

DETERMINATION OF THE PARAMETERS IN LOTKA-VOLTERRA EQUATIONS FROM POPULATION MEASUREMENTS—ALGORITHMS AND NUMERICAL EXPERIMENTS*

BENJAMIN HONERAY LEE[†]

Abstract. The Lotka-Volterra model is a system of differential equations often used to predict changes in populations of organisms in an ecosystem over time. The Lotka-Volterra model is dependent on parameters such as growth rate and species-species interactions. These parameters in turn can determine certain population dynamics such as cyclical behavior over time or even extinction of one or more species. This means that determining parameters from population measurements can provide useful predictions on how populations will change in the future. In this work, we study models that exhibit (i) stable equilibrium, (ii) limit cycles, (iii) extinction, (iv) chaos. Our goal is to understand how well the parameters in the models can be determined from population data. The approach we take is to reduce the problem to that of linear regression by estimating the time rate-of-change of the populations from data. Since the regression problem can be ill-conditioned, we consider regularization strategies to ensure stability in the parameter estimation even when there is noise in the data. Numerical experiments are conducted to gain further insights into the parameter estimation problem in the four types of behavior.

1. Introduction. The Lotka-Volterra (LV) model is often used to simulate and predict population dynamics of organisms within an ecosystem [1][2]. For example, in the field of bacteriotherapy, predicting population dynamics can help prevent extinction of beneficial microbiota [3][4]. On the other hand, miscalculating the changes in a patient’s gut microbiota can lead to disease [5][6].

The LV model has several parameters that characterize population behaviors such as growth rate, carrying capacity, and competition among species [7][8]. These parameters allow the LV model to predict how populations will change over time. For example, certain parameters can characterize population dynamics of three or more species over time as either: *stable equilibrium*, where each species population reaches and stays at a certain nonzero value; *limit cycle*, where the population of each species cycles over time; *extinction of one species*, where one population is reduced to zero; *extinction of two species*, where two populations are reduced to zero; or, *chaotic dynamics*, where the populations of at least four species fluctuate independently in an unpredictable pattern.

The goal of this investigation is to examine how the parameters in the LV model can be determined from measured population data over time. In particular, we wish to quantify how robustly one can recover the model parameters in different behavioral regimes. We examine two methods of obtaining the parameters necessary for the LV system to function properly: truncated singular value decomposition (tSVD) and LASSO regression [9][10][11].

These two approaches have been examined in the literature, most notably by Varah [12], who describes a spline least squares method for estimating parameters of ordinary differential equations (ODEs). While our approach similarly handles noise in the population data and approximates the rate of change in population data using cubic splines, we take advantage of technological advances since the date of that publication in 1982. Instead of relying on the human eye and basic interactive graphics to subjectively position the knots of the cubic spline, we solve each cubic polynomial for its coefficients in MATLAB (version R2020a). Also, instead of examining predator-prey interactions, we examine multiple species competing for the same limited resource. Specifically, we simulate the interactions between three species to examine stable equilibrium, limit cycle behavior, and extinction, and between four species to examine chaotic dynamics. We also study the use of sparsity promoting regularization, i.e. incorporating an ℓ_1 penalty. Such an approach is known as LASSO in the statistics literature [9][10].

This paper is organized as follows. In Section 2, we describe the LV system and examine the different behaviors it exhibits depending on the parameters. In Section 3, we show how the inverse problem of parameter determination can be cast as a linear regression problem. We also describe an intriguing observation we call *noise insemination* that affects the regression problem. In Section 4, we first perform numerical studies to demonstrate how the inverse problem can become ill-conditioned. After characterizing the conditioning of the regression problem, we then present the results of numerical experiments in parameter recovery with and without the presence of noise using two different approaches. Lastly, in Section 5, we discuss our findings and prospects for future research.

[†]FADIL SANTOSA, FACULTY SPONSOR

*Submitted to the editors 27 November 2020.

2. The Lotka-Volterra system. The Lotka-Volterra model is expressed as

$$(2.1) \quad \frac{dy_i}{dt} = \alpha_i y_i + \sum_{j=1}^L \beta_{ij} y_i y_j$$

where L is the number of species competing for a common resource ($L \geq 3$) and $y_i(t)$ is the population of species i at time t . The parameter α_i is the intrinsic growth rate of species i , while β_{ii} controls the carrying capacity of the ecosystem if only species i exists. The cross terms, β_{ij} for $i \neq j$ are negative and represent the competition between species i and species j . Although not within the scope of this paper, nondiagonal β_{ij} can also be positive to simulate cooperative or mutualistic relationships [13]. We can think of the model parameter α as a column vector with L elements α_i , and parameter β as an L -by- L matrix with L^2 elements β_{ij} .

In the inverse problem, we are given noisy measurements of the population at times t_j , $j = 1, \dots, N$ (i.e., $y_i(t_j)$). The goal is to determine the unknown parameters α_i and β_{ij} . For the case of $L = 3$ species, we will have 3 α 's ($\alpha_1, \alpha_2, \alpha_3$) and 9 β 's ($\beta_{11}, \beta_{12}, \dots, \beta_{33}$), resulting in a total of 12 unknowns. To ensure that the LV system is not underdetermined, we will use at least 7 time points to construct the system, for a total of at least 21 equations for 12 unknowns.

Specific α and β result in different LV model population behaviors or dynamics, and we describe them in more detail below.

2.1. Stable equilibrium. Depending on α and β , each species population can reach a specific nonzero value over time. When this happens for every species in the LV model, the model is said to reach *stable equilibrium*. In this case, all species in the system coexist with one another despite competition for a common resource. For example, if the parameters are

$$\alpha = \begin{bmatrix} 3 \\ 4 \\ 7.2 \end{bmatrix}, \quad \beta = \begin{bmatrix} -2 & -1 & 0 \\ 0 & -1 & -2 \\ -2.6 & -1.6 & -3 \end{bmatrix},$$

then each population reaches stable equilibrium (Fig. 2.1).

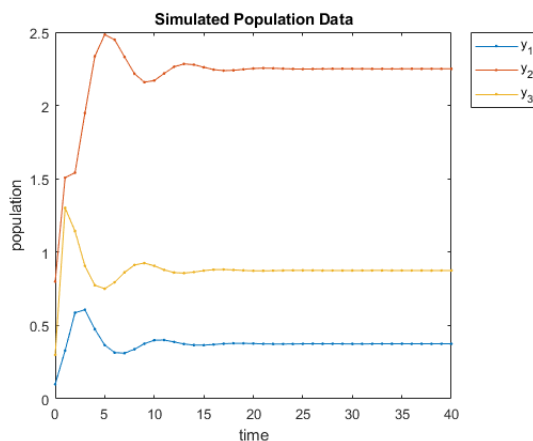


Fig. 2.1: **Stable Equilibrium Behavior.** Simulated population data $y_1(t)$ (blue —), $y_2(t)$ (red —), $y_3(t)$ (yellow —) is plotted against time $t = [0 : 1 : 40]$. The initial condition is $y(0) = [0.1 \ 0.8 \ 0.3]^T$. The population is measured in nondimensional units, so the vertical axis can be multiples of 100, 1000, etc.

2.2. Limit cycle. After a certain amount of transient behavior, the population of each species can become periodic over time. In this case, the LV model displays *limit cycle* behavior, where competition between different species results in each population fluctuating cyclically over time. This behavior was once thought to only be possible if the LV model contained at least three species (i.e. $L \geq 3$) [14]. However,

recent research has demonstrated that LV models with two species can also display limit cycle behavior [15]. As an example of a limit cycle, if the parameters are

$$\alpha = \begin{bmatrix} 3 \\ 4 \\ 7.2 \end{bmatrix}, \quad \beta = \begin{bmatrix} -0.5 & -1 & 0 \\ 0 & -1 & -2 \\ -2.6 & -1.6 & -3 \end{bmatrix},$$

then we get a limit cycle (Fig. 2.2).

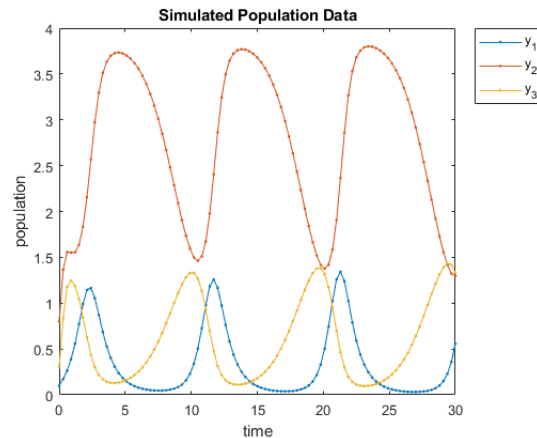


Fig. 2.2: **Limit Cycle Behavior.** Simulated population data $y_1(t)$ (blue —), $y_2(t)$ (red —), $y_3(t)$ (yellow —) is plotted against time $t = [0 : 0.3 : 30]$. The initial condition is $y(0) = [0.1 \ 0.8 \ 0.3]^T$.

2.3. Extinction of one or more species. In this scenario, one or more species populations become(s) zero, reflecting *extinction of one or more species* in the LV model. When

$$\alpha = \begin{bmatrix} 3 \\ 4 \\ 7.2 \end{bmatrix}, \quad \beta = \begin{bmatrix} -2 & -1 & -1 \\ -1 & -1 & -2 \\ -2.6 & -1.6 & -3 \end{bmatrix},$$

we have extinction of species y_2 (Fig. 2.3).

