ is sampled in the interval $(0, N)$, and every node has equal probability $i_0$ of being infectious at the beginning of the simulation.

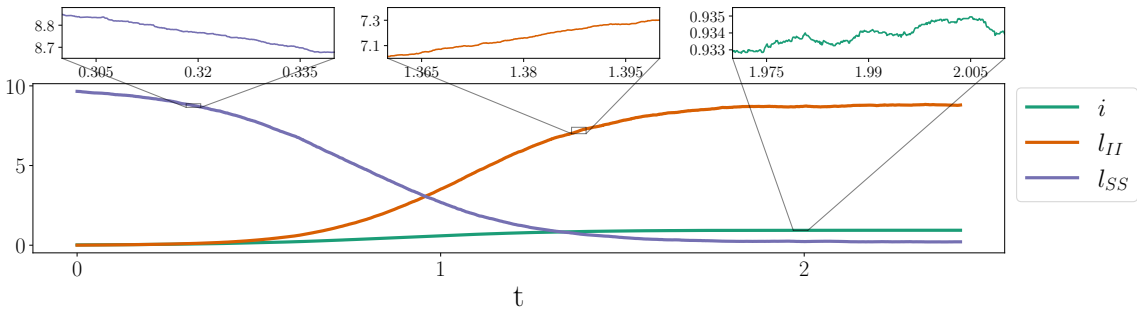


**Figure 7:** Example time series from the coevolving SIS system with a network of $N = 20,000$ nodes and $M = 200,000$ edges. Zooming in shows the stochastic influence on the system evolution, especially around equilibrium. $p = .000000037, r = .000000156, w = .0000509781$ (approximately), $i_0 = .01825$, $l_{II0} = .00295$, $l_{SS0} = 9.6544$. Each variable in the time series has 24,299 measurements.

The selection of system parameters $p$, $r$, and $w$ is a matter of objective. Scaling all of the parameters by the same factor amounts to slowing down or speeding up the process by that factor, and although there are three parameters, all of the behavior can be observed by varying just two of them [14]. All simulations use fixed $r$ and sample $p$ and $w$ across a grid. We choose parameter values to match the examples shown in [1] and [14]. Numerical simulations in these parameter ranges demonstrate behavior that we are interested in capturing with our data-driven equations, particularly the emergence of bistability in the system variables as $w$ increases [1, 14] (see Figure 21 in Chapter 4).

## 3.2 Numerical Differentiation

For each coarse variable, expressed as a vector $\mathbf{x}$, and a chosen odd whole number $k$, perform a $k$-point moving average of the form

$$y_i = \frac{1}{k} \sum_{j=i-\frac{k-1}{2}}^{i+\frac{k-1}{2}} x_j \tag{3.1}$$

to obtain a vector $\mathbf{y}$, assuming equal time intervals between each measurement. This optional first step reduces the local variation of the time series to reduce the footprint of the underlying system's stochastic component and to make the data more amenable to numerical differentiation.

Following the optional preprocessing with a moving average, the method proceeds with numerical differentiation of the resulting time series $\mathbf{y}$. One method recommended for sparse model identification in [19], is the total variation regularized derivative [20] which takes an optimization

approach that balances both smoothness and agreement with antidifferentiation. It encourages the derivative estimate $\hat{f}$ to be smooth by penalizing its total variation $\|\frac{d}{dt}f\|_1$ in the objective

$$\hat{f} = \underset{f}{\operatorname{argmin}} \left\{ \left\| \int_{t_0}^{t_{\text{final}}} (f)dt - y \right\|_2^2 + \gamma \left\| \frac{d}{dt}(f) \right\|_1 \right\}.$$

The use of an $L_1$ penalty allows the method to estimate derivatives that are discontinuous [20]. This feature is useful in such cases but is extraneous for many cases of sparse model identification, including the coevolving networks in which we are interested.

An appropriate alternative to differentiate noisy but intrinsically smooth time series is the method introduced in [21], which instead of using an $L_1$ penalty uses a penalty of the form $(\|f\|_2^2 + \|\frac{d}{dt}f\|_2^2)$ in its objective. The two methods form a useful duo, but the bias that their regularizations introduce can still compromise the model identification. Indeed, experiments with model identification for example ODE systems show that a simple difference quotient procedure can sometimes outperform regularized differentiation methods, as we investigate in Section 3.5 (see Figures 10 and 13).

The differentiation method presented here attempts to split the difference between the noise amplification of a difference quotient calculation and the bias of a regularized derivative. Like [20] and [21], this derivative is computed through an optimization procedure. Similar to [22] and resembling the hyperparameter tuning of [23], we formulate the optimization as a smoothing of the input time series, such that its derivative computed with a difference quotient is well behaved. We call this procedure the "denoising derivative." First consider the motivating preliminary formulation

$$\hat{\mathbf{y}} = \underset{\mathbf{z}}{\operatorname{argmin}} \left\{ \gamma \, \|\text{Quad}[\, \text{DQ}(\mathbf{z})\,] - \mathbf{z}\|_2^2 + \|\mathbf{y} - \mathbf{z}\|_2^2 \right\}. \tag{3.2}$$

The left term of the cost function requests that the derivative is consistent in antidifferentiation, so that $\frac{d}{dt}\hat{\mathbf{y}}$ becomes $\hat{\mathbf{y}}$ when integrated. The operator Quad refers to the application of a numerical quadrature such as the trapezoid rule, Simpson's rule, or similar, while DQ refers to the computation of a difference quotient such as

$$\frac{d}{dt}y_i \approx \frac{y_{i+1} - y_{i-1}}{2\delta}.$$

The second term of the cost function in Equation 3.2 prevents the estimate $\hat{\mathbf{y}}$ from straying too far from $\mathbf{y}$, with the adjustable parameter $\alpha$ scaling the term's influence. As $\alpha \to 0^+$, the denoising derivative reduces to the difference quotient calculation. Larger $\alpha$ values prioritize the smoothness of the estimate. In this way the denoising derivative bridges the gap between the noise-amplifying but unbiased difference quotient and the noise-robust but biased regularized differentiation methods, filling a niche for the problem of sparse model identification and its need for both accuracy and stability in the ultimate model.

Equation 3.2 is less than ideal for actual implementation because the composition of numerical quadrature and numerical differentiation introduces an artifact near the endpoints of the time interval. Specifically, the composition amounts to a moving average with extraneous effects near the endpoints of the time series (see Figure 8 for an explanation and Figure 9 for an example), so for this reason we proceed with a modified version of Equation 3.2. The replacement, Equation 3.3, directly invokes a moving average to avoid the introduction of an artifact. Discerning readers may notice that Equations 3.2 and 3.3 are not entirely equivalent, as Equation 3.3 optimizes through a moving average of $\mathrm{DQ}(\mathbf{z})$, not $z$. This change takes the opportunity to smooth the difference quotient, which has more local variability than the original time series.

$$\hat{\mathbf{y}} = \underset{\mathbf{z}}{\operatorname{argmin}} \left\{ \gamma \left\| \mathrm{MA}[\,\mathrm{DQ}(\mathbf{z})\,] - \mathrm{DQ}(\mathbf{z}) \right\|_2^2 + \left\| \mathbf{y} - \mathbf{z} \right\|_2^2 \right\} \tag{3.3}$$

Figure 10 in Section 3.5 compares the performance of the denoising derivative with the $L_1$ procedure of [20] and the $L_2$ procedure of [21].

To implement the denoising derivative we use the convex optimization library CVXPY [24, 25], an interface for the Python programming language. Convex optimization also allows for the use of constraints, which may be useful depending on the problem. In extending this work to the voter model, for example, a constraint to the denoising derivative could require $\hat{\mathbf{y}}$ to equal $\mathbf{y}$ at the endpoints, since in some parameter settings the ending values of the coarse variables depend continuously on the initial values [6], and their exact values ought to be maintained for maximum accuracy in the final model.

## 3.3 Selecting Basis Functions

Following numerical differentiation, the next step of sparse model identification is establishing a collection of basis functions, sometimes called a library [19], for the ODE system. Depending on the application for model identification, understanding of the system mechanics could guide

Given a time series $\mathbf{y}$, the application of a difference quotient $\frac{d}{dt}y_i \approx \frac{y_{i+1}-y_{i-1}}{2\delta}$ can be expressed as a matrix multiplication of $\mathbf{y}$:

$$\mathrm{DQ}(\mathbf{y}) = \frac{1}{2\delta}\begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix}$$

Likewise, numerical quadrature with the trapezoid rule can be expressed with the matrix multiplication of a vector $\mathbf{y\prime}$:

$$\mathrm{Quad}(\mathbf{y\prime}) = \frac{\delta}{2}\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 & 0 & 0 \\ 1 & 2 & 2 & 2 & 1 & 0 \\ 1 & 2 & 2 & 2 & 2 & 1 \end{pmatrix}\begin{pmatrix} y_1\prime \\ y_2\prime \\ y_3\prime \\ y_4\prime \\ y_5\prime \\ y_6\prime \end{pmatrix}$$

The composition of quadrature and differentiation is the multiplication of these two matrices:

$$\mathrm{Quad}(\mathrm{DQ}(\mathbf{y})) = \frac{1}{4}\begin{pmatrix} -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -2 & 0 & 2 & 1 & 0 & 0 & 0 \\ -1 & -2 & -1 & 1 & 2 & 1 & 0 & 0 \\ -1 & -2 & -1 & 0 & 1 & 2 & 1 & 0 \\ -1 & -2 & -1 & 0 & 0 & 1 & 2 & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix}$$

which resembles a moving average but shifted and with extraneous terms in the first three columns. These artifacts resulting from composing two numerical procedures lead to poor estimation of the derivative near the endpoints of the time series. For this reason we directly invoke a moving average in the loss function of the denoising derivative.