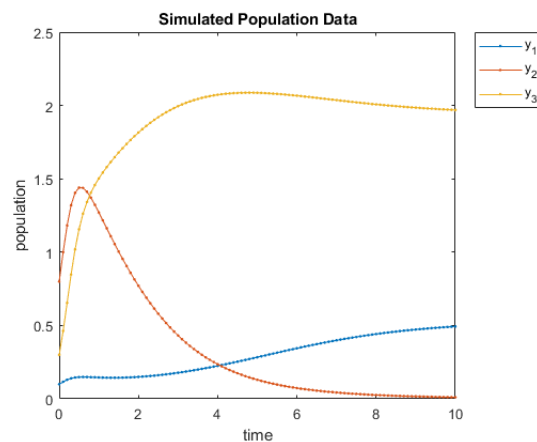


Fig. 2.3: **Extinction of One Species.** Simulated population data $y_1(t)$ (blue —), $y_2(t)$ (red —), $y_3(t)$ (yellow —) is plotted against time $t = [0 : 0.1 : 10]$. The initial condition is $y(0) = [0.1 \ 0.8 \ 0.3]^T$.

On the other hand, when

$$\alpha = \begin{bmatrix} 3 \\ 1 \\ 7.2 \end{bmatrix}, \quad \beta = \begin{bmatrix} -0.1 & -1 & -0.1 \\ -1 & -0.1 & -2 \\ -2.6 & -0.6 & -3 \end{bmatrix},$$

we have extinction of two species (y_2 and y_3) as shown in Fig. 2.4.

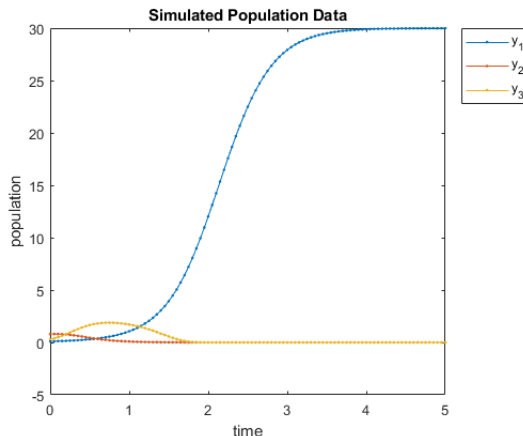


Fig. 2.4: **Extinction of Two Species.** Simulated population data $y_1(t)$ (blue —), $y_2(t)$ (red —), $y_3(t)$ (yellow —) is plotted against time $t = [0 : 0.05 : 5]$. The initial condition is $y(0) = [0.1 \ 0.8 \ 0.3]^T$.

2.4. Chaotic dynamics. When each population in the system wildly fluctuates without any discernible pattern, the LV model is said to display *chaotic dynamics*. This behavior is only possible if the LV model contains at least four species (i.e. $L \geq 4$). Parameters that result in this behavior have been investigated by Vano et al [16], and we use

$$\alpha = \begin{bmatrix} 1 \\ 0.72 \\ 1.53 \\ 1.27 \end{bmatrix}, \quad \beta = \begin{bmatrix} -1 & -1.09 & -1.52 & 0 \\ 0 & -0.72 & -0.3168 & -0.9792 \\ -3.5649 & 0 & -1.53 & -0.7191 \\ -1.5367 & -0.6477 & -0.4445 & -1.27 \end{bmatrix}.$$

This behavior is illustrated in Fig. 2.5.

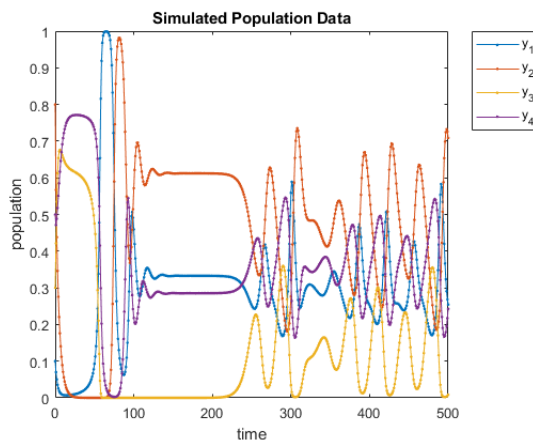


Fig. 2.5: **Chaotic Dynamics.** Simulated population data $y_1(t)$ (blue —), $y_2(t)$ (red —), $y_3(t)$ (yellow —), $y_4(t)$ (purple —) is plotted against time $t = [0 : 1 : 500]$. The initial condition is $y(0) = [0.1 \ 0.8 \ 0.3 \ 0.5]^T$.

3. Inverting for model parameters. We now wish to solve the inverse problem posed by the LV model and population measurements. Suppose we have L species and the population data set consists of

$$y_i(t_j) \text{ for } i = 1, \dots, L, \quad j = 1, \dots, N.$$

To turn the inverse problem into a regression problem, we use the data to define a matrix A of size (NL) -by- $(L + L^2)$, given by

$$(3.1) \quad A = \begin{bmatrix} \text{diag}(y(t_0)) & y_1(t_0) \text{diag}(y(t_0)) & y_2(t_0) \text{diag}(y(t_0)) & \dots & y_L(t_0) \text{diag}(y(t_0)) \\ \text{diag}(y(t_1)) & y_1(t_1) \text{diag}(y(t_1)) & y_2(t_1) \text{diag}(y(t_1)) & \dots & y_L(t_1) \text{diag}(y(t_1)) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \text{diag}(y(t_N)) & y_1(t_N) \text{diag}(y(t_N)) & y_2(t_N) \text{diag}(y(t_N)) & \dots & y_L(t_N) \text{diag}(y(t_N)) \end{bmatrix},$$

where

$$\text{diag}(y(t_j)) = \begin{bmatrix} y_1(t_j) & 0 & \dots & 0 \\ 0 & y_2(t_j) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & y_L(t_j) \end{bmatrix}$$

We let \mathbf{w} represent the time rate-of-change of the population, i.e.

$$(3.2) \quad \mathbf{w} = \frac{d}{dt} [y_1(t_0) \quad y_2(t_0) \quad \dots \quad y_L(t_0) \quad \dots \quad y_1(t_N) \quad y_2(t_N) \quad \dots \quad y_L(t_N)]^T.$$

We now represent the unknown parameters as an $L + L^2$ vector \mathbf{x} , where

$$\mathbf{x} = [\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_L \quad \beta_{11} \quad \beta_{21} \quad \dots \quad \beta_{LL}]^T.$$

The LV system implies that

$$A\mathbf{x} = \mathbf{w}.$$

This is a linear system with $L + L^2$ unknowns and NL equations. When $NL > L + L^2$, the system is overdetermined. The inverse problem now amounts to solving the linear system $A\mathbf{x} = \mathbf{w}$. When \mathbf{w} is in the range of A , there is a solution to the equation.

Since the vector \mathbf{w} is not observed and we do not have access to $y_i(t_j)$ as a function of time, rather its samples at specific time points, we need to estimate \mathbf{w} from these samples. To this end, we use cubic spline interpolation to estimate $y_i(t_j)$ and differentiate the function to approximate \mathbf{w} . However, measurements of $y_i(t_j)$ in practice often contain noise. This “seed of noise” then propagates through all variables derived from $y_i(t_j)$, in this case A and \mathbf{w} . We call this phenomenon *noise insemination*.

The definitions for the elements of A and \mathbf{w} are unchanged, but noise insemination has now polluted them. To account for this, let us use A_g to denote the noise-inseminated A and \mathbf{w}_g to denote the noise-inseminated \mathbf{w} . A_g and \mathbf{w}_g , while still derived in the same way as A and \mathbf{w} , will now be inseminated with noise from $y_i(t_j)$. In light of noise insemination, the inverse problem now becomes that of estimating \mathbf{x} in

$$(3.3) \quad A_g \mathbf{x} \approx \mathbf{w}_g$$

We do not expect \mathbf{w}_g to be in the range of A_g . Therefore, we will approach the solution using least-squares. This complicated issue and its effects on parameter recovery will be studied in numerical experiments in Section 4.

3.1. Inversion methods. We solve the inverse-problem described above using two different methods. We use truncated singular value decomposition (tSVD) to solve (3.3) to give us control over the ill-conditioning. We also examine a second method based on adding an ℓ_1 -penalty. The ℓ_1 -penalty term involves only the β parameters, whose sparsity is promoted by this method.

3.1.1. Truncated SVD. One method to solve (3.3) is to use a tSVD. We first write the SVD of the matrix A

$$A_g = U D V^T.$$

Here, U and V are NL -by- NL and $(L+L^2)$ -by- $(L+L^2)$ orthonormal matrices whose columns $\{u_i\}$ and $\{v_i\}$ span the column and row spaces of matrix A_g , respectively. The diagonal matrix D is of size NL -by- $(L+L^2)$ and its entries are the singular values of A in decreasing order. Let us denote the diagonal entries of D by d_j , $j = 1, \dots, (L+L^2)$. Then the truncated SVD solution x^\dagger [17] is given by

$$x^\dagger = \sum_{j=1}^p \frac{u_j^T \mathbf{w}_g}{d_j} v_j,$$

where p is the index value used to truncate U , D , and V . The number p must be chosen such that d_1/d_p , the condition number of the inverse of the tSVD, is not too large. Thus, p is the first j whose diagonal entry d_j/d_1 is less than the threshold value 0.001, which was obtained by trial and error. Since we know the ground truth value of the parameters, we can adjust the threshold to balance accuracy and sensitivity. Indeed, one can view this value as a tuning parameter which in practice can be determined if the size of the measurement errors is known.

3.1.2. ℓ_1 penalized least-squares. Another inversion method we employ is the LASSO regression [10], which roughly amounts to regularizing the least-squares solution of (3.3) with an ℓ_1 penalty. That is, we will solve for \mathbf{x} in

$$\min_{\mathbf{x}} \|A_g \mathbf{x} - \mathbf{w}_g\|^2 + \lambda \|P \mathbf{x}\|_1, \quad \lambda > 0,$$

where P is a diagonal matrix with entries 0 for elements of \mathbf{x} corresponding to α , and 1 otherwise (i.e. we only wish to promote sparsity for the β parameters).