**Figure 8:** Why we replace Equation 3.2 with Equation 3.3 to implement the denoising derivative.

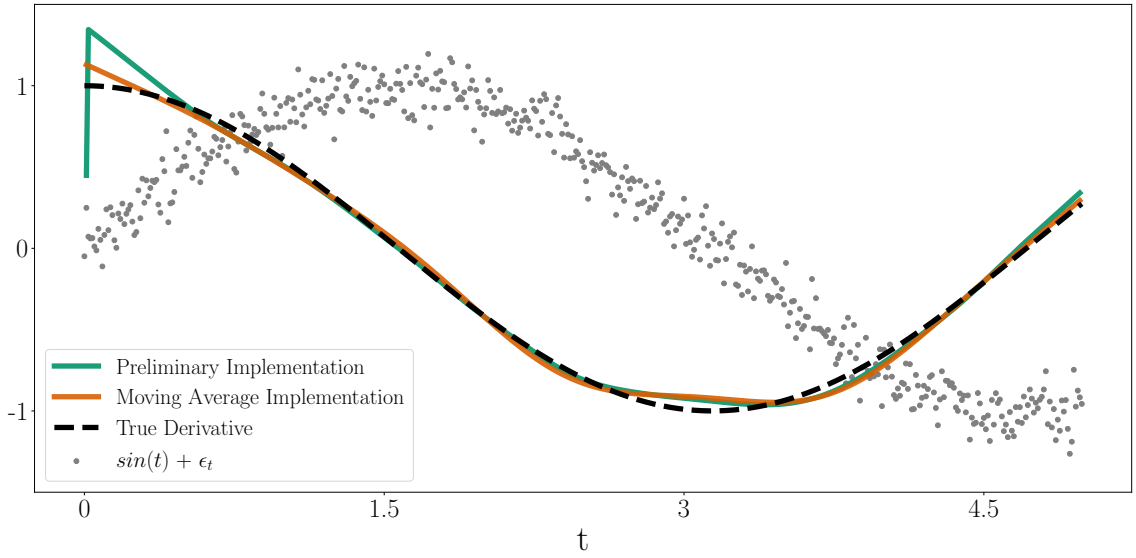


**Figure 9:** Comparison of the preliminary composition-based implementation of the denoising derivative with the moving average-based implementation. Notice how the preliminary implementation strays further at the endpoints.

the choice of basis functions, or alternatively, it can be useful to have a default library that makes no assumptions. In the examples to follow we use multiplications of the system variables and parameters, as advocated in [31]. We also take advantage of a property mentioned in Section 3.1, which is the speeding-up-or-down effect of multiplying all of the system variables by a constant. Look again at the analytically-derived pair approximation from [1]:

$$\begin{aligned}
\frac{d}{dt}i &= pl_{SI} - ri \\
\frac{d}{dt}l_{II} &= pl_{SI}(\frac{l_{SI}}{s} + 1) - 2rl_{II} \\
\frac{d}{dt}l_{SS} &= (r + w)l_{SI} - \frac{2pl_{SI}l_{SS}}{s}.
\end{aligned}$$

You may notice a commonality; the terms are all multiples of exactly one system parameter. This way, if the parameters are all multiplied by a positive factor $a$, the derivatives are all multiplied by $a$, and the system moves $a$ times as fast. We would like to maintain this property in our data-driven model, so we correspondingly make all of our library functions multiples of exactly one system parameter.

Sparse model identification proceeds by winnowing the library to suggest a sparse subset of its basis functions for each dynamic variable, eliminating functions whose coefficients are small in

the regression.

## 3.4 Choosing Coefficients

With the basis functions that remain, the final model identification follows from a least-squares regression, matching function evaluations to numerical derivatives. In the application to coevolving networks, known conserved quantities augment the regression. Namely, in the coevolving SIS example, the number of nodes is constant, and the number of edges is constant, so $\frac{d}{dt}i + \frac{d}{dt}s = 0$, and $\frac{d}{dt}l_{II} + \frac{d}{dt}l_{SS} + \frac{d}{dt}l_{SI} = 0$. These conservations are constraints on the regression, and it works as follows.

The identified model is an $n$-dimensional ordinary differential equation,

$$\frac{d}{dt}\mathbf{x} = \hat{\mathbf{f}}(\mathbf{x})$$

with each component $\hat{f}_1 \dots \hat{f}_n$ a linear combination of its library.

$$\hat{f}_i = \hat{C}_1 g_1 + \cdots + \hat{C}_{s_i} g_{s_i}$$

Suppose the model should hold the conservation $\hat{f}_1 + \cdots + \hat{f}_n = 0$. For a time series $y_1 \dots y_N$ and derivatives $\frac{d}{dt}y_i \dots \frac{d}{dt}y_N$, the estimated coefficients $\hat{C}$ emerge from an optimization of the form

$$\hat{C} = \underset{C}{\operatorname{argmin}} \left\{ \sum_{j=1}^{N} \left\| \frac{d}{dt}y_j - f(y_i) \right\|_2^2 + \alpha \sum_{j=1}^{N} (f_1(y_j) + \cdots + f_n(y_j))^2 + \lambda_1 \|C\|_1 + \lambda_2 \|C\|_2^2 \right\}. \quad (3.4)$$

The parameters $\lambda_1$ and $\lambda_2$ scale optional $L_1$ and $L_2$ penalties to improve conditioning and encourage sparsity in $\hat{C}$ [50]. Experiments with the coevolving SIS system indicate that using a combination of $L_1$ and $L_2$ penalties leads to the best performance. The parameter $\alpha$ scales the conservation term, though its value appears to have little effect on the outcome. As with differentiation, this optimization is obtained numerically through convex optimization, implemented with the library CVXPY [24, 25]. As introduced for model identification in [19], the method uses a sequential thresholding procedure, iteratively removing terms with coefficients smaller in absolute value than a specified value until no more terms are removed.

## 3.5 Example: Model Identification of a Classical SIR System

As an example to illustrate the method presented here, not in the context of networks, consider the system of equations 3.5 for the SIR epidemiological model

$$\frac{d}{dt}S = -\beta IS \quad , \quad \frac{d}{dt}I = \beta IS - \epsilon I \quad , \quad \frac{d}{dt}R = \epsilon I \tag{3.5}$$

with $\beta = 2$ and $\gamma = 1$ [26]. It is helpful to begin with a system of equations like this one, whose true form is already known, before moving on to model identification with experimental data. Equation 3.5 possesses the conserved quantity $\frac{dS}{dt} + \frac{dI}{dt} + \frac{dR}{dt} = 0$, similar to the conservations present in many coevolving network systems.

First we use this example system to compare the model identification performance of the three regularized differentiation methods introduced in Section 3.2, namely the denoising derivative introduced in this thesis, the derivative with Tikhonov penalty $\gamma(\|f\|_2^2 + \|\frac{d}{dt}f\|_2^2)$ [21], and the total variation derivative with penalty $\gamma\|\frac{d}{dt}f\|_1$ [20]. For the total variation derivative we use an implementation from the `derivative` library for Python [27], with setting `order=2`. For training data we integrate Equation 3.5 using the Python function `scipy.integrate.solve_ivp` [46] using solver BDF starting from three initial conditions: $(S = .9,\ I = .1,\ R = 0)$, $(S = .1,\ I = .9,\ R = 0)$, and $(S = .4,\ I = 0,\ R = .6)$. All three trajectories take 1000 evenly spaced measurements on the interval $(0 \leq t \leq 15)$, and to replicate experimental data, independent, normally distributed noise with mean 0 and standard deviation .001 is added to each measurement in all three dimensions.

In this example we skip the initial moving average treatment and proceed directly to numerical differentiation. In addition, the denoising derivative provides a smooth version of the time series, which we will use in our experiments with the coevolving SIS system, but for the sake of fair comparison we use the original time series for model identification in this example. For all three methods of numerical differentiation, we let their regularization parameter take the values $\gamma = 10^0$, $\gamma = 10^{-1}, \ldots, \gamma = 10^{-8}$, $\gamma = 10^{-9}$ and compare model identification performance as $\gamma$ varies. The denoising derivative uses an equally weighted 3-point moving average.

All equations in the identified systems are linear combinations of the library $\{S, I, R, SS, II, RR, SI, SR, IR\}$. The coefficients result from optimizing the special case of Equation 3.4

$$\hat{C} = \underset{C}{\operatorname{argmin}} \left\{ \sum_{j=1}^{N} \left\| \frac{d}{dt}y_j - f(y_i) \right\|_2^2 + \alpha \sum_{j=1}^{N} (f_1(y_j) + f_2(y_j) + f_3(y_j))^2 + \lambda \|C\|_1 \right\} \tag{3.6}$$

with $\alpha = 1$ and $\lambda = 10^{-3}$, with the term $\alpha \sum_{j=1}^{N} (f_1(y_j) + f_2(y_j) + f_3(y_j))^2$ acknowledging the conserved quantity $\frac{dS}{dt} + \frac{dI}{dt} + \frac{dR}{dt} = 0$. In this first example we refrain from sequential thresholding and simply choose coefficients once, keeping all of them regardless of their magnitude. For simplicity we use only the $L_1$ penalty in coefficient selection, for every example in this section.

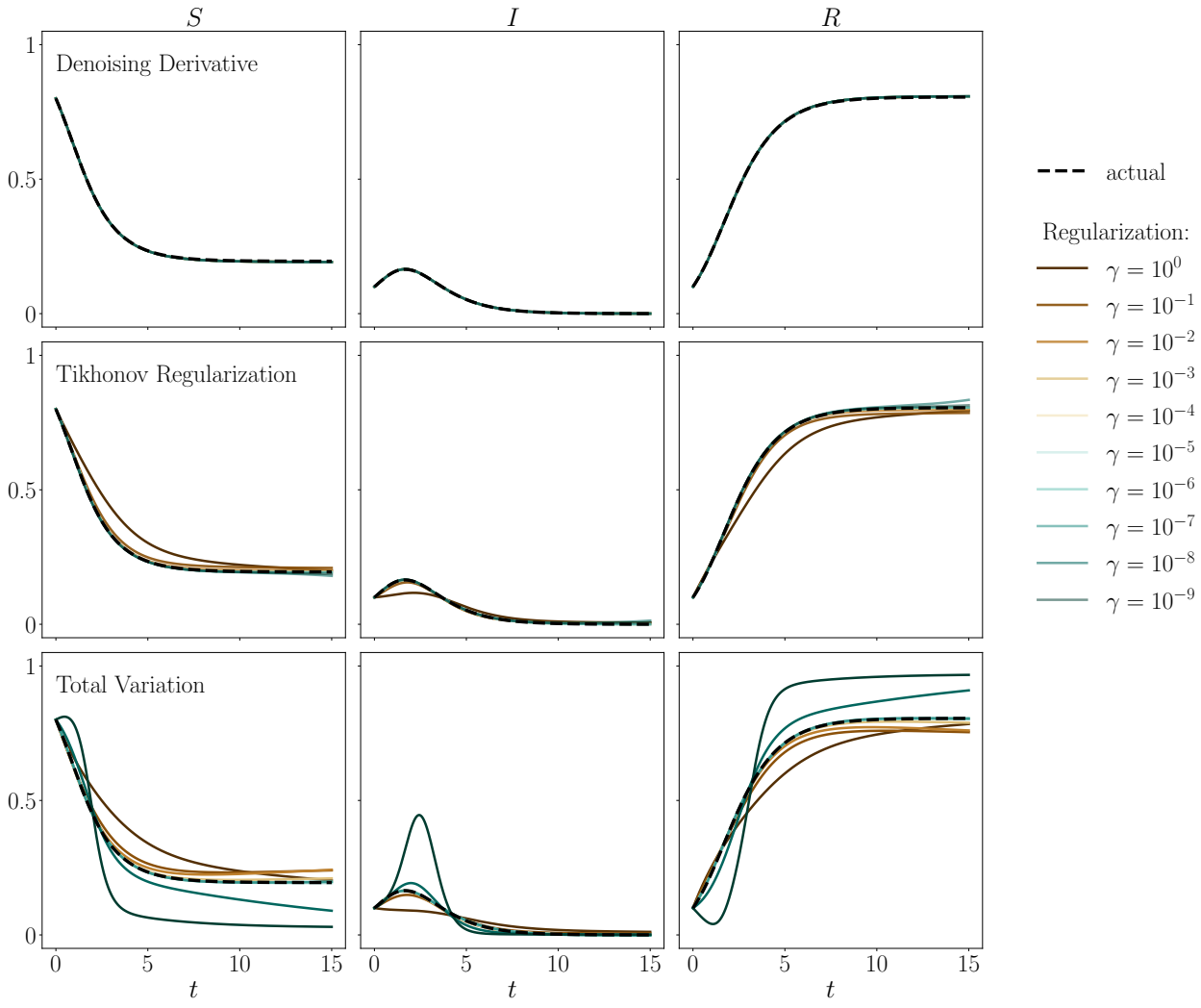


**Figure 10:** Three regularized differentiation methods when used for model identification, over different values of the differentiation's regularization parameter ($\gamma$). Dashed black curves show the numerical integration of a test SIR trajectory. Notice that the denoising derivative performs consistently for varying amount of regularization, while the other methods require some tweaking.