We let J denote the cost function

$$J = \|A_g \mathbf{x} - \mathbf{w}_g\|^2 + \lambda \|P \mathbf{x}\|_1.$$

Since J is not differentiable everywhere, we approximate the gradient using the following formula

$$\nabla J = 2A_g^T(A\mathbf{x} - \mathbf{w}_g) + P \left[\frac{\lambda \mathbf{x}_j}{|\mathbf{x}_j + \tau|} \right],$$

where $[a_j]$ is a column vector of size (NL) with elements a_j . The parameter τ will be chosen to be small as it roughly represents the amount by which we ‘round’ the corners of the gradient at locations where it is not differentiable.

We first find an initial guess \mathbf{x} and then use the resulting cost and gradient to iteratively update \mathbf{x} in such a way that minimizes the cost function J . Since A_g and \mathbf{w}_g are known, we determine our initial \mathbf{x} by evaluating $(A_g^T A_g)^{-1} A_g^T \mathbf{w}_g$.

We next use steepest descent to update \mathbf{x} in such a way that decreases J [18][19]. We define $p = -\nabla J$ and an arbitrary initial step length f to iteratively calculate

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{prev}} + fp.$$

We then use \mathbf{x}_{new} to determine J_{new} and ∇J_{new} . Calculating J_{new} and ∇J_{new} allows us to use the sufficient decrease condition (SDC) to determine if the change in \mathbf{x} results in sufficient decrease in J . The SDC is as follows:

$$J_{\text{new}} \leq J_{\text{prev}} - cfp^T p, \quad c \in (0, 1)$$

If this condition is satisfied, we then repeat the above steps using \mathbf{x}_{new} , J_{new} , and ∇J_{new} . If the SDC is not satisfied, then we begin backtracking. Backtracking is the iterative process inside the steepest descent algorithm that slowly decreases the step length f by a scalar ρ as shown below:

$$f = f \rho, \quad \rho \in (0, 1).$$

This new step length is then used to redefine \mathbf{x}_{new} , J_{new} , and ∇J_{new} . This process is repeated until the SDC is satisfied, at which point we continue steepest descent. The stopping criterion we adopted is to stop when the gradient is sufficiently small or when an arbitrary large number of steepest descent steps K has been taken.

4. Numerical experiments.

4.1. **Data generation.** We first construct our simulated population data $y_i(t_j)$ by solving the ODE in (2.1) with a given α and β designed to exhibit one of the five previously mentioned population behaviors. To facilitate solving for L species, we convert the LV model ODE into matrix format.

$$\frac{d}{dt} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_L \end{bmatrix} = \begin{bmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha_L \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_L \end{bmatrix} + \begin{bmatrix} y_1 & 0 & \dots & 0 \\ 0 & y_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & y_L \end{bmatrix} \begin{bmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1L} \\ \beta_{21} & \beta_{22} & \dots & \beta_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{L1} & \beta_{L2} & \dots & \beta_{LL} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_L \end{bmatrix}$$

To solve the ODE, we use MATLAB’s `ode45` function, sampling the solution at regular intervals Δt and choosing an initial condition based on the specific population dynamic being examined. Solving the ODE above generates the simulated population data $y_i(t_j)$, and we can now begin to use the previously described algorithms to attempt to recover the parameters α and β .

4.2. **Reconstruction results using tSVD.** In this section, we first study the condition number of A_g and the estimated time-derivative \mathbf{w}_g as a function of sampling rate when solving for \mathbf{x} . Next, we implement the tSVD approach and study its efficacy in parameter recovery from population data. We will consider different dynamics with both noiseless and noisy data.

In these experiments, the “ground truth” variables associated with the population data $y_i(t_j)$ are denoted with ‘*’, and the variables recovered from the inversion are denoted with ‘g’ for “guess.” For example, the true time-derivative \mathbf{w} will now be \mathbf{w}_* , while the estimated time-derivative using cubic spline interpolation will be \mathbf{w}_g .

4.2.1. **Condition number and estimated time-derivative.** Using α_* and β_* (where “*” represents “ground truth” in Section 4.2) for stable equilibrium from Section 2.1, we generate data $y_i(t_j)$ using MATLAB’s ODE solver as described in Section 4.1. $y_i(t_j)$ is generated over the time window $[0, 7]$ with different sampling rates. No noise is added to $y_i(t_j)$. After forming matrix A_g from $y_i(t_j)$ using (3.1), we evaluate A_g ’s condition number using MATLAB’s `cond` function. We use cubic spline interpolation to estimate \mathbf{w}_* from population data $y_i(t_j)$ using (3.2). The estimated derivative vector is denoted by \mathbf{w}_g . We know that the true time-derivative of $y_i(t_j)$ at the time samples t_j is $\mathbf{w}_* = A \mathbf{x}_*$. We measure the error by calculating

$$e_w = \|\mathbf{w}_* - \mathbf{w}_g\| / \|\mathbf{w}_*\|$$

using the ℓ_2 norm.

We determine the condition number and e_w at different sampling rates for stable equilibrium, limit cycle behavior, extinction of one species, extinction of two species, and chaotic dynamics (Table 4.1).

The overall trend we observe for the well-conditioned population dynamics in Table 4.1(a) (stable equilibrium, limit cycle, extinction of one species), is that as Δt decreases, the accuracy of \mathbf{w}_g increases (i.e. e_w decreases), but the condition number worsens (increases). Exceptions to this trend occur at $\Delta t = 1$ to 0.5 for stable equilibrium, where e_w increases, and at $\Delta t = 1$ to 0.5 for extinction of one species, where e_w increases and condition number decreases. These exceptions are believed to be due to a quirk or accident in the cubic spline interpolation, but the overall trend is maintained for the well-conditioned population dynamics.

Like the well-conditioned dynamics, the overall trend we observe for the ill-conditioned population dynamics in Table 4.1(b) (extinction of two species, chaotic dynamics) is that as Δt decreases, the accuracy of \mathbf{w}_g increases. However, for the ill-conditioned dynamics, the condition number generally improves (decreases) with decreasing Δt . We note that the condition numbers for extinction of two species and chaotic dynamics are at least two orders of magnitude greater than those of the well-conditioned population dynamics in Table 4.1(a). This ill-conditioning suggests that extinction of two species and chaotic dynamics will be much more sensitive to noise than the well-conditioned behaviors. We anticipate this will make parameter recovery from noisy population data slightly less accurate.

	Population Dynamic		Δt			
			1	0.5	0.25	0.125
(a)	Stable Equilibrium	CN	256.81	340.51	383.88	412.04
		e_w	0.36084	0.41734	0.15009	0.012521
	Limit Cycle	CN	494.30	657.68	729.68	775.93
		e_w	0.43317	0.31747	0.11172	0.0092114
	Extinction of One Species	CN	476.09	412.13	478.69	517.30
		e_w	0.34647	0.39287	0.14149	0.012599
(b)	Extinction of Two Species	CN	3.4129×10^{15}	3.39710×10^5	9.8022×10^4	1.0904×10^5
		e_w	0.41596	0.12800	0.018383	0.0042166
	Chaotic Dynamics	CN	1.0116×10^5	7.1572×10^4	7.1400×10^4	7.4593×10^4
		e_w	0.11170	0.029008	0.0052168	0.00071735

Table 4.1: **Condition Number and e_w for All Population Dynamics.** Condition number (CN) and e_w using different Δt for (a) stable equilibrium, limit cycle behavior, and extinction of one species, and for (b) extinction of two species and chaotic dynamics. The time window is $[0,7]$. The condition numbers for the last two population dynamics (b) are at least two orders of magnitude larger than the first three population dynamics (a). As generally expected, the higher the sampling rate, the more accurate the estimate \mathbf{w}_g .

These observations for Table 4.1 suggest that we must choose Δt with care in order to improve our chances of accurate recovery of model parameters from data, especially when the data are noisy or ill-conditioned.

4.2.2. Parameter recovery with tSVD for noiseless and noisy data. Consulting Table 4.1, we choose $\Delta t = 0.25$ and a time window of $[0,7]$ for all dynamics, which appears to balance conditioning and derivative accuracy. This choice gives $N = 29$, which renders the problem of determining 12 model parameters (20 for chaotic dynamics) over-determined. We will be examining both noiseless and noisy generated population data $y_i(t_j)$.

To simulate noise, we use MATLAB’s `normrnd` function to generate normally distributed random noise with a given standard deviation. If $y_i(t_j)$ are the outputs of the ODE solver, the noisy data are

$$h_i(t_j) = y_i(t_j) + \psi n_i(t_j),$$

where $n_i(t_j)$ is $\mathcal{N}(0, 1)$ and ψ is a scalar. When $\psi = 0$, the data are noiseless. We define ψ for particular noise levels $e_y \in [0, 1]$, where

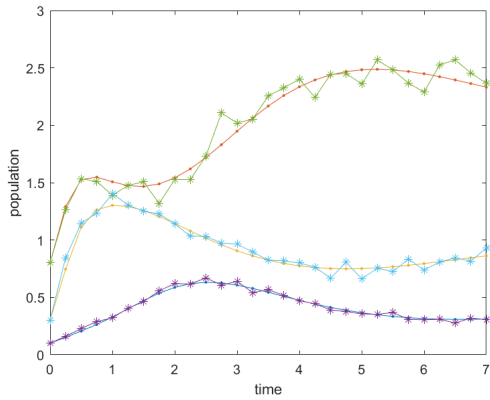
$$\begin{aligned} e_y &= \frac{\|h_i(t_j) - y_i(t_j)\|}{\|y_i(t_j)\|} \\ &= \frac{\|n_i(t_j)\|}{\|y_i(t_j)\|} \\ \Rightarrow \psi &= \frac{e_y \|y_i(t_j)\|}{\|n_i(t_j)\|}. \end{aligned}$$

In Figure 4.1 we compare noisy population data $h_i(t_j)$ using a noise level of $e_y = 5\%$ with $y_i(t_j)$.

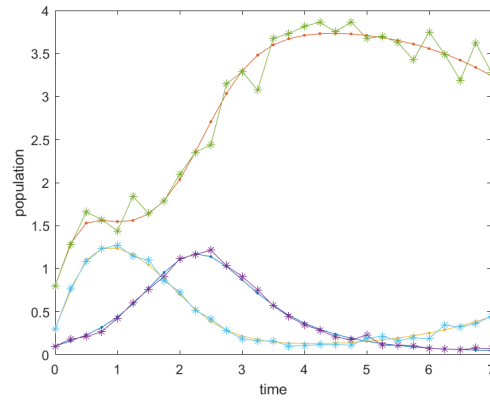
Since we expect ill-conditioned functions to be much less tolerant to noise than well-conditioned ones, we will when implementing the tSVD described in Section 3.1.1 select p so that it balances the accuracy and stability of parameter recovery based on the conditioning of the population data. Specifically, $d_p/d_1 < 0.001$. The results of our inversion for each population dynamic are summarized in Table 4.2.

To test the accuracy of α_g and β_g recovered from the inversion, we define $g_i(t_j)$ to be a new population function at times t_j , which serves as a guess for $y_i(t_j)$. We use MATLAB’s ODE solver to obtain $g_i(t_j)$ using the recovered parameters α_g, β_g , and the initial conditions for $y_i(t_j)$. We then compare $g_i(t_j)$ with the noisy data $h_i(t_j)$ for each population dynamic at different noise levels e_y (figures in Table 4.2). The ℓ_2 norm prediction error $e_g = \|h_i(t_j) - g_i(t_j)\|/\|h_i(t_j)\|$ and the parameter recovery errors $e_\alpha = \|\alpha_* - \alpha_g\|/\|\alpha_*\|$, $e_\beta = \|\beta_* - \beta_g\|/\|\beta_*\|$ are also reported in Table 4.2.

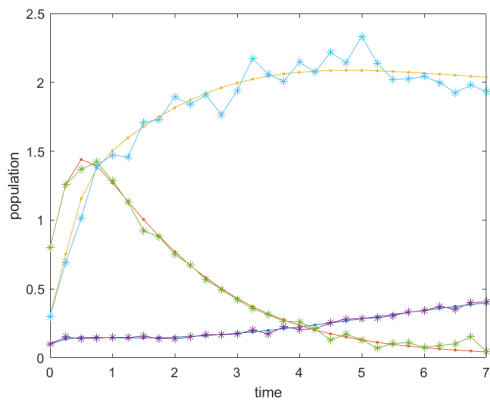
DETERMINING PARAMETERS IN LV EQUATIONS



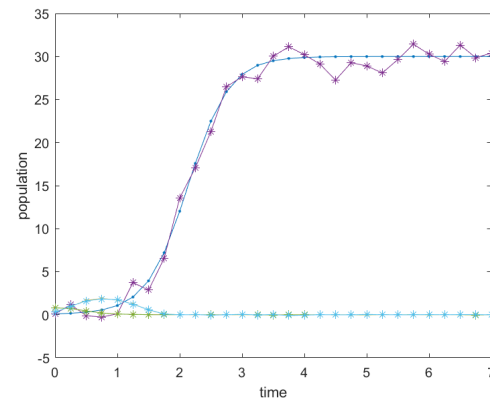
(a) Stable Equilibrium



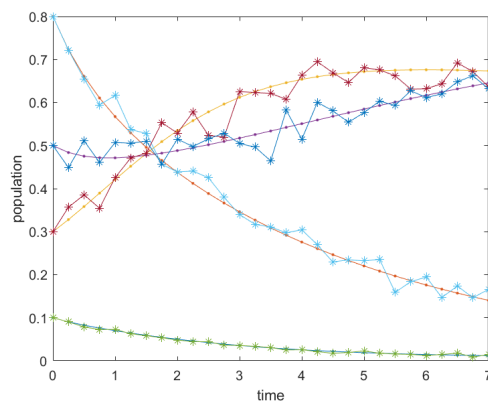
(b) Limit Cycle



(c) Extinction of One Species



(d) Extinction of Two Species



(e) Chaotic Dynamics

Fig. 4.1: **Noisy vs. Noiseless Population Data.** $y_i(t_j)$ with no noise (blue —, red —, yellow —) is plotted against $h_i(t_j)$ with $e_y = 5\%$ noise (purple —*, green —*, cyan —*) over the time window $[0,7]$ with $\Delta t = 0.25$. For chaotic dynamics, $y_i(t_j)$ with no noise (blue —, red —, yellow —, purple —) is plotted against $h_i(t_j)$ with $e_y = 5\%$ noise (green —*, cyan —*, maroon —*, and teal —*). Chaotic dynamics data is taken over the same time interval as the other four, a much smaller interval than in Fig. 2.5. This smaller time interval makes the data seem less chaotic.

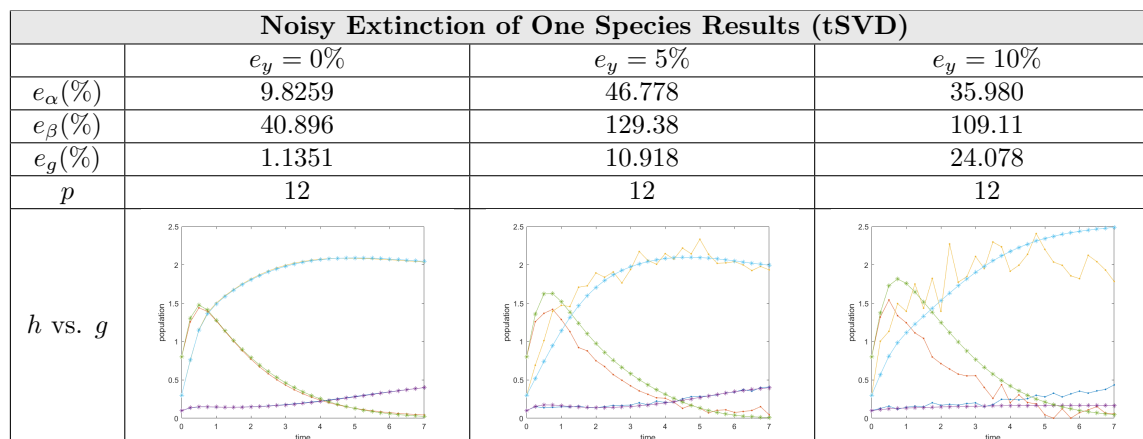
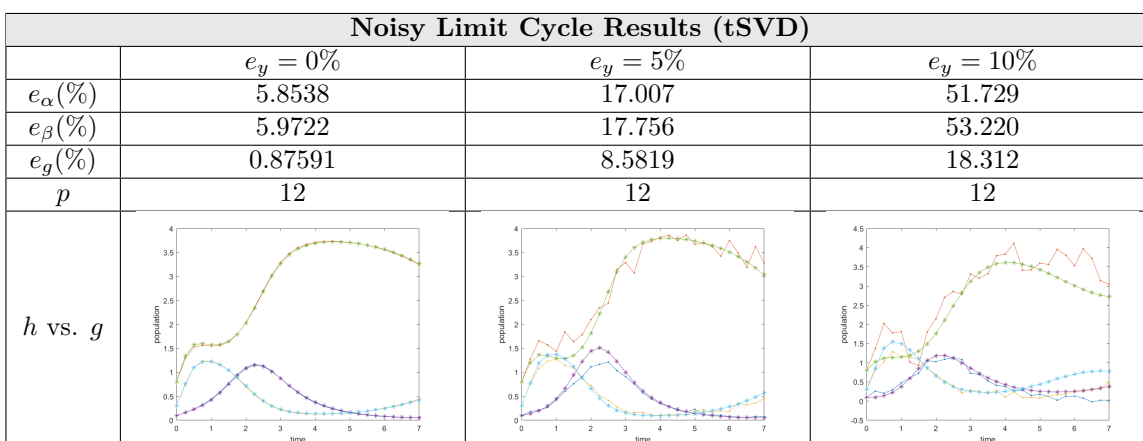
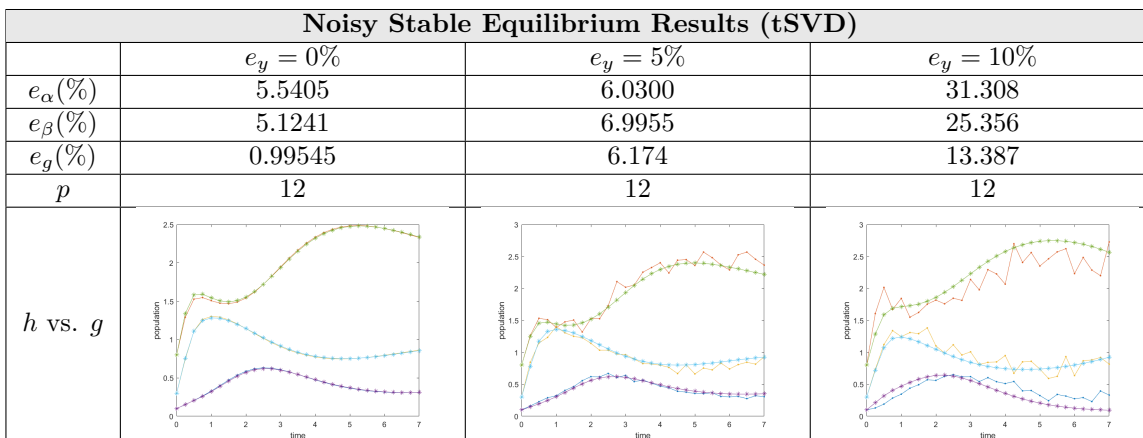
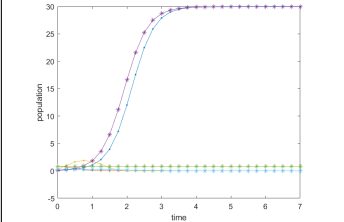
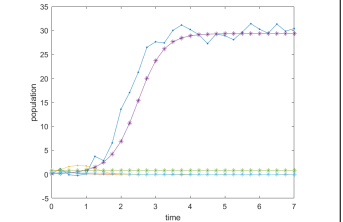
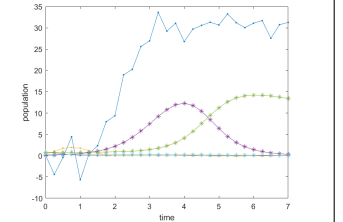