$$
\begin{aligned}
d(S)/dt =& 7.312195290815341e-17(S) - 0.0029846110491668736(I) \\
& - 0.002305440327027366(R) + 0.0035241332379654772(SS) \\
& + 0.010762043275610847(II) + 0.0024868711205385742(RR) \\
& - 2.0076248770571117(SI) - 3.440099612857256e-17(SR) \\
& - 1.847699547111957e-16(IR) \\
d(I)/dt =& 1.228359177988061e-16(S) - 0.7502237574779195(I) \\
& - 0.00016891337361823534(R) - 0.0015269183408076125(SS) \\
& - 0.25500283456471895(II) - 3.456445170450389e-16(RR) \\
& + 1.756149630694258(SI) + 0.001323760970765102(SR) \\
& - 0.2447354020299107(IR) \\
d(R)/dt =& 1.0735585662730585e-16(S) + 0.995236488253796(I) \\
& + 0.00785171488185314(R) - 0.0007519092829280412(SS) \\
& + 3.390521668553451e-17(II) - 0.007857054581881757(RR) \\
& + 0.0057878129647582(SI) - 0.007481891361761569(SR) \\
& - 2.186338771921895e-16(IR)
\end{aligned}
$$

**Figure 11:** Model identified using the denoising derivative with equally weighted 3-point moving average, $\gamma = 1$, and coefficient selection with $\alpha = 1$ and $\lambda = 10^{-3}$, as seen in Figure 10. Notice that some terms are quite small.

Figure 11 displays one system of equations obtained using the denoising derivative. Figure 10 displays integrations of the identified models from the starting point ($S = .8$, $I = .1$, $R = .1$) alongside the integration of the true system of equations. All three methods lead to satisfactory models for at least some values of $\gamma$, but the denoising derivative is more consistent in its performance.

The next example compares model identification performance with and without the use of conserved quantities to inform the coefficient selection, the former using Equation 3.6 and the latter omitting the term $\alpha \sum_{j=1}^{N} (f_1(y_j) + f_2(y_j) + f_3(y_j))^2$. The training data is the same as the previous example but with normally distributed noise of standard deviation .01 added to each point. The

denoising derivative is used with an equally weighted 3-point moving average and regularization parameter $\gamma = 10^{-2}$, and the denoised estimate of the trajectory is used in the regression in place of the original trajectories. Both with and without use of conserved quantities, the regression's $L_1$ parameter takes the values $\gamma = 10^{-1}$, $\gamma = 10^{-2}$, $\dots$ , $\gamma = 10^{-9}$, $\gamma = 10^{-10}$.

Figure 12 displays integrations of the identified models from the starting point $(S = .8, I = .1, R = .1)$ alongside the integration of the true system of equations. In both cases, there are values of $\lambda$ that lead to satisfactory models, and the difference is less striking in this second example, but when using conserved quantities the predictions tend to remain more accurate as $\lambda$ varies.

Finally, we compare the model identification performance of the denoising derivative with a simple difference quotient as the noise level varies. The training data is the same as the previous two examples but adds normally distributed noise with standard deviations $\sigma = .001$, $\sigma = .01$, $\sigma = .05$, and $\sigma = .1$. The denoising derivative uses an equally weighted 3-point moving average and regularization parameter $\gamma = 10$. The other derivative is a 3-point difference quotient sourced from the Python `derivative` library [27]. Each case performs coefficient selection with $\alpha = 1$, $\lambda = 10^{-3}$. Figure 13 compares both identified models with an integration of the true system. The denoising derivative yields more accurate predictions as $\sigma$ increases. It is worth noticing, however, that the difference quotient performs well at lower noise levels. When compared with Figure 10, the case $\sigma = .001$ shows that depending on the choice of regularization parameter, the simple difference quotient can outperform the total variation derivative and the derivative with Tikhonov penalty. This observation poses the denoising derivative as a bridge between the noise-amplifying but unbiased simple difference quotient and the accurate but biased regularized differentiation methods.

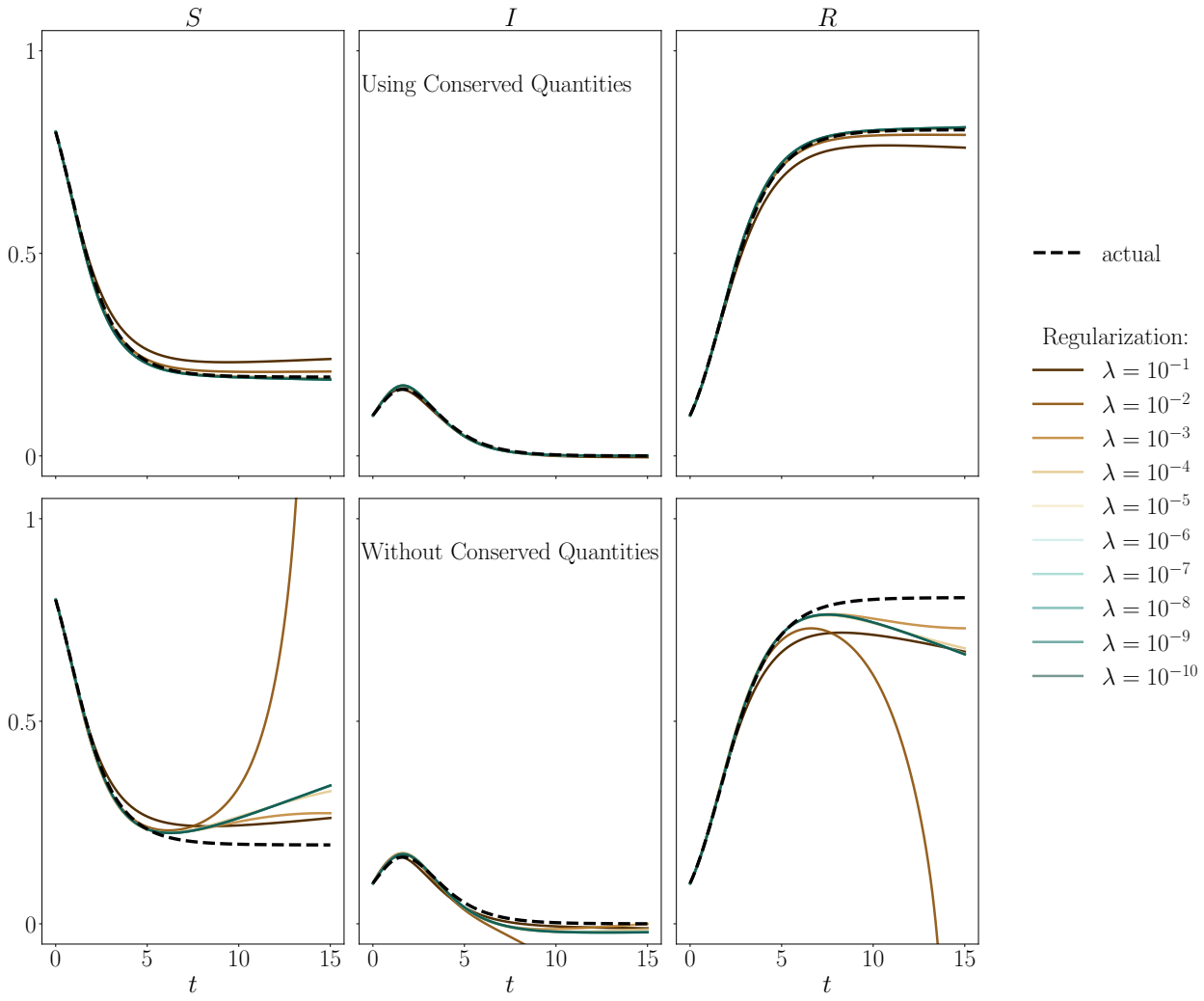


**Figure 12:** Model identification with and without use of conserved quantities, over different values of the model identification's $L_1$ regularization parameter ($\lambda$). Dashed black curves show the numerical integration of a test SIR trajectory. The use of conserved quantities results in improved predictive ability.