Table 4.2: **Parameter recovery with and without noise using tSVD.** Parameter errors (%) e_α , e_β , prediction error (%) e_g , truncating index p , and plots of noisy $h_i(t_j)$ vs. $g_i(t_j)$ for different e_y (% noise levels) over the time window $[0,7]$ with $\Delta t = 0.25$. $h_1(t)$ (blue —), $h_2(t)$ (red —), $h_3(t)$ (yellow —), $g_1(t)$ (purple —*), $g_2(t)$ (green —*—), $g_3(t)$ (cyan —*—) (continued)

We observe that $y_i(t_j)$ ($h_i(t_j)$ with 0% noise) always yields the best guess $g_i(t_j)$ for all five population dynamics, as expected. We expected parameter errors e_α , e_β , and prediction error e_g to increase with increasing noise levels, and this was the case for e_g across all population dynamics. This was also true for e_α

Noisy Extinction of Two Species Results (tSVD)			
	$e_y = 0\%$	$e_y = 5\%$	$e_y = 10\%$
$e_\alpha(\%)$	93.138	92.254	92.778
$e_\beta(\%)$	95.135	98.082	103.56
$e_g(\%)$	7.6601	12.255	87.961
p	4	5	6
h vs. g			

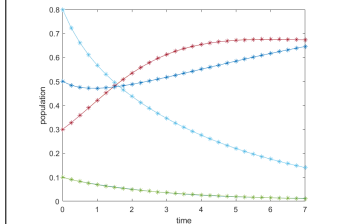
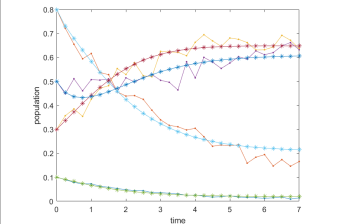
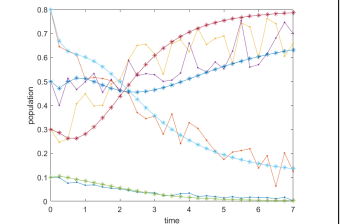
Noisy Chaotic Dynamics Results (tSVD)			
	$e_y = 0\%$	$e_y = 5\%$	$e_y = 10\%$
$e_\alpha(\%)$	50.566	152.04	355.75
$e_\beta(\%)$	82.773	646.57	1025.3
$e_g(\%)$	0.083097	6.8172	13.688
p	15	19	19
h vs. g			

Table 4.2: (*continued*) Parameter recovery with and without noise using tSVD. Parameter errors (%) e_α , e_β , prediction error (%) e_g , truncating index p , and plots of noisy $h_i(t_j)$ vs. $g_i(t_j)$ for different e_y (% noise levels) over the time window $[0,7]$ with $\Delta t = 0.25$. $h_1(t)$ (blue —), $h_2(t)$ (red —), $h_3(t)$ (yellow —), $g_1(t)$ (purple —*), $g_2(t)$ (green —*), $g_3(t)$ (cyan —*). For noisy chaotic dynamics results using tSVD, $h_1(t)$ (blue —), $h_2(t)$ (red —), $h_3(t)$ (yellow —), $h_4(t)$ (purple —), $g_1(t)$ (green —*), $g_2(t)$ (cyan —*), $g_3(t)$ (maroon —*), $g_4(t)$ (teal —*)

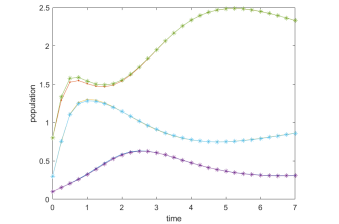
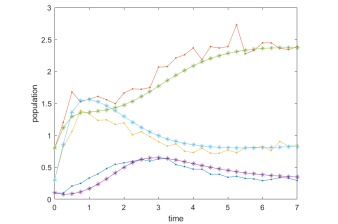
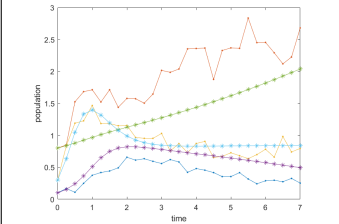
and e_β , with the exceptions of extinction of one species from 5% to 10% noise and extinction of two species from 0% to 5% noise (Table 4.2). Parameter errors e_α and e_β were similar to each other for limit cycle and extinction of two species, but this is likely an anecdotal observation. Interestingly, while the parameter errors e_α and e_β were relatively insensitive to added noise in extinction of two species, the prediction error e_g rapidly rose, culminating at $e_y = 10\%$ noise, where the resulting $g_i(t_j)$ no longer contained the correct surviving species. The parameter errors e_α and e_β fluctuated wildly with noise in chaotic dynamics (Table 4.2), likely because the additional errors increased the total number of possible guesses of α_g and β_g .

Another notable observation among the five dynamics in Table 4.2 is that e_g is always smaller than e_α and e_β . This suggests that although the α and β recovered via tSVD typically generate valid alternate models for the same population data, they are very different from their ground truth counterparts. This would explain the small error in g and large errors in α and β .

We observe that the truncation variable p was relatively insensitive to added noise for well-conditioned population dynamics but changed when adding noise to the ill-conditioned dynamics, i.e. extinction of two species and chaotic dynamics (Table 4.2). The variability of p is likely because the noise added to the population data $y_i(t_j)$ “corrupted” the matrix A_g used in the tSVD as described in Section 3.1.1. This noise in A_g in turn corrupted the matrix D of the tSVD, and for ill-conditioned population dynamics such as extinction of two species and chaotic dynamics, this noise was enough to lead to different values of p for different noise levels.

4.3. Reconstruction results using LASSO regression. In this section, we study LASSO regression as another parameter recovery technique. Similar to the tSVD experiment in Section 4.2.1, we study the condition number of A_g and the estimated time-derivative \mathbf{w}_g as a function of sampling rate when solving for \mathbf{x}_* . In this experiment, we study the impact of sparsity-promoting ℓ_1 -norm regularization of the cost function J as described in Section 3.1.2 on the same five population dynamics from Section 2. Next, we implement the LASSO regression approach on noisy versions of the five dynamics and study this approach's efficacy in parameter recovery.

4.3.1. Parameter recovery with LASSO regression for noiseless and noisy data. We now examine the effects of noise on recovering parameters α_* and β_* through LASSO regression. For the steepest descent step described in Section 3.1.2, we empirically define $K = 500$, $\lambda = 0.1$, $\tau = 0.001$, $c = 0.01$, $f = 0.05$, and $\rho = 0.7$ (because of the additional species, chaotic dynamics causes Matlab's `ode45` to crash at $\lambda = 0.1$, so we use $\lambda = 0.04$). The error parameters e_g , e_α , and e_β are defined as in Section 4.2, with α_g and β_g representing the α and β determined from LASSO regression rather than from tSVD. Because we are only changing the solving algorithm, the previously determined relationships between the condition number, time-derivative error, and sampling rate remain the same.

Noisy Stable Equilibrium Results (LASSO)																														
	$e_y = 0\%$	$e_y = 5\%$	$e_y = 10\%$																											
$e_\alpha(\%)$	4.9233	67.333	49.869																											
$e_\beta(\%)$	4.5822	60.619	49.179																											
$e_g(\%)$	0.84953	10.626	30.269																											
β_g	<table border="1"> <tr><td>-1.98</td><td>-1.02</td><td>-0.01</td></tr> <tr><td>-0.03</td><td>-1.09</td><td>-2.18</td></tr> <tr><td>-2.57</td><td>-1.66</td><td>-3.12</td></tr> </table>	-1.98	-1.02	-0.01	-0.03	-1.09	-2.18	-2.57	-1.66	-3.12	<table border="1"> <tr><td>-0.54</td><td>0.35</td><td>2.38</td></tr> <tr><td>-0.23</td><td>-0.72</td><td>-0.99</td></tr> <tr><td>-2.68</td><td>-1.84</td><td>-3.28</td></tr> </table>	-0.54	0.35	2.38	-0.23	-0.72	-0.99	-2.68	-1.84	-3.28	<table border="1"> <tr><td>-2.44</td><td>-1.00</td><td>0.54</td></tr> <tr><td>-0.09</td><td>-0.04</td><td>0.04</td></tr> <tr><td>-2.80</td><td>-1.00</td><td>-1.95</td></tr> </table>	-2.44	-1.00	0.54	-0.09	-0.04	0.04	-2.80	-1.00	-1.95
-1.98	-1.02	-0.01																												
-0.03	-1.09	-2.18																												
-2.57	-1.66	-3.12																												
-0.54	0.35	2.38																												
-0.23	-0.72	-0.99																												
-2.68	-1.84	-3.28																												
-2.44	-1.00	0.54																												
-0.09	-0.04	0.04																												
-2.80	-1.00	-1.95																												
h vs. g																														