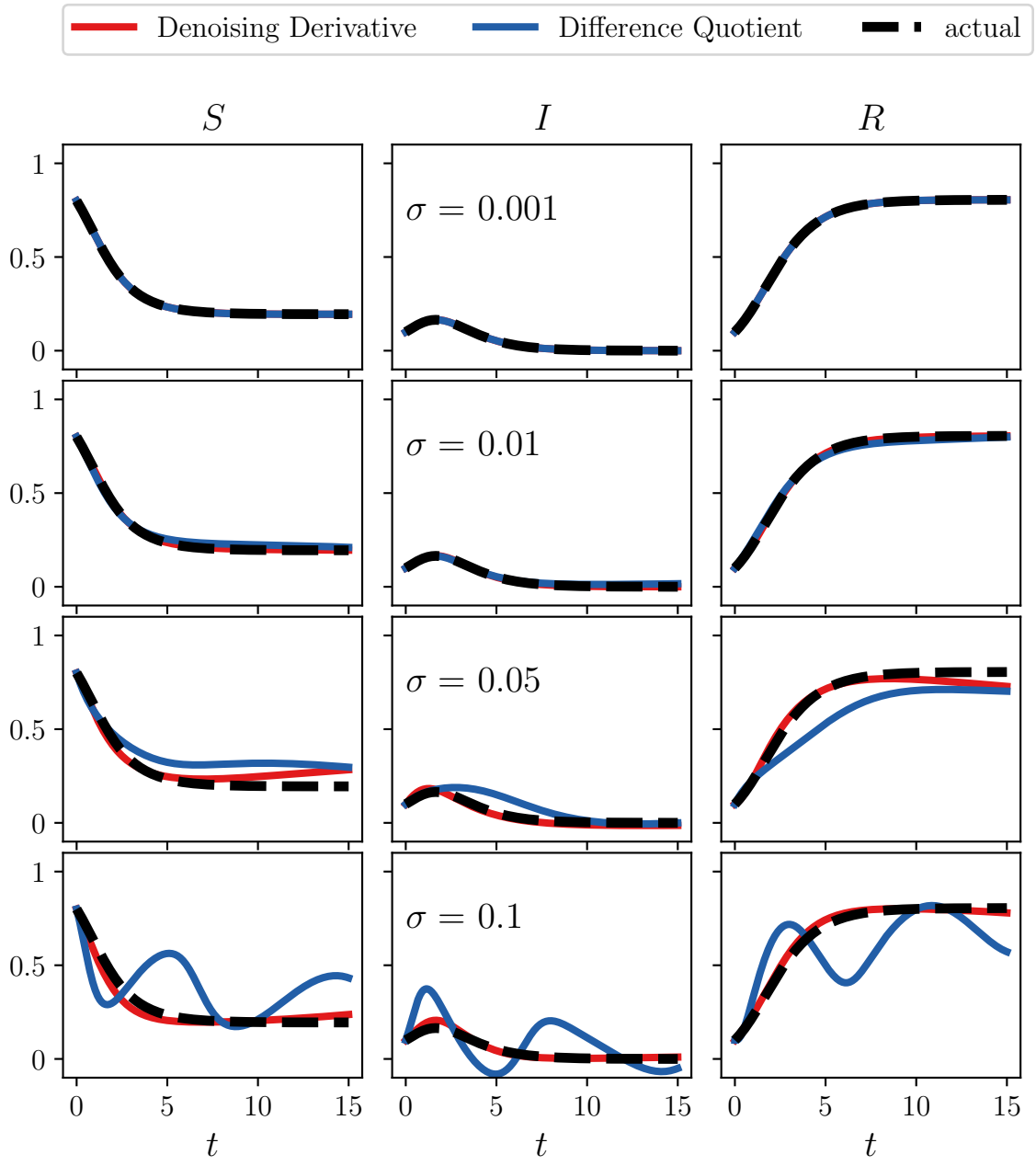


**Figure 13:** Model identification over four noise levels with the denoising derivative (equally weighted 3-point moving average, $\gamma = 10$) and a simple 3-point difference quotient [27]. Coefficient selection uses $\alpha = 1$, $\lambda = 10^{-3}$. The denoising derivative yields more accurate predictions as $\sigma$ increases. Notice, however, the accuracy of the difference quotient at low $\sigma$ and compare to some of the results in Figure 10.

# 4. Results

This chapter presents three examples of pair approximations for the coevolving SIS system obtained through sparse model identification. In the first example we identify a model using simulations with $N = 5,000$ nodes and $M = 17,500$ edges. All simulations take $r = .005$, $p$ covers the interval $(0, .008)$, and $w$ covers the interval $(0, .04)$. In the second example we identify a model using simulations with $N = 5,000$ nodes and $M = 5,000$ edges. All simulations take $r = .002$, $p$ covers the interval $(0, .006)$, and $w$ covers the interval $(0, .04)$. In the third example we identify a model using simulations with $N = 5,000$ nodes and $M = 50,000$ edges. All simulations take $r = .005$, $p$ covers the interval $(0, .04)$, and $W$ covers the interval $(0, .04)$. The parameter regions

for these examples correspond to examples in [14] and [1]. We assign initial networks Erdős-Rényi random graph topology.

In order for the identified models to describe the effect of the system parameter $r$, we use the scaling property mentioned in Chapter 3 to synthetically sample different values of $r$. Given time series of measurements and their corresponding numerically computed derivatives, the parameters and derivatives at each time point are multiplied by a scale factor. In this case, the scale factors alternate through 1, 2, 4, 6, and 8 at each time point. It would also be valid to simply simulate with the desired values of $r$.

The model selection produces equations by iteratively fitting to 30 initial terms:

$$\begin{gathered}
\{p, r, w, \\
pi, ri, wi, \\
pl_{II}, rl_{II}, wl_{II}, \\
pl_{SS}, rl_{SS}, wl_{SS}, \\
pi^2, ri^2, wi^2, \\
pl_{II}^2, rl_{II}^2, wl_{II}^2, \\
pl_{SS}^2, rl_{SS}^2, wl_{SS}^2, \\
pil_{II}, ril_{II}, wil_{II}, \\
pil_{SS}, ril_{SS}, wil_{SS}, \\
pl_{II}l_{SS}, rl_{II}l_{SS}, wl_{II}l_{SS}\}
\end{gathered} \tag{4.1}$$

using Equation 3.4 and removing terms with coefficients smaller than a threshold.

After obtaining a model we assess its performance by integrating along sample trajectories and by creating bifurcation plots. Both tasks are aided by obtaining the Jacobian matrix for the data-driven equations. As one of the benefits of describing time series with differential equations, as opposed to some kind of black-box model, the Jacobian matrix can be obtained symbolically using a computer algebra system to improve integration performance when supplied to a stiff ODE solver. We obtain the symbolic Jacobian using the Python library Sympy [45], and we integrate the equations, with the help of the Jacobian, using the integrator `scipy.integrate.BDF` [46]. To create bifurcation plots as shown in Figures 17, 21, and 24, we numerically solve the equations and stop integration when a convergence criterion is satisfied.

## 4.1 First Example

In all simulations used for Example 1, the networks contain $N = 5,000$ nodes and $M = 17,500$ edges and use $r = .005$. The values of $w$ sampled are 0, .01, .02, .03, and .04. For each value of $w$ we sample 30 values of $p$, equally spaced from $p = 0$ to $p = .008$. For each combination of parameters we simulate from five initial conditions: $i_0 = .001$, $i_0 = .01$, $i_0 = .05$, $i_0 = .99$, and $i_0 = .999$. Each simulation runs for 2000 steps, which is typically sufficient for the number of infectious nodes to reach equilibrium.

The recordings of $i$, $l_{II}$, and $l_{SS}$, and the computed series of $s$ and $l_{SI}$, are smoothed with a 41-point moving average, tapered to avoid cutting off the endpoints. Numerical differentiation uses the denoising derivative with $\gamma = .1$ and equally weighted 3-point moving average. The smoothed time series given by the denoising derivative is maintained, and the output of the derivative is smoothed with another 5-point moving average, tapered to avoid cutting off the endpoints. To reduce memory load, every 50 points are maintained from each series of positions and derivatives, leaving 40 time samples from each simulation. The remaining points are then alternately scaled by 1, 2, 4, 6, and 8.

Using the set of terms 4.1 and optimizing according to Equation 3.4 with $\alpha = 1$, $\lambda_1 = .001$, and $\lambda_2 = .00001$, terms with coefficients of absolute value less than $10^{-4}$ are removed, and the process repeats until no terms are removed. In this example, 3 optimizations result in 5 equations with 23, 23, 25, 27, and 28 terms each. The equations for $i$, $l_{II}$, and $l_{SS}$ (the others are unused) are displayed in Figure 14.

Figures 15 and 16 display integrations of the resulting equations against actual simulations and the integration of the analytically derived pair approximations (Equation 1.1). The bifurcation plots in Figure 17 display steady states after a stretch of time, covering the parameters used as training data. The steady states of the simulations average the values of the system variables between steps 9,000 and 10,000. The steady states of the analytically derived pair approximations give the values after integrating over 10,000 units of time. The data-driven equations are considered to reach equilibrium when $i$ remains within a window of size .005 after 100 units of time. If the integration does not satisfy the convergence criterion, its value after 10,000 units of time is taken as its steady state. This rather particular criterion addresses one of the key weaknesses of the data-driven equations. Probably owing to the complexity of the data-driven equations and the emperical nature of their derivation, they demonstrate instability around their equilibrium values. Even after predicting well for a long interval and reaching the correct equilibrium, they have a tendency

$$
\begin{aligned}
d(i)/dt =& 0.0809(pi) - 0.1243(ri) - 0.0823(wi) - 0.0697(rii) + 0.1442(pil_{SS}) \\
& - 0.1877(ril_{II}) - 0.2791(ril_{SS}) + 0.0826(wil_{II}) - 0.0724(wil_{SS}) + 1.3658(p) \\
& - 0.1604(w) + 0.5962(pl_{II}) + 0.8973(pl_{SS}) - 0.1799(wl_{II}) + 0.0482(wl_{SS}) \\
& + 0.5161(pl_{II}l_{SS}) - 0.2935(pl_{II}l_{II}) - 0.4498(pl_{SS}l_{SS}) - 0.6053(rl_{II}l_{SS}) \\
& - 0.2425(rl_{II}l_{II}) + 0.0837(wl_{II}l_{SS}) + 0.0565(wl_{II}l_{II}) + 0.0016(wl_{SS}l_{SS}) \\
d(l_{II})/dt =& 1.6433(pi) + 0.2255(pii) - 0.2677(rii) + 0.2044(pil_{II}) - 0.2704(pil_{SS}) \\
& - 0.6635(ril_{II}) + 0.5135(wil_{II}) + 0.0170(wil_{SS}) + 4.4206(p) + 0.2643(r) \\
& - 0.5692(w) + 4.7223(pl_{II}) + 0.2085(pl_{SS}) + 0.7232(rl_{II}) - 0.3192(rl_{SS}) \\
& - 0.1182(wl_{II}) - 0.1034(wl_{SS}) + 0.9237(pl_{II}l_{SS}) - 1.9400(pl_{II}l_{II}) - 0.5637(pl_{SS}l_{SS}) \\
& - 2.4757(rl_{II}l_{SS}) - 1.8641(rl_{II}l_{II}) + 0.0788(rl_{SS}l_{SS}) - 0.1242(wl_{II}l_{II}) \\
& + 0.0977(wl_{SS}l_{SS}) \\
d(l_{SS})/dt =& - 0.0347(pi) + 0.7670(ri) - 0.0591(wi) + 0.5517(rii) - 0.9824(wii) \\
& + 0.3106(pil_{II}) - 1.1035(pil_{SS}) + 0.7502(ril_{II}) + 0.3928(ril_{SS}) - 0.7552(wil_{II}) \\
& + 0.1211(wil_{SS}) - 3.6570(p) + 0.0591(r) + 0.5669(w) - 5.0167(pl_{SS}) \\
& + 1.5303(rl_{II}) - 0.0204(rl_{SS}) + 1.4178(wl_{II}) + 0.6085(wl_{SS}) - 3.0719(pl_{II}l_{SS}) \\
& + 0.3405(pl_{II}l_{II}) + 2.0789(pl_{SS}l_{SS}) + 1.6048(rl_{II}l_{SS}) - 0.6702(rl_{II}l_{II}) \\
& - 0.4207(wl_{II}l_{SS}) - 0.1571(wl_{II}l_{II}) - 0.2657(wl_{SS}l_{SS})
\end{aligned}
$$

**Figure 14:** Data-driven model for Example 1. Coefficients are truncated for brevity.

to quickly diverge. The convergence criterion attempts to balance accuracy while keeping the integration short enough to avoid diverging. Indeed, some of the points in the bifurcation plots are missing, in which case the data-driven equations diverge at the given parameter setting. The steady states of the data-driven equations are evaluated from two intial conditions: one close to $i_0 = 0$ and another close to $i_0 = 1$, with the values of $r$ and $w$ the same as the training data. They are evaluated at 20 values of $p$ equally spaced in the interval used for training data. The steady states of the analytically derived pair approximation are evaluated at every initial condition and parameter combination used for training data (30 values of $p$). This discrepancy tests the performance of the data-driven equations slightly offset from their training data, and it reduces the number of computations necessary.