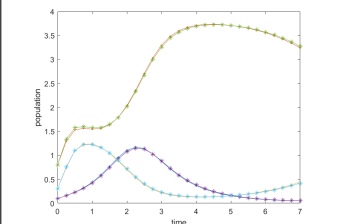
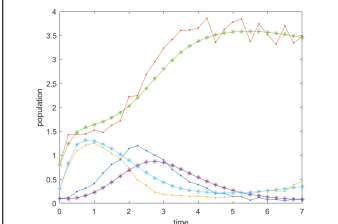
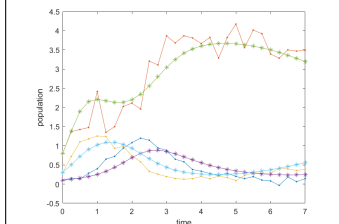
Noisy Limit Cycle Results (LASSO)																														
	$e_y = 0\%$	$e_y = 5\%$	$e_y = 10\%$																											
$e_\alpha(\%)$	5.7068	55.196	48.817																											
$e_\beta(\%)$	5.7807	54.699	49.162																											
$e_g(\%)$	0.98515	10.869	15.009																											
β_g	<table border="1"> <tr><td>-0.51</td><td>-1.00</td><td>-0.00</td></tr> <tr><td>-0.05</td><td>-1.10</td><td>-2.20</td></tr> <tr><td>-2.61</td><td>-1.68</td><td>-3.16</td></tr> </table>	-0.51	-1.00	-0.00	-0.05	-1.10	-2.20	-2.61	-1.68	-3.16	<table border="1"> <tr><td>0.22</td><td>0.12</td><td>2.20</td></tr> <tr><td>-0.15</td><td>-0.86</td><td>-1.44</td></tr> <tr><td>-2.53</td><td>-1.89</td><td>-3.69</td></tr> </table>	0.22	0.12	2.20	-0.15	-0.86	-1.44	-2.53	-1.89	-3.69	<table border="1"> <tr><td>-0.51</td><td>-0.40</td><td>1.12</td></tr> <tr><td>0.07</td><td>-0.96</td><td>-1.82</td></tr> <tr><td>-2.29</td><td>-0.58</td><td>-1.22</td></tr> </table>	-0.51	-0.40	1.12	0.07	-0.96	-1.82	-2.29	-0.58	-1.22
-0.51	-1.00	-0.00																												
-0.05	-1.10	-2.20																												
-2.61	-1.68	-3.16																												
0.22	0.12	2.20																												
-0.15	-0.86	-1.44																												
-2.53	-1.89	-3.69																												
-0.51	-0.40	1.12																												
0.07	-0.96	-1.82																												
-2.29	-0.58	-1.22																												
h vs. g																														

Table 4.3: **Parameter recovery with and without noise using LASSO.** Parameter errors (%) e_α , e_β , prediction error (%) e_g , β_g , and plots of noisy $h_i(t_j)$ vs. $g_i(t_j)$ for different e_y (% noise levels) over the time window $[0,7]$ with $\Delta t = 0.25$. $h_1(t)$ (blue ---), $h_2(t)$ (red ---), $h_3(t)$ (yellow ---), $g_1(t)$ (purple ---), $g_2(t)$ (green ---), $g_3(t)$ (cyan ---) (continued)

DETERMINING PARAMETERS IN LV EQUATIONS

Noisy Extinction of One Species Results (LASSO)			
	$e_y = 0\%$	$e_y = 5\%$	$e_y = 10\%$
$e_\alpha(\%)$	8.9173	43.866	42.178
$e_\beta(\%)$	39.036	234.09	82.546
$e_g(\%)$	1.6302	8.1967	19.418
β_g	$\begin{bmatrix} -2.06 & -1.01 & -1.10 \\ -3.13 & -1.20 & -2.11 \\ -2.68 & -1.66 & -3.09 \end{bmatrix}$	$\begin{bmatrix} 0.19 & -0.13 & -0.19 \\ -13.67 & -1.77 & -1.73 \\ -2.35 & -1.00 & -2.21 \end{bmatrix}$	$\begin{bmatrix} -2.30 & -0.78 & -0.61 \\ -5.50 & -0.98 & -1.76 \\ -2.01 & -0.57 & -1.41 \end{bmatrix}$
h vs. g			
Noisy Extinction of Two Species Results (LASSO)			
	$e_y = 0\%$	$e_y = 5\%$	$e_y = 10\%$
$e_\alpha(\%)$	38.864	136.80	138.40
$e_\beta(\%)$	69.073	200.88	208.34
$e_g(\%)$	0.95900	27.831	94.325
β_g	$\begin{bmatrix} -0.10 & -1.30 & -0.05 \\ -2.23 & -2.97 & -2.74 \\ -2.42 & -0.24 & -2.73 \end{bmatrix}$	$\begin{bmatrix} -0.09 & -1.05 & -0.81 \\ -0.17 & -1.65 & -2.88 \\ -0.04 & 7.43 & 1.00 \end{bmatrix}$	$\begin{bmatrix} -0.02 & -4.66 & 0.86 \\ -0.03 & 1.47 & -1.55 \\ -0.02 & 7.30 & 0.98 \end{bmatrix}$
h vs. g			
Noisy Chaotic Dynamics Results (LASSO)			
	$e_y = 0\%$	$e_y = 5\%$	$e_y = 10\%$
$e_\alpha(\%)$	57.869	124.55	110.01
$e_\beta(\%)$	73.365	806.01	869.61
$e_g(\%)$	8.3064	21.533	19.481
β_g	$\begin{bmatrix} 1.30 & -0.63 & -0.45 & -0.00 \\ 1.66 & -0.57 & 0.00 & -0.20 \\ -2.38 & 0.03 & -1.27 & -0.29 \\ -0.00 & -0.33 & -0.00 & -0.58 \end{bmatrix}$	$\begin{bmatrix} -24.16 & -0.00 & -2.18 & 0.00 \\ -17.48 & -0.66 & -3.53 & -0.01 \\ -32.65 & 3.21 & -2.79 & 0.00 \\ -1.79 & 0.67 & 0.41 & 0.00 \end{bmatrix}$	$\begin{bmatrix} 16.01 & -4.51 & -4.94 & 2.91 \\ 39.72 & -5.51 & 0.00 & 0.88 \\ -11.91 & 2.18 & -0.92 & 0.00 \\ -0.00 & -1.28 & -0.62 & -4.04 \end{bmatrix}$
h, g			

Table 4.3: (continued) Parameter recovery with and without noise using LASSO. Parameter errors (%) e_α , e_β , prediction error (%) e_g , β_g , and plots of noisy $h_i(t_j)$ vs. $g_i(t_j)$ for different e_y (% noise levels) over the time window [0,7] with $\Delta t = 0.25$. $h_1(t)$ (blue —), $h_2(t)$ (red —), $h_3(t)$ (yellow —), $g_1(t)$ (purple —*), $g_2(t)$ (green —*), $g_3(t)$ (cyan —*). For noisy chaotic dynamics results using tSVD, $h_1(t)$ (blue —), $h_2(t)$ (red —), $h_3(t)$ (yellow —), $h_4(t)$ (purple —), $g_1(t)$ (green —*), $g_2(t)$ (cyan —*), $g_3(t)$ (maroon —*), $g_4(t)$ (teal —*)

Similar to Section 4.2.2, we choose time window $[0,7]$ and $\Delta t = 0.25$. We will also use the same noise levels ($e_y = 0\%$, 5% , 10%) from Section 4.2.2. With LASSO regression, there is no p that is used for truncation, so we will examine β_g for each population dynamic and noise level to assess the effects of sparsity-promoting regularization. Lastly, we compare e_α , e_β , and e_g from tSVD with those from LASSO regression.

As with tSVD, $h_i(t_j)$ with 0% noise using LASSO regression always yields the best guess $g_i(t_j)$ for all five population dynamics (Table 4.3). Unlike tSVD where the prediction error e_g increased with increasing noise for all population dynamics, this relationship was not always seen in LASSO regression. For example, e_g at 5% noise was worse than at 10% noise for chaotic dynamics (Table 4.3).

$h_i(t_j)$ vs. $g_i(t_j)$ for tSVD was visually comparable to that for LASSO regression. One notable mismatch between $h_i(t_j)$ and $g_i(t_j)$ that was present in both tSVD and LASSO regression was at 10% noise for extinction of two species (Table 4.3).

While e_α and e_β nearly always increased with increasing noise as expected for tSVD, this was largely not the case in LASSO regression. The only place where this does occur was in extinction of two species (Table 4.3). For the other dynamics, both e_α and e_β are larger with 5% noise than with 10% noise.

Inspection of β_g shows that while sparsity-promoting regularization can make β_g slightly parsimonious in noiseless conditions, adding noise makes it harder to obtain a sparse β_g . Additionally, for population dynamics with non-sparse “ground truth” β such as extinction of one/two species (Table 4.3), sparsity-promoting regularization had a difficult time finding parsimonious β_g . For chaotic dynamics, although the “ground truth” β from Section 2.4 did already contain zeros, sparsity-promoting regularization found alternative parsimonious β_g for not only noiseless but also noisy conditions.

Lastly, we observe from Table 4.4 that parameter recovery using LASSO regression is typically better (i.e. lower e_α , e_β , e_g) than tSVD in noiseless conditions. On the other hand, when noise levels are increased tSVD is generally better, although LASSO regression can sometimes still be better, especially at high noise-levels such as 10% noise.

		$e_y = 0\%$		$e_y = 5\%$		$e_y = 10\%$	
		tSVD	LASSO	tSVD	LASSO	tSVD	LASSO
Stable Equilibrium	$e_\alpha(\%)$	5.5405	4.9233	6.0300	67.333	31.308	49.869
	$e_\beta(\%)$	5.1241	4.5822	6.9955	60.619	25.356	49.179
	$e_g(\%)$	0.99545	0.84953	6.174	10.626	13.387	30.269
Limit Cycle	$e_\alpha(\%)$	5.8538	5.7068	17.007	55.196	51.729	48.817
	$e_\beta(\%)$	5.9722	5.7807	17.756	54.699	53.220	49.162
	$e_g(\%)$	0.87591	0.98515	8.5819	10.869	18.312	15.009
Extinction of One Species	$e_\alpha(\%)$	9.8259	8.9173	46.778	43.866	35.980	42.178
	$e_\beta(\%)$	40.896	39.036	129.38	234.09	109.11	82.546
	$e_g(\%)$	1.1351	1.6302	10.918	8.1967	24.078	19.418
Extinction of Two species	$e_\alpha(\%)$	93.138	38.864	92.254	136.80	92.778	138.40
	$e_\beta(\%)$	95.135	69.073	98.082	200.88	103.56	208.34
	$e_g(\%)$	7.6601	0.95900	12.255	27.831	87.961	94.325
Chaotic Dynamics	$e_\alpha(\%)$	50.566	57.869	152.04	124.55	355.75	110.01
	$e_\beta(\%)$	82.773	73.365	646.57	806.01	1025.3	869.61
	$e_g(\%)$	0.083097	8.3064	6.8172	21.533	13.688	19.481

Table 4.4: **tSVD vs. LASSO for Noisy Population Dynamics.** Comparison of parameter errors e_α , e_β , and prediction error e_g , for different noise levels e_y and different algorithms (tSVD vs. LASSO) over the time window $[0,7]$ with $\Delta t = 0.25$. For each tSVD vs LASSO comparison, the smaller error is bolded to indicate which algorithm was more successful.

5. Discussion. The inverse problem of recovering parameters from population data constructed from the Lotka-Volterra model is applicable in many fields such as bacteriotherapy, where predicting extinction or growth of certain gut microbiota can help prevent disease. Investigating how to solve this inverse problem

provides useful insight into how different types of population dynamics and other factors such as conditioning and noise can affect parameter recovery. Two methods of parameter recovery were explored: truncated SVD and LASSO regression.

5.1. tSVD. Truncated SVD for parameter recovery from population data was relatively quick and straightforward. As anticipated for all five population dynamics, the best guess of the population data occurred when no noise was added, and the prediction error increased with increasing noise. This could not be said, however, for the parameter errors, which showed no definite pattern of behavior. The parameter errors were generally larger than their corresponding prediction error; this suggests that using tSVD for parameter recovery results in multiple alternate LV model parameters that could lead to the same population data.

The idea of noise “inseminating” the components of a tSVD calculation was a particularly appealing explanation of the challenges we faced when determining the truncation index and variables in the tSVD that are typically considered as noiseless in classical regression. A more in-depth understanding of this could be a topic of future work.

The condition numbers of extinction of two species and chaotic dynamics were appreciably higher than the more well-conditioned population dynamics (stable equilibrium, limit cycle, and extinction of one species). While the ill-conditioned population dynamics did generally have larger parameter and prediction errors than the well-conditioned dynamics, there were a few exceptions. This may be in part related to the selection of sampling rate which was determined empirically.

Our experiments using tSVD on different population dynamics revealed a relationship between the condition number, time-derivative estimation, and sampling rate that may be useful to keep in mind when determining/predicting the accuracy of parameter recovery in future experiments. Namely, for well-conditioned dynamics, as sampling rate increases, the condition number worsens and the time-derivative accuracy improves. But, for ill-conditioned dynamics, as sampling rate increases, both the condition number and time-derivative accuracy improve.

5.2. LASSO. LASSO regression is the other technique we studied for parameter recovery. Here, we investigated the efficacy of sparsity-promoting regularization on β parameter recovery via LASSO regression. While LASSO regression’s ℓ_1 -norm regularization and backtracking with steepest descent algorithms were slower and relatively more complex than tSVD, LASSO regression generally decreased parameter errors for noiseless population data compared to tSVD across all population dynamics except chaotic dynamics. However, neither tSVD nor LASSO regression was obviously better when examining prediction error.

While the sparsity-promoting regularization in LASSO regression did make β_g slightly parsimonious in noiseless conditions, a satisfactorily sparse β_g was difficult to elucidate. Factors such as added noise and non-sparse “ground truth” β tended to make sparsity-promoting regularization less effective.

5.3. Effects of Noise. When using tSVD, there is a direct relationship between noise levels and parameter/prediction errors across all five population dynamics. On the other hand, parameter recovery using LASSO regression does not have this overarching trend. Rather, LASSO regression has overall smaller parameter errors in noiseless conditions, with tSVD performing increasingly better (i.e. smaller parameter/prediction errors) than LASSO regression as noise is added. At 10% noise, both tSVD and LASSO have approximately equal chances of obtaining the smaller error values. Lastly, the parameters in α are generally less affected by noise than the parameters in β , and this is clearer as the conditioning of the population dynamic worsens.

5.4. tSVD vs. LASSO. Comparing the two parameter recovery methods, tSVD typically finds combinations of LV model parameters that are vastly different from the ground truth parameters but still generate the same population data, while LASSO regression uses sparsity-promoting regularization to generate slightly more parsimonious LV model parameter guesses. From Table 4.4, it seems that LASSO regression is typically better than tSVD in noiseless conditions. On the other hand, when noise levels are increased tSVD is generally better, although LASSO regression can sometimes still be better, especially at high noise-levels such as 10% noise.

Lastly, despite sometimes not having as much accuracy as tSVD, LASSO regression has the greater potential to more accurately recover parameters and predict population changes. During our experiments, we arbitrarily set the maximum number of steepest descent iterations to 500. Given the iterative nature

of the steepest descent with backtracking algorithm, LASSO regression with more iterations (e.g. 500,000) could conceivably yield much more accurate results. The main drawback is that with the steepest descent algorithm described, this would likely take much more time than tSVD. We are aware that there are efficient LASSO algorithms in the public domain that could compete favorably with tSVD. An important part of this research is the implementation of the regularized steepest descent with backtracking, which we opted for solving the LASSO regression.

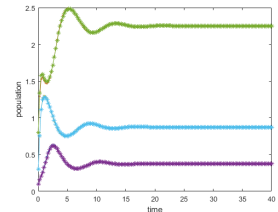
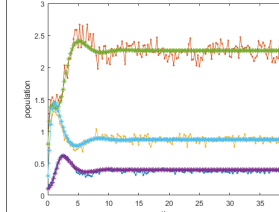
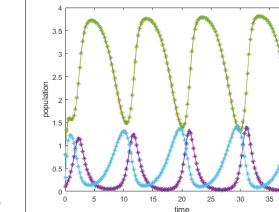
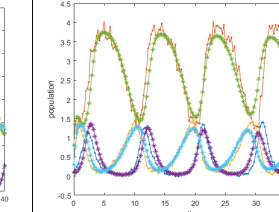
Future Predictions (tSVD)			
Stable Equilibrium		Limit Cycle	
$e_y = 0\%$	$e_y = 5\%$	$e_y = 0\%$	$e_y = 5\%$
$e_\alpha = 5.5405$	$e_\alpha = 28.951$	$e_\alpha = 5.8538$	$e_\alpha = 49.176$
$e_\beta = 5.1241$	$e_\beta = 25.903$	$e_\beta = 5.9722$	$e_\beta = 48.931$
$e_g = 0.41608$	$e_g = 5.4564$	$e_g = 1.2187$	$e_g = 20.257$
			

Table 5.1: **Predicting populations using tSVD.** Parameter errors (%) e_α , e_β , prediction error (%) e_g , and plots of noisy $h_i(t_j)$ vs. $g_i(t_j)$ for different e_y (% noise levels) over the time window $[0,40]$ with $\Delta t = 0.25$. $g_i(t_j)$ is given population measurements $h_i(t_j)$ only from the time interval $[0,7]$ and has to predict the remaining time points using tSVD. $g_i(t_j)$ is then compared with $h_i(t_j)$ for $t = [0, 40]$. $h_1(t)$ (blue —), $h_2(t)$ (red —), $h_3(t)$ (yellow —), $g_1(t)$ (purple —*), $g_2(t)$ (green —*), $g_3(t)$ (cyan —*).

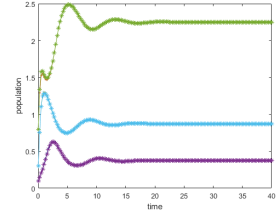
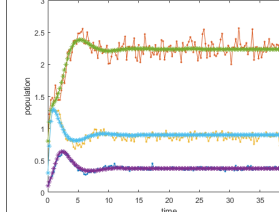
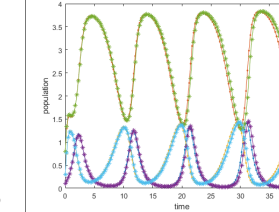
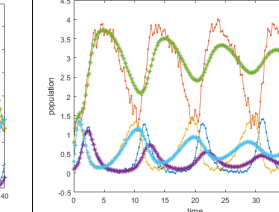
Future Predictions (LASSO)			
Stable Equilibrium		Limit Cycle	
$e_y = 0\%$	$e_y = 5\%$	$e_y = 0\%$	$e_y = 5\%$
$e_\alpha = 4.9233$	$e_\alpha = 19.161$	$e_\alpha = 5.7068$	$e_\alpha = 23.772$
$e_\beta = 4.5822$	$e_\beta = 17.824$	$e_\beta = 5.7807$	$e_\beta = 26.666$
$e_g = 0.35350$	$e_g = 5.3305$	$e_g = 6.5940$	$e_g = 23.883$
			

Table 5.2: **Predicting populations using LASSO.** Parameter errors (%) e_α , e_β , prediction error (%) e_g , and plots of noisy $h_i(t_j)$ vs. $g_i(t_j)$ for different e_y (% noise levels) over the time window $[0,40]$ with $\Delta t = 0.25$. $g_i(t_j)$ is given population measurements $h_i(t_j)$ only from the time interval $[0,7]$ and has to predict the remaining time points using LASSO regression. $g_i(t_j)$ is then compared with $h_i(t_j)$ for $t = [0, 40]$. $h_1(t)$ (blue —), $h_2(t)$ (red —), $h_3(t)$ (yellow —), $g_1(t)$ (purple —*), $g_2(t)$ (green —*), $g_3(t)$ (cyan —*).