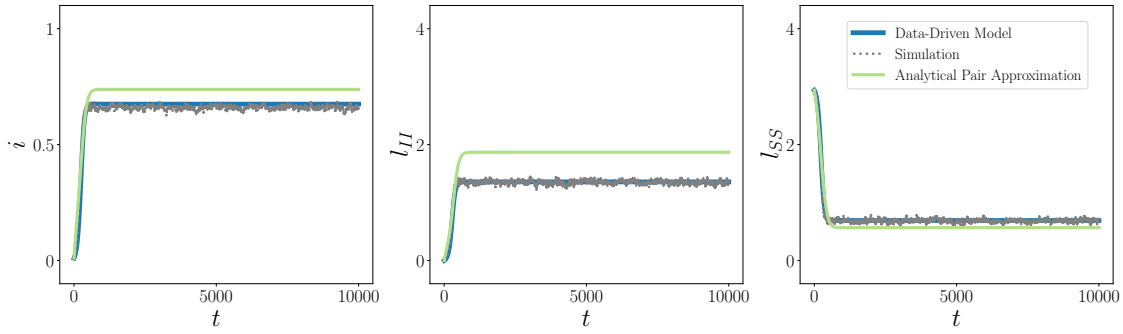


**Figure 15:** Comparison of the data-driven and analytically derived pair approximations on a trajectory of the coevolving SIS system from Example 1. Both evolve from initial condition $(i = .01, l_{II} = .0002, l_{SS} = 2.9384)$, with system parameters $p = .00346667, r = .005, w = .01$.

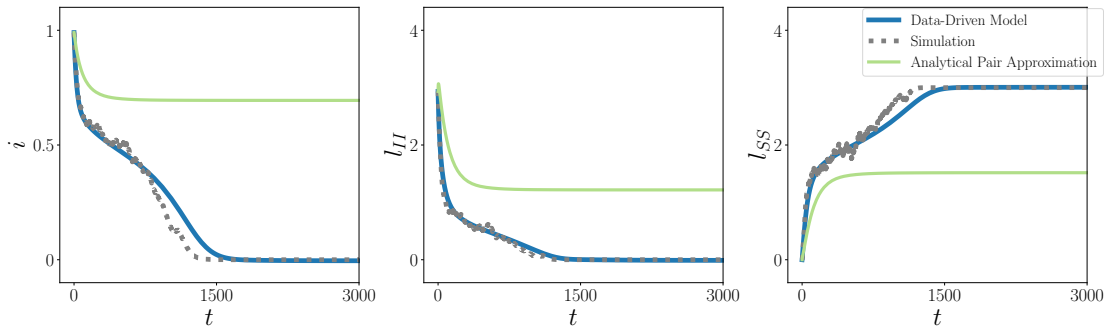


**Figure 16:** Comparison of the data-driven and analytically derived pair approximations on a trajectory of the coevolving SIS system from Example 1. Both evolve from initial condition $(i = .99, l_{II} = 2.9348, l_{SS} = 0)$, with system parameters $p = .00453333, r = .005, w = .04$.

Figures 15, 16, and 17 show that the data-driven equations can outperform the analytically-derived

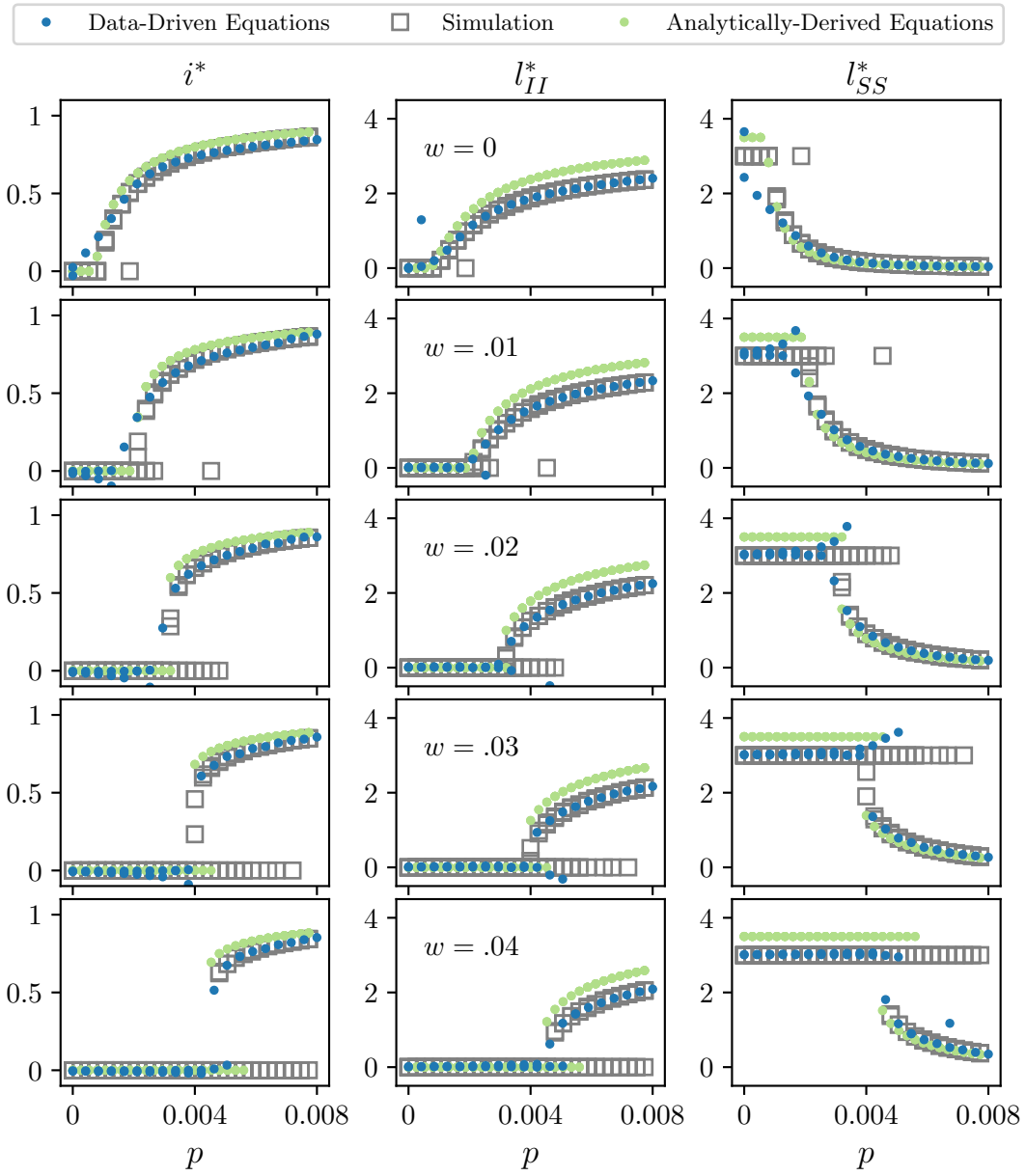


**Figure 17:** Bifurcation plots for Example 1.

pair approximations in some cases. It is interesting to note that in all three examples in this chapter, the data-driven equations show some of the same shortcomings as the analytically derived pair approximations. In particular, they both capture the discontinuous dependence on the parameter $p$, but they both struggle in the bistable region, leaving an empty stretch of grey squares at $i^* = 0$. The data-driven equations struggle a bit more in the later examples.

## 4.2 Second Example

In all simulations used for Example 2, the networks contain $N = 5,000$ nodes and $M = 5,000$ edges and use $r = .005$. The values of $w$ sampled are 0, .01, .02, .03, and .04. For each value of $w$ we sample 30 values of $p$, equally spaced from $p = 0$ to $p = .04$. For each combination of parameters we simulate from five initial conditions: $i_0 = .001$, $i_0 = .01$, $i_0 = .05$, $i_0 = .99$, and $i_0 = .999$. Each simulation runs for 2000 steps.

The treatment of the training data is largely the same as in Example 1. The denoising derivative uses $\gamma = 10^4$, and the coefficient optimization uses $\alpha = 1$, $\lambda_1 = .1$, and $\lambda_2 = .001$. Terms are again removed with a threshold, but we use individualized thresholds $10^{-2}$, $10^{-2}$, $10^{-1}$, $10^{-1}$, and $10^{-1}$ for the 5 equations. 6 cycles of optimization and term removal result in 5 equations with 15, 15, 14, 13, and 11 terms each. The bifurcation plots consider the data-driven equations converged when $i$ remains in a window of size .001 for 100 units of time.

Figure 18 displays the system of equations obtained in this example. Figures 19 and 20 plot integrations of the resulting equations against actual simulations and the integration of the analytically derived pair approximations (Equation 1.1).

## 4.3 Third Example

In all simulations used for Example 3, the networks contain $N = 5,000$ nodes and $M = 50,000$ edges and use $r = .002$. The values of $w$ sampled are 0, .01, .02, .03, and .006. For each value of $w$ we sample 30 values of $p$, equally spaced from $p = 0$ to $p = .04$. For each combination of parameters we simulate from five initial conditions: $i_0 = .001$, $i_0 = .01$, $i_0 = .05$, $i_0 = .99$, and $i_0 = .999$. Each simulation runs for 2000 steps.