5.5. Predicting Future Populations. One question that might arise is how well one can predict future population trajectories using one of the methods discussed above. For example, can these methods predict the correct equilibria in stable equilibrium dynamics or the same oscillations in limit cycle behavior when not given the entire time interval? To explore this, we extended the time interval for the population data to $t = [0, 40]$ but only gave the parameter recovery algorithm population data for $t = [0, 7]$. After the parameter recovery algorithm (LASSO or tSVD) created its population function $g_i(t_j)$, we compared it with

the full population data over the entire $[0,40]$ interval.

From Tables 5.1 and 5.2, it is immediately apparent that the predicted population function $g_i(t_j)$ was remarkably good at predicting the population equilibria for all species in stable equilibrium, regardless of the parameter recovery method used (tSVD or LASSO). While the recovered LV model parameters were different from the original ‘ground truth’ parameters, these alternate models were still able to very accurately predict the population data.

In the limit cycle case, the oscillating trajectories did not cycle together entirely. This was particularly the case using LASSO regression. Perhaps LASSO regression’s sparsity-promoting regularization caused one of the β parameters to be smaller than it should be. However, as discussed before, LASSO regression has great potential to improve, especially when allowed more steepest descent iterations, so this error may be mitigated.

6. Conclusion. In this study, we examined the Lotka-Volterra model and how it can simulate different types of population dynamics by using different sets of parameters. We showed how the inverse problem of determining parameters from the population data of a specific population dynamic can be cast as a linear regression problem. Additionally, we found a property we dubbed *noise insemination*, which describes the effects of noise in the population data on the regression problem. Numerical studies were then performed to demonstrate how the inverse problem can become ill-conditioned. Lastly, after characterizing the conditioning of this regression problem, we presented the results of the experiments on parameter recovery with and without noise using both tSVD and LASSO regression.

The experiments on parameter recovery with tSVD vs. LASSO regression led to several key observations. Truncated SVD generally recovered parameters faster than LASSO, but the parameters tSVD recovered, while closely matching the given population data, were often significantly different from the original “ground-truth” parameters. This was reflected by the calculated errors of the recovered parameters, which were significantly greater than the error of the reconstructed population data, suggesting that there are multiple different sets of parameters that can lead to very similar population behaviors.

On the other hand, while LASSO regression recovered parameters more slowly than tSVD, the resulting parameters were more parsimonious and could therefore be more useful than parameters recovered via tSVD. Additionally, although recreating the population data from parameters recovered from LASSO regression was not quite as accurate as from those recovered from tSVD, LASSO regression with its iterative ℓ_1 penalized least-squares algorithm has a lot of potential for improvement.

The addition of noise to the simulated population data led to a potentially useful relationship between the condition number, time-derivative estimation, and sampling rate when recovering parameters from noisy population data. Specifically, for well-conditioned dynamics, as sampling rate increases, the condition number worsens, and the time-derivative accuracy improves. On the other hand, for ill-conditioned dynamics, as sampling rate increases, both the condition number and time-derivative accuracy improve. Finally, noise in the population data led to an unexpected observation we called *noise insemination*, where noise in the population data corrupts variables in the regression problem, thus affecting the final recovered parameters.

Directions for future research could focus on a better understanding of noise insemination as well as better optimization methods. Specifically, studies could investigate how noise in $y_i(t_j)$ leads to error in the variables A_g and \mathbf{w}_g in the regression problem, and how that, in turn, impacts parameter recovery. Investigating the effects of changing λ and τ on parameter recovery via LASSO would also be a valuable endeavor. Future work could also investigate how much parameter recovery is improved by using more efficient LASSO algorithms. Another direction of research is getting some understanding of the impact of different types of measurement errors. In this work, we looked only at amplitude errors. There could also be errors in sampling time. Regarding optimization methods, investigation into other types of algorithms such as subgradients [20], iterative thresholding [21], and projected gradient descent [22] are considerations to explore. Knowing the particular dynamic of a set of population measurements may also help predict how well the parameters for the data can be recovered. These advancements in turn will help better predict how populations of organisms will change in the future, thus paving the way for increasingly fine-tuned bacteriotherapies and other real-world uses of the Lotka-Volterra model.

Acknowledgements

My sincerest thanks to my mentor, Prof. Fadil Santosa, who suggested this research topic, took me on as a mentee, and introduced me to undergraduate mathematics research as a high-school student. I would also like to thank the editors and referees at SIURO for their helpful, thorough, and patient comments to improve this manuscript.

References

- [1] A.J. LOTKA, *Elements of Mathematical Biology* (formerly published under the title *Elements of Physical Biology*). Dover Publications, New York, 1956.
- [2] V. VOLTERRA, *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi*, in *Memoria della Reale Accademia Nazionale dei Lincei*, Rome, Italy, 1926, pp. 31-113.
- [3] V. BUCCI, G.K. GERBER, B. TZEN, N. LI, M. SIMMONS, T. TANOUE, E. BOGART, L. DENG, V. YELISEYEV, M. L. DELANEY, Q. LIU, B. OLLE, R. R. STEIN, K. HONDA, AND L. BRY, *MDSINE: Microbial dynamical systems inference engine for micro-biome time-series analyses*. *Genome Biology*, 17 (2016), pp. 121-138.
- [4] V. BUCCI AND J. XAVIER, *Towards predictive models of the human gut microbiome*. *J. Mol. Biol.*, 426 (2014), pp. 3907-3916.
- [5] M. MCSEVENEY, *FDA in brief: FDA warns about potential risk of serious infections caused by multi-drug resistant organisms related to the investigational use of fecal microbiota for transplantation*, <https://www.fda.gov/news-events/fda-brief/fda-brief-fda-warns-about-potential-risk-serious-infections-caused-multi-drug-resistant-organisms>, (19 June 2019).
- [6] G. GERBER, *The dynamic microbiome*, *FEBS Letters*, 588 (2014), pp. 4131-4139.
- [7] M. HIRSCH, *Systems of differential equations which are competitive or cooperative: I. Limit sets*. *SIAM J. Math. Anal.*, 13 (1982), pp. 167-179.
- [8] M. HIRSCH, *Systems of differential equations that are competitive or cooperative ii: Convergence almost everywhere*, *SIAM J. Math. Anal.*, 16 (1985), pp. 423-439.
- [9] V. ROTH, *The generalized LASSO*. *IEEE Trans. Neural Netw. Learn. Syst.*, 15 (2004), pp. 16-28.
- [10] R. TIBSHIRANI, *Regression shrinkage and selection via the lasso*. *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, 58 (1996), pp. 267-288.
- [11] F. SANTOSA AND W. SYMES, *Linear inversion of band-limited reflection seismograms*, *SIAM Journal on Scientific and Statistical Computing*, 7 (1986), pp. 1307-1330.
- [12] J.M. VARAH, *A spline least squares method for numerical parameter estimation in differential equations*, *SIAM Journal on Scientific and Statistical Computing*, 3 (1982), pp. 28-46.
- [13] R. R. STEIN, V. BUCCI, N. C. TOUSSAINT, C. G. BUFFIE, G. RÄTSCHE, E. G. PAMER, C. SANDER, AND J. B. Xavier, *Ecological Modeling from Time-Series Inference: Insight into Dynamics and Stability of Intestinal Microbiota*, *PLOS Computational Biology*, 9 (2013), pp. 1-11.
- [14] J. HOFBAUER AND J.W.-H. SO, *Multiple limit cycles for three dimensional Lotka-Volterra equations*, *Appl. Math. Lett.*, 7 (1994), pp. 65-70.
- [15] I. NAGY, V. G. ROMANOVSKI, AND J. Tóth, *Two Nested Limit Cycles in Two-Species Reactions*, *Multidisciplinary Digital Publishing Institute*, 8 (2020), pp. 1658-1673.
- [16] J.A. VANO, J.C. WILDENBERG, M.B. ANDERSON, J.K. NOEL, AND J.C. SPOTT, *Chaos in low-dimensional Lotka-Volterra models of competition*, *Nonlinearity*, IOP Publishing, 19 (2006), pp. 2391-2404.
- [17] C. VOGEL, *Computational Methods for Inverse Problems*, *SIAM Monograph Series on Frontiers in Applied Mathematics*, Philadelphia, PA, 2002.
- [18] J. NOCEDAL AND S. WRIGHT, *Numerical Optimization*, Springer, 1st ed., New York, NY, 1999.
- [19] K. ATKINSON, W. HAN, AND D. STEWART, *Numerical Solution of Ordinary Differential Equations*, John Wiley & Sons, Hoboken, NJ, 2009.
- [20] A.M. BAGIROV, L. JIN, N. KARIMITSA, A. AL NUAIMAT, AND N. SULTANOVA, *Subgradient method for nonconvex nonsmooth optimization*, *J. Optim. Theory Appl.*, 157 (2012), pp. 416-435.
- [21] T. BLUMENSATH AND M.E. DAVIES, *Iterative thresholding for sparse approximations*, *J. Fourier Anal. Appl.*, 14 (2008), pp. 629-654.
- [22] Y. CHEN AND M.J. WAINWRIGHT, *Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees*, <https://arxiv.org/abs/1509.03025> (10 Sept 2015).