The treatment of the training data is largely the same as in Example 1. The denoising derivative uses $\gamma = 1$, and the coefficient optimization uses $\alpha = 1$, $\lambda_1 = .001$, and $\lambda_2 = .001$. Terms are removed with individualized thresholds $10^{-4}$, $10^{-4}$, $10^{-3}$, $10^{-3}$, and $10^{-3}$ for the 5 equations. 3 cycles of optimization and term removal result in 5 equations with 19, 19, 26, 26, and 20 terms

$$
\begin{aligned}
d(i)/dt = &-1.0006438403665256(ri) + 0.09777038130707723(pii) \\
&- 0.5501074035856202(rii) + 0.09844952403641638(wii) \\
&- 0.1111508218669941(ril_{II}) - 0.05126570955930179(ril_{SS}) \\
&- 0.24260996897557557(wil_{SS}) + 0.6638855330462314(p) \\
&- 0.0432689191465659(r) - 0.11260960525735672(w) \\
&- 0.1264719889050649(rl_{II}) - 0.7350979272600652(pl_{II}l_{II}) \\
&- 0.6576451221453764(pl_{SS}l_{SS}) + 0.02848693941797711(rl_{SS}l_{SS}) \\
&+ 0.1125858639606359(wl_{SS}l_{SS}) \\
d(l_{II})/dt = &0.14473164636098368(pi) - 1.000874570365459(ri) \\
&- 0.6218683306693004(rii) + 0.2663423127246033(wii) \\
&- 0.20250435862301122(pil_{SS}) - 0.34564186407688013(ril_{II}) \\
&+ 0.9535702436934349(p) - 0.26166736215596803(w) \\
&- 0.48849104620051426(pl_{SS}) - 0.4289347236398379(rl_{II}) \\
&- 1.071879678926603(pl_{II}l_{II}) - 0.4601443824311606(pl_{SS}l_{SS}) \\
&- 0.17661416328506777(rl_{II}l_{II}) + 0.2595418720021471(wl_{SS}l_{SS}) \\
d(l_{SS})/dt = &0.9285387595803297(ri) + 0.17343395174346787(ril_{SS}) \\
&- 0.32834718710861555(wil_{II}) - 0.7763010244683582(p) \\
&+ 0.17574608278496298(r) + 0.2804551388649336(w) \\
&+ 0.31431187025266333(pl_{II}) - 0.24901421987824918(pl_{SS}) \\
&+ 0.2804389296421176(rl_{II}) + 0.49201794552515693(pl_{II}l_{II}) \\
&+ 1.01447714962108(pl_{SS}l_{SS}) - 0.17542188189093966(rl_{SS}l_{SS}) \\
&- 0.2770356563483522(wl_{SS}l_{SS})
\end{aligned}
$$

**Figure 18:** Data-driven model for Example 2.

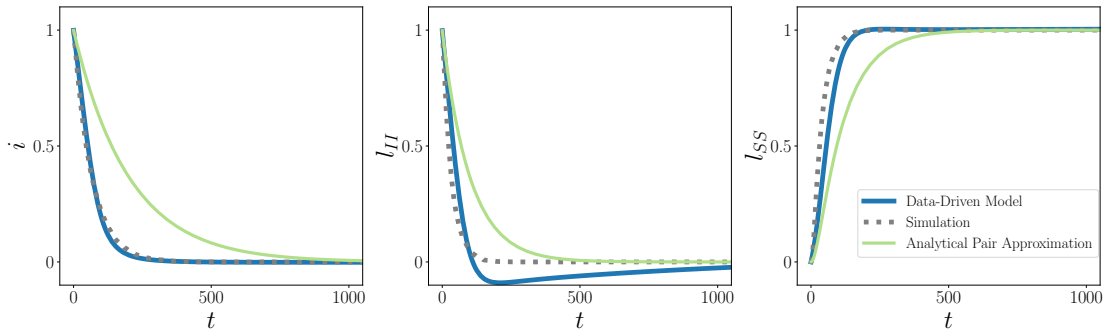


**Figure 19:** Comparison of the data-driven and analytically derived pair approximations on a trajectory of the coevolving SIS system from Example 2. Both evolve from initial condition $(i = .999, l_{II} = .9978, l_{SS} = 0)$, with system parameters $p = 0, r = .005, w = .04$.

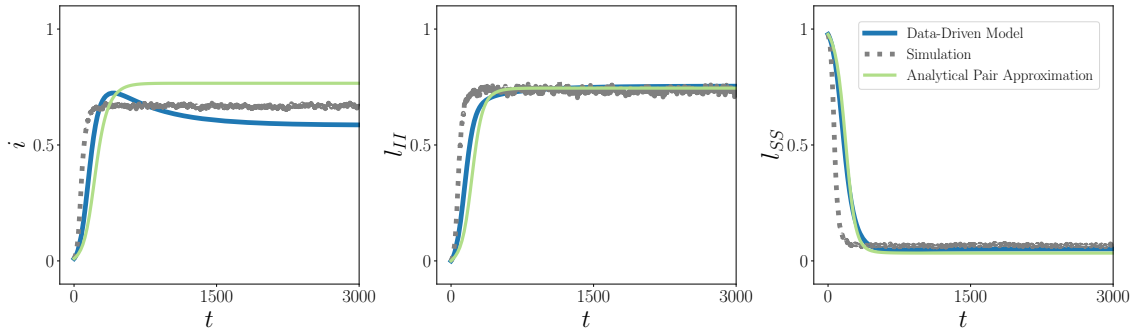


**Figure 20:** Comparison of the data-driven and analytically derived pair approximations on a trajectory of the coevolving SIS system from Example 2. Both evolve from initial condition $(i = .01, l_{II} = 0, l_{SS} = .9772)$, with system parameters $p = .0173333, r = .005, w = 0$.

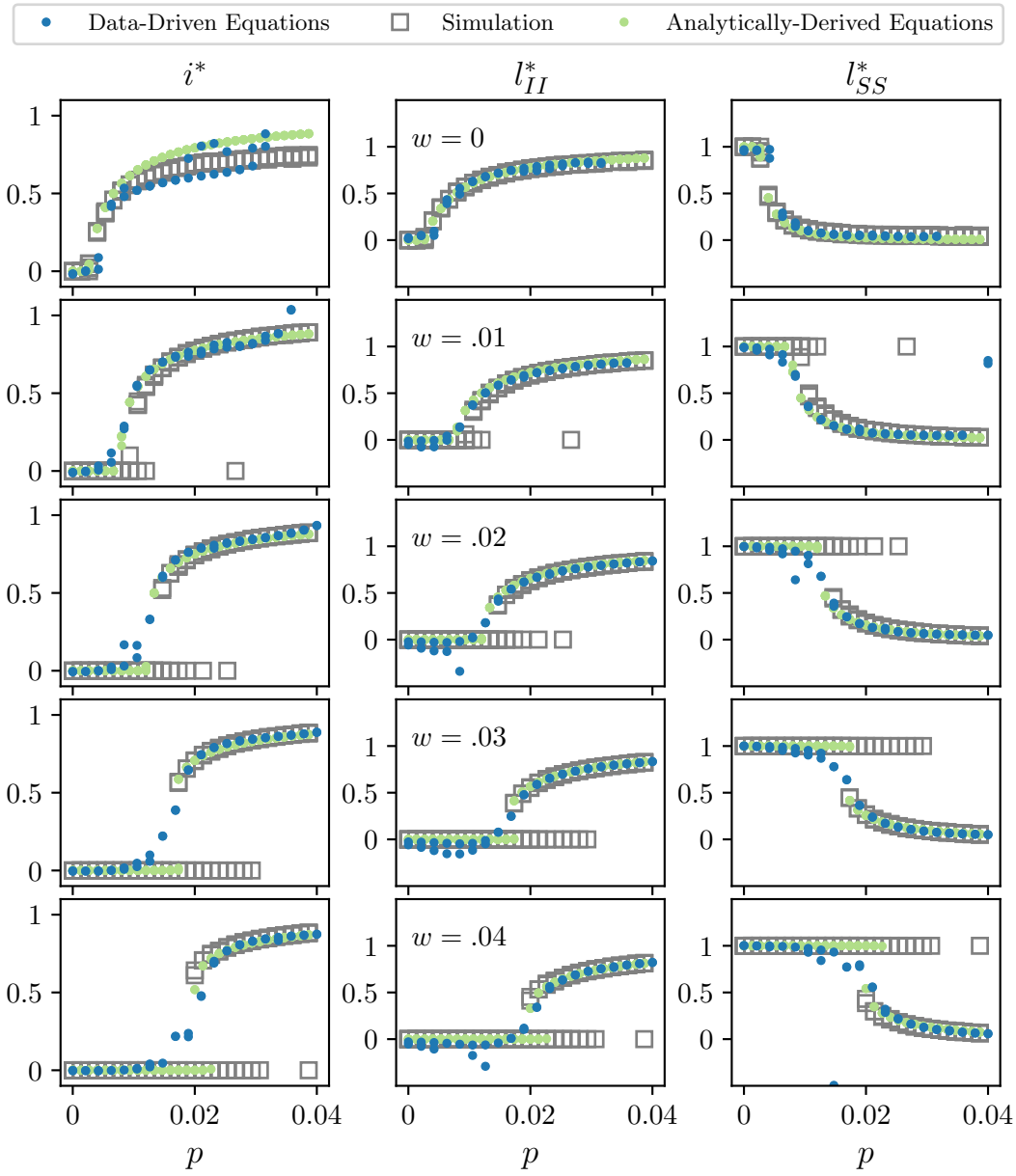


**Figure 21:** Bifurcation plots for Example 2.

$$
\begin{aligned}
d(i)/dt =& 0.0002(pil_{II}) + 0.0107(pil_{SS}) - 0.0106(wil_{II}) + 0.0016(p) + 0.0053(w) \\
&+ 0.0132(pl_{II}) + 0.0170(pl_{SS}) + 0.0017(rl_{SS}) + 0.0030(wl_{II}) + 0.0286(wl_{SS}) \\
&+ 0.0901(pl_{II}l_{SS}) + 0.0007(pl_{II}l_{II}) + 0.0195(pl_{SS}l_{SS}) + 0.0113(rl_{II}l_{SS}) \\
&- 0.0039(rl_{II}l_{II}) + 0.0006(rl_{SS}l_{SS}) - 0.0024(wl_{II}l_{SS}) + 0.0005(wl_{II}l_{II}) \\
&- 0.0034(wl_{SS}l_{SS}) \\
d(l_{II})/dt =& 0.0284(pi) + 0.0019(ri) + 0.0083(pii) - 0.0271(wii) + 0.0509(pil_{II}) \\
&+ 0.1420(pil_{SS}) + 0.0123(ril_{SS}) - 0.1101(wil_{II}) - 0.0536(wil_{SS}) + 0.0569(p) \\
&+ 0.0141(r) + 0.0840(w) + 0.1921(pl_{II}) + 0.1852(pl_{SS}) + 0.0437(rl_{II}) + 0.0397(rl_{SS}) \\
&+ 0.1348(wl_{II}) + 0.2172(wl_{SS}) + 1.0015(pl_{II}l_{SS}) + 0.0026(pl_{II}l_{II}) \\
&+ 0.0971(pl_{SS}l_{SS}) + 0.1514(rl_{II}l_{SS}) - 0.0602(rl_{II}l_{II}) - 0.0269(wl_{II}l_{SS}) \\
&- 0.0056(wl_{II}l_{II}) - 0.0254(wl_{SS}l_{SS}) \\
d(l_{SS})/dt =& -0.0244(pi) - 0.0051(pii) + 0.0303(wii) - 0.0348(pil_{II}) - 0.1459(pil_{SS}) \\
&+ 0.0028(ril_{II}) - 0.0171(ril_{SS}) + 0.1326(wil_{II}) - 0.0108(wil_{SS}) - 0.0553(p) \\
&- 0.0135(r) - 0.0948(w) - 0.1543(pl_{II}) - 0.2120(pl_{SS}) - 0.0267(rl_{II}) - 0.0534(rl_{SS}) \\
&- 0.0706(wl_{II}) - 0.3313(wl_{SS}) - 0.8949(pl_{II}l_{SS}) - 0.2278(pl_{SS}l_{SS}) \\
&- 0.1232(rl_{II}l_{SS}) + 0.0426(rl_{II}l_{II}) - 0.0115(rl_{SS}l_{SS}) + 0.0435(wl_{II}l_{SS}) \\
&- 0.0037(wl_{II}l_{II}) + 0.0405(wl_{SS}l_{SS})
\end{aligned}
$$

**Figure 22:** Data-driven model for Example 3. Coefficients are truncated for brevity.

each. The bifurcation plots consider the data-driven equations converged when $i$ remains in a window of size .001 for 100 units of time.

Figure 22 displays the system of equations obtained in this example. Figure 23 plots an integration of the resulting equations against an actual simulation and an integration of the analytically derived pair approximations (Equation 1.1).

Figure 23 exemplifies how the data-driven equations can evolve accurately and reach the correct equilibrium before suddenly destabilizing and diverging. Notice that in the bifurcation plots of Figure 24, some intervals of $p$ display no blue dots, whereupon the integrations of the data-driven

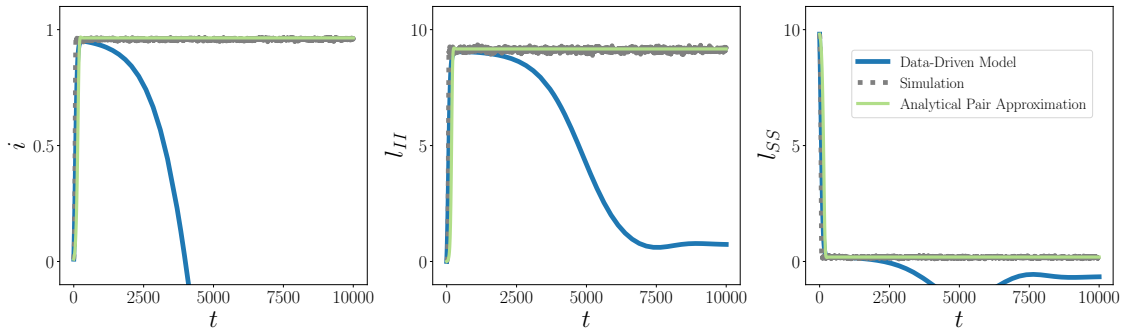


**Figure 23:** Comparison of the data-driven and analytically derived pair approximations on a trajectory of the coevolving SIS system from Example 3. Both evolve from initial condition $(i = .01, l_{II} = .0012, l_{SS} = 9.805)$, with system parameters $p = .003, r = .002, w = .03$.

equations diverge without satisfying the convergence criterion.

## 4.4 Discussion

It is worth addressing that the analysis of these examples compares the performance of the data-driven models with the same data that is used to train them. While validation against out-of-sample data demonstrates a model's ability to generalize beyond the training data, at the moment we are focused on simply obtaining a model that describes the data used to train it. Figures 17, 21, and 24 illustrate that there is room for improvement in two primary areas: first, replicating the bistability that emerges with increasing values of $w$ and, second, consistently evolving to the steady states without diverging. Validation with out-of-sample data will be of interest upon meeting these goals. Still, agreement with training data carries a bit more importance when performing regression to fit differential equations than it does when fitting algebraic equations, given the more layered objective of creating equations that admit both stable and accurate solution when supplied to an ODE solver.

The preceeding results suggest a few actionable steps for improvement. The most apparent action is to perform the simulations with larger networks. Increasing from $N = 5,000$ nodes to at least $N = 30,000$ nodes is computationally obtainable and would probably result in smoother training data more amenable to numerical differentiation and description with a low-dimensional, deterministic model. A related step to take is to be thoughtful about how long we simulate for our training data. Once the simulation reaches its steady state, simulating for longer time gives no new information. A solution would be to simulate only until a convergence criterion is

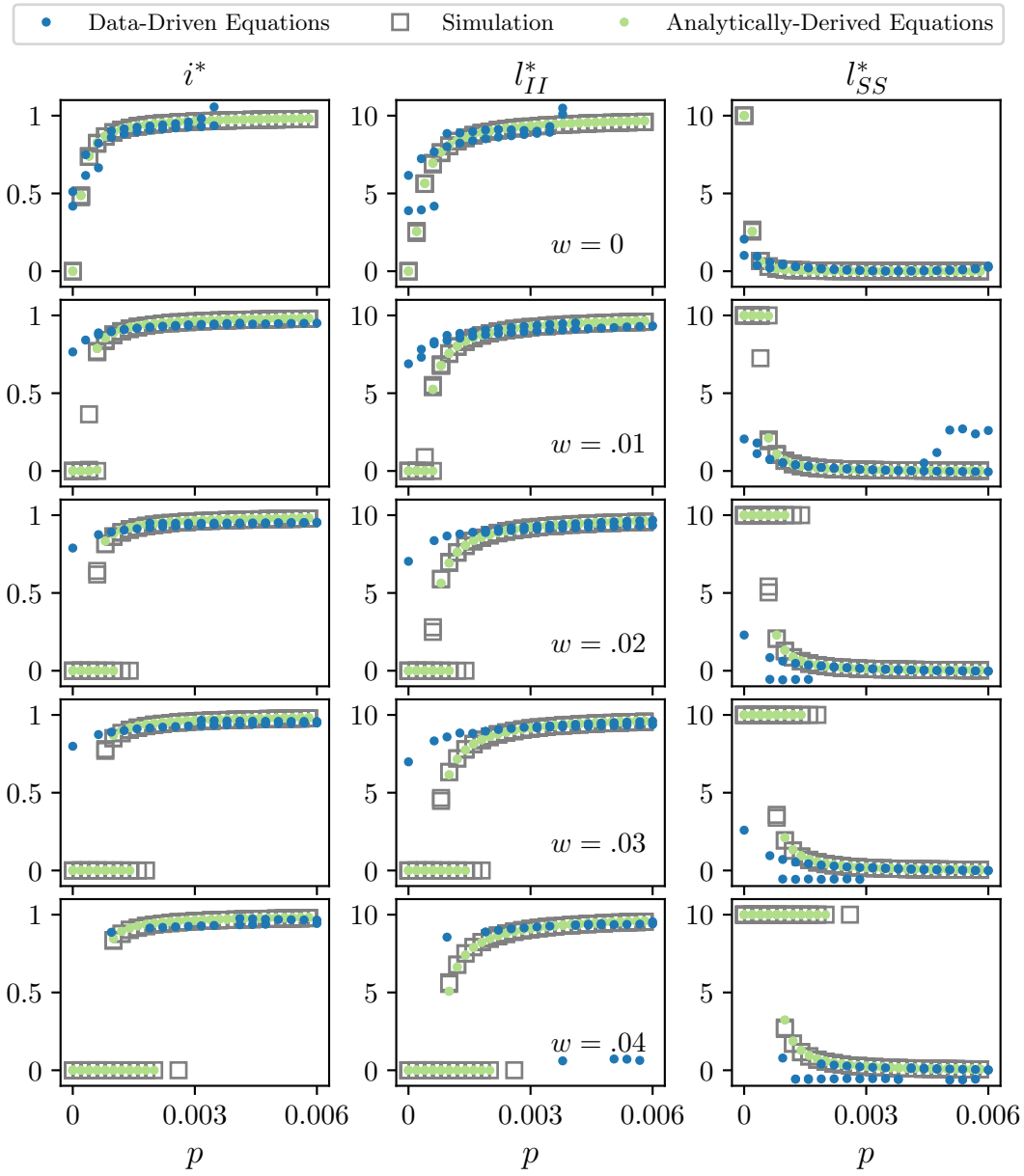


**Figure 24:** Bifurcation plots for example 3.

satisfied. In order for parameter settings with short time series to have equal representation in the regression, an interpolation method can allow every time series to have the same number of time points.

The choice of basis functions, regularization parameters, threshold values, and settings for smoothing and numerical differentiation are all chosen manually in these examples. All of these settings require tweaking in order to acheive the reported results, and automating their selection with cross-validations would streamline the experience of using the presented method.

# 5. Conclusion

Some directions for further work stand out. We first would like to create a data-driven model that describes the coevolving SIS system for general mean degree, rather than creating different models for different mean degrees. This extension would make for a more sportsmanlike comparison with the analytically derived pair approximations and perhaps with the more accurate approximate master equations. Another direction, in line with the greater purpose of this work, is to create data-driven equations for coevolving network systems that do not already have analytically derived equations to describe them. For instance, repeating this work for an equivalent coevolving SIR model would yield a model that is not currently in circulation. These next steps proceed

through the same method, and we will achieve them by further refining the techniques presented here.

Reproducing some of the key behavior of the coevolving SIS system is an encouraging step towards a reliable procedure to construct data-driven equations for coevolving network systems. Subjectively, even after the review in Chapter 2, the application of coevolving networks may not immediately appear to be worth the effort. Given all of the things in science and engineering that can be described by systems of differential equations, one may wonder why we direct our attention to simplistic computational models. Part of the value of this application is its difficulty. Working against the dependence on several system parameters and the stochastic effects ill-suited to numerical differentiation, every modest observation and improvement in understanding we make about model identification is available to people working in other application areas. In a different direction, the greater value of the work is, perhaps, enabling better understanding of networks.

In Chapter 2 we saw a selection of coevolving network systems and discussed how they model systems with interconnected, interacting parts, in particular when the pattern of connection changes in time. Some of the application settings are epidemics, spread of opinions, and evolutionary game theory. In the broader context, networks are used to codify food webs, transportation systems, communication infrastructure, the world wide web, biochemical function, the brain, academic collaboration, and more [4]. Thresholding on measures of time series correlation is used to build complex networks from spatiotemporal climate records [47, 48]. All of these systems change in time, just like our coevolving networks. Sometimes this change is rapid and consequential. Diseases break out, dissent turns to revolution, ecosystems collapse, supply chains fail, normal brain function turns to epilepsy [49], and positive feedbacks may lead to virtually irreversible change in the Earth's climate system. Figures 17, 21, and 24 in Chapter 4 illustrate the sensitive dependence of the coevolving SIS system on its parameters. Increasing the value of $p$ causes the steady state at zero infections to disappear, and the steady state changes to a significant portion of the population infected. Applying this line of analysis to other network systems, bridging the gap between complex, fine-scale interactions and essential, global-scale behavior, could enable better understanding of consequential transitions and possibly how to predict, avoid, or cause them.

# References


[1] Gross, T., Dommar D'Lima, C., Blasius, B. (2006). Epidemic Dynamics on an Adaptive Network. *Physical Review Letters*, 96, 208701. 10.1103/PhysRevLett.96.208701

[2] Gross, T., Blasius, B. (2007). Adaptive Coevolutionary Networks: a Review. *Journal of the Royal Society Interface*, 5, 259-271. 10.1098/rsif.2007.1229

[3] Newman, M.E.J. (2003). "The Structure and Function of Complex Networks." *SIAM Review, 45*(2), 167-256. https://doi.org/10.1137/S003614450342480

[4] Newman, M.E.J. (2018). *Networks*. Oxford University Press, 2018. DOI: 10.1093/oso/9780198805090.001.0001

[5] Holme, P., Newman, M.E.J. (2006). "Nonequilibrium Phase Transition in the Coevolution of Networks and Opinions." *Physical Review E, 74*(5). 10.1103/PhysRevE.74.056108

[6] Durrett, R., Gleeson, J.P., Lloyd, A.L., Mucha, P.J., Shi, F., Sivakoff, D., Socolar, J.E.S., Varghese, C. (2012). "Graph Fission in an Evolving Voter Model." *Proceedings of the National Academy of Sciences, 109*(10), 3682-3687. https://doi.org/10.1073/pnas.1200709109

[7] Weibull, J.W. (1997). *Evolutionary Game Theory*. MIT Press, 1997. isbn: 9780262731218

[8] Fu, F., Wu, T., Wang, L. (2009). "Partner Switching Stabilizes Cooperation in Coevolutionary Prisoner's Dilemma." *Physical Review E, 79*(3). 10.1103/PhysRevE.79.036101

[9] Poundstone, W. (1993). *Prisoner's Dilemma: John Von Neumann, Game Theory and the Puzzle of the Bomb*. Anchor Books. isbn: 9780385415804

[10] Gleeson, J.P. (2011) "High-Accuracy Approximation of Binary-State Dynamics on Networks." *Physical Review Letters*, 107, 068701. https://doi.org/10.1103/PhysRevLett.107.068701

[11] Lee, H.W., Malik, N., Shi, F., Mucha, P.J. (2019). "Social Clustering in Epidemic Spread on Coevolving Networks." *Physical Review E, 99*(6). 10.1103/PhysRevE.99.062301

[12] Malik, N., Mucha, P.J. (2013). "Role of Social Environment and Social Clustering in Spread of Opinions in Coevolving Networks." *Chaos* (4), 043123. doi: 10.1063/1.4833995

[13] Malik, N., Shi, F., Lee, H.W., Mucha, P.J. (2016). "Transitivity Reinforcement in the Coevolving Voter Model." *Chaos* (12), 123112. doi: 10.1063/1.4972116

[14] Marceau, V., Noël, P.A., Hébert-Dufresne, L., Allard, A., Dubé, L.J. (2010). “Adaptive Networks: Coevolution of Disease and Topology." *Physical Review E, 82*(3). 10.1103/PhysRevE.82.036116

[15] Keeling, M.J., Eames, K.T.D. (2005). “Networks and Epidemic Models." *Journal of the Royal Society Interface, 2*(4), 295-307. 10.1098/rsif.2005.0051

[16] Lee, H.W., Malik, N., Mucha, P.J. (2018). “Evolutionary Prisoner’s Dilemma Games Coevolving on Adaptive Networks." *Journal of Complex Networks, 6*(1), 1-23. https://doi.org/10.1093/comnet/cnx018

[17] Demirel, G., Vazquez, F., Böhme, G.A., Gross, T. (2014). “Moment-Closure Approximations for Discrete Adaptive Networks." *Physica D: Nonlinear Phenomena, 267*, 68-80. https://doi.org/10.1016/j.physd.2013.07.003

[18] Erdös, P., Rényi, A. (1960) “On the Evolution of Random Graphs." *Publ. Math.Inst. Hung. Acad. Sci, 5*(1), 17–60.

[19] Brunton, S.L., Proctor, J.L., Kutz, J.N. (2016). “Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems." *Proceedings of the National Academy of Sciences, 113*(15), 3932-3937. 10.1073/pnas.1517384113

[20] Chartrand, R. (2011). “Numerical Differentiation of Noisy, Nonsmooth Data." *International Scholarly Research Notices*, vol. 2011, Article ID 164564. https://doi.org/10.5402/2011/164564

[21] Cullum, J. (1971). “Numerical Differentiation and Regularization." *SIAM Journal of Numerical Analysis, 8*(2), 254-265. https://doi.org/10.1137/0708026

[22] Cheng, J., Jia, X.Z., Wang, Y.B. (2007). “Numerical Differentiation and its Applications." *Inverse Problems in Science and Engineering, 15*(4), 339-357. https://doi.org/10.1080/17415970600839093

[23] Van Breugael, F.V., Kutz, J.N., Brunton, B.W. (2020). “Numerical Differentiation of Noisy Data: A Unifying Multi-Objective Optimization Framework." *IEEE Access, 8*, 196865-196877. 10.1109/ACCESS.2020.3034077

[24] Diamond, S., Boyd, S. (2016). “CVXPY: A Python-Embedded Modeling Language for Convex Optimization." *Journal of Machine Learning Research, 17*(83), 1-5.

[25] Agrawal, A., Verschueren, R., Diamond, S., Boyd, S. (2018). “A Rewriting System for Convex Optimization Problems." *Journal of Control and Decision, 5*(1), 42-60.

[26] Hethcote, H.W. (2000). “The Mathematics of Infectious Diseases." *SIAM Review, 42*(4), 599-653. 10.1137/S0036144500371907

[27] Goldschmidt, A. and contributors. https://pypi.org/project/derivative/

[28] Aster, R.C., Borchers, B., Thurber, C.H. (2018). *Parameter Estimation and Inverse Problems* (3rd edition). Elsevier. isbn: 9780128046517

[29] Crutchfield, J.P., McNamara, B.S. (1987). “Equations of Motion from a Data Series." *Complex Systems, 1*, 417-452.

[30] Kukreja, S.L., Löfberg, J., Brenner, M.J. (2006). “A Least Absolute Shrinkage and Selection Operator (LASSO) for Nonlinear System Identification." *IFAC Proceedings Volumes, 39*(1), 814-819. https://doi.org/10.3182/20060329-3-AU-2901.00128

[31] Wang, W.X., Yang, R., Lai, Y.C., Kovanis, V., Grebogi, C. (2011). “Predicting Catastrophes in Nonlinear Dynamical Systems by Compressive Sensing." *Physical Review Letters, 106*(15), 154101. 10.1103/PhysRevLett.106.154101

[32] Yang, R., Lai, Y.C., Grebogi, C. (2012). “Forecasting the Future: Is It Possible for Adiabatically Time-Varying Nonlinear Dynamical Systems?" *Chaos 22*, 033119. https://doi.org/10.1063/1.4740057

[33] Mangan, N.M., Kutz, J.N., Brunton, S.L., Proctor, J.L. (2017). “Model Selection for Dynamical Systems via Sparse Regression and Information Criteria." *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 473*(2204), 20170009. 10.1098/rspa.2017.0009

[34] Schaeffer, H., McCalla, S.G. (2017). “Sparse Model Selection via Integral Terms." *Physical Review E, 96*(2), 023302. 10.1103/PhysRevE.96.023302

[35] Boninsegna, L., Nüske, F., Clementi, C. (2018). “Sparse Learning of Stochastic Dynamical Equations." *The Journal of Chemical Physics, 148*(24), 241723. 10.1063/1.5018409

[36] Zhang, L., Schaeffer, H. (2019). “On the Convergence of the SINDy Algorithm." *Multiscale Modeling & Simulation, 17*(3), 948-972. 10.1137/18M1189828

[37] Lai, Y.C. (2021). “Finding Nonlinear System Equations and Complex Network Structures from Data: A Sparse Optimization Approach." Chaos: An Interdisciplinary Journal of Nonlinear Science, 31(8), 082101. 10.1063/5.0062042

[38] Abdullah, F., Wu, Z., Christofides, P.D. (2022). “Handling Noisy Data in Sparse Model Identification Using Subsampling and Co-teaching." *Computers & Chemical Engineering, 157*, 107628. https://doi.org/10.1016/j.compchemeng.2021.107628

[39] Kaheman, K., Brunton, S.L., Kutz, J.N. (2022). “Automatic Differentiation to Simultaneously Identify Nonlinear Dynamics and Extract Noise Probability Distributions from Data." *Machine Learning: Science and Technology, 3*(1), 015031. https://doi.org/10.1088/2632-2153/ac567a

[40] Messenger, D.A., Bortz, D.M. (2021). “Weak SINDy: Galerkin-Based Data-Driven Model Selection." *Multiscale Modeling & simulation, 19*(3), 1474-1497. 10.1137/20M1343166

[41] Daniels, B.C., Nemenman, I. (2015). “Automated Adaptive Inference of Phenomenological Dynamical Models." *Nature Communications, 6*, 8133. https://doi.org/10.1038/ncomms9133

[42] Bakhtiarnia, A., Fahim, A., Miandoab, E.M. (2021). “Parameter Identification of Complex Network Dynamics." *Nonlinear Dynamics, 104*, 3991-4005. https://doi.org/10.1007/s11071-021-06482-4

[43] Bramburger, J.J., Dylewsky, D., Kutz, J.N. (2020). “Sparse Identification of Slow Timescale Dynamics." *Physical Review E, 102*(2), 022204. 10.1103/PhysRevE.102.022204

[44] Tibshirani, R. (1996). “Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society, 58*(1), 267-288.

[45] Meurer, A, Smith, C.P., Paprocki, M., Čertík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., Rathnayake, T., Vig, S., Granger, B.E., Muller, R.P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M.J., Terrel, A.R., Roučka, Š, Saboo, A., Fernando, I., Kulal, S., Cimrman, R., Scopatz, A. (2017). “SymPy: Symbolic Computing in Python." *PeerJ Computer Science, 3*, e103. https://doi.org/10.7717/peerj-cs.103

[46] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P. Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y. Moore, E.W., VanderPlas, J., Laxalde, D.,Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. (2020). “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods, 17*, 261-272. 10.1038/s41592-019-0686-2

[47] Tsonis, A.A., Swanson, K.L., Roebber, P.J. (2006). “What Do Networks Have to Do with Climate?" *Bulletin of the American Meteorological Society, 87(5)*, 585-596. https://doi.org/10.1175/BAMS-87-5-585

[48] Malik, N., Bookhagen, B., Marwan, N., Kurths, J. (2012). “Analysis of Spatial and Temporal Extreme Monsoonal Rainfall Over South Asia Using Complex Networks." *Climate Dynamics, 39*, 971–987. https://doi.org/10.1007/s00382-011-1156-4

[49] Li, J., Song, J., Tan, N. Cao, C., Du, M., Xu, S., Wu, Y. (2021). “Channel Block of the Astrocyte Network Connections Accounting for the Dynamical Transition of Epileptic Seizures." *Nonlinear Dynamics, 105*, 3571–3583. https://doi.org/10.1007/s11071-021-06737-0

[50] Zou, H., Hastie, T. (2005). “Regularization and Variable Selection Via the Elastic Net." *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67*, 301-320. https://doi.org/10.1111/j.1467-9868.2005.00503.x