AMATH 563 Homework 02
Dynamics and Model Discovery
Joseph J. Williams
2020 May 01

This investigation focused on modeling and predicting the populations of snowshoe hares and Canadia lynxes; populations (in thousands) were given every two years from 1845 to 1903. A first-attempt dynamic mode decomposition (DMD) model was made that modeled and forecasted the populations very poorly, but subsequent attempts with time-delay embedding saw significantly more success, with a variety of Hankel matrices explored and errors computed. As an alternate approach, the sparse identification of nonlinear dynamics (SINDy) algorithm, a sparsity-promoting regression analysis, was used to discover the best fit nonlinear dynamics, which were able to model the populations significantly more richly with the right choice of library functions. KL divergences and AIC and BIC scores were computed for all the models, with the SINDy models having the lowest KL divergences and SINDy with $\lambda_3 = 2.5$, generating the sparest nonlinear model, having the lowest AIC and BIC scores.

Sec. I Introduction and Overview

The populations of snowshoe hares and Canadian lynxes from 1845 to 1903 is a classic dataset used as a starting point for studying and understanding population dynamics and, more broadly, nonlinear dynamic systems. Nonlinear systems are notoriously impervious to even the most formidable of mathematicians; at most, we can hope only to approximate the system as linear in some way in order to even have hope of understanding it. Indeed, our professor himself is fond of saying (paraphrased), "People get hurt when they try to understand nonlinear dynamics."

Oftentimes, when studying dynamical systems $d\mathbf{x}/dt = f(\mathbf{x}, t, \beta)$ in the sciences, we seek to understand a given classic dynamical system f such as the Navier-Stokes equations. However, a modern problem is to be given a wealth of data without any any known dynamical system representation, such as user data from a large website, electrical readings from the brain, or PIV data from a turbulent experiment – and modern problems require modern solutions, namely, using data-driven solutions such as dynamic mode decomposition (DMD), an operator theoretic representations, and the sparse identification of nonlinear dynamics (SINDy) algorithm, a type of data-driven, sparsity-promoting regression, to create a model. This has the advantage of being able to be updated as new data is taken in, minimizing the disruption of unmeasured quantities to the dynamical system.

There are four main goals in modern dynamical systems: future state prediction, design and optimization, estimation and control, and interpretability and physical understanding. This study will be focusing on the primarily the first and fourth goals with regards to understanding and predicting the populations of hares and lynxes. First, models will be created first using DMD, first without time-delay embedding and then with, exploring a variety of time-delay embedding methodologies; second, we will explore the possibility of using the Lotka-Volterra equations to model our dynamics before using SINDy with a wide library of terms to build the best-fit nonlinear dynamical model for the population dynamics. We will then compute the KL divergences and AIC and BIC scores of our models.

The final part to this report details applying DMD and SINDy to images from Belousov-Zhabotinsky (BZ) chemical oscillator; this final part has elements of the previous study in it as well.

Sec. II Theoretical Background

As discussed previously, DMD does not require any knowledge of governing equations of the system; instead, it is based purely on measurement data, specifically m such samples. The methodology is in truth quite straightforward and lends itself to implementation well. Based on proper orthogonal decomposition (POD), DMD provides a modal decomposition, and each mode consists of spatially correlated structures that exhibit the same linear behavior in time; this may be understood as as both a reduced set of modes and a model for how these modes evolve in time.

First, form a matrix **X** from the first (m – 1) samples, arranged as column vectors, and a matrix **X**' from the last (m – 1) samples, also arranged as column vectors. We seek the transformation matrix A s.t. A**X** = **X**'. Since these systems are usually vastly overdetermined (far more equations than unknowns, i.e far more measurements in time than unique measurement variables) we seek the best-fit A; this is computed i a straightforward way by computing the singular value decomposition X = UΣV* of **X** and finding the Moore-Penrose pseudoinverse $\mathbf{X}^{\dagger}$ s.t. A = **X**'$\mathbf{X}^{\dagger}$. In words, A is the best-fit matrix that take the first (m – 1) measurements in time of the state variables to their collectively best-fit state Δt in the future.

There are a variety of improvements that have been made to the overall DMD algorithm, resulting in algorithms such as optimized DMD, exact DMD, multi-resolution DMD, and extended DMD, to name but a few. As a second pass at modeling and predicting the data with DMD, we will be working with time-delay embedding, in which we construct the Hankel matrices **H** and **H**' and find as before the best-fit A s.t. A**H** = **H**'; reference Fig 2.1 below:

$$\mathbf{H} = \begin{bmatrix} x(t_1) & x(t_2) & \cdots & x(t_{m_c}) \\ x(t_2) & x(t_3) & \cdots & x(t_{m_c+1}) \\ \vdots & \vdots & \ddots & \vdots \\ x(t_{m_o}) & x(t_{m_o+1}) & \cdots & x(t_m) \end{bmatrix} = \mathbf{U\Sigma V}^*$$

Figure 2.1: The SVD of the Hankel matrix H, a result of time-delay embedding

There are a variety of ways to time-delay embed a system. Recognizing that the original matrix **X** is very wide (reflecting a large amount of data) and comparatively short (reflecting many fewer measured quantities), creating **H** allows us to control the dimension of the matrix whose SVD we seek and thus obtain more DMD modes. This is controlled through m_c, the width of the time-delay embed (i.e. how many temporal samples make up a single row), and m_o, the degree of the time-delay embed (i.e. how many rows of temporal samples will there be).[1] Thus, increasing m_o and decreasing m_c makes **H** taller and skinnier, making the system less overdetermined.

With or without time-delay embedding, we compute the SVD and pseudoinverse of **X** or **H** and then compute A; in fact, the algorithm is usually implemented such that a reduced matrix $\tilde{A}$ is obtained, which will have the same eigenvalues as the full matrix A. We then find the spectral decomposition $\tilde{A}W = W\Lambda$ of $\tilde{A}$. The final steps are to reconstruct the high-dimensional DMD modes $\mathbf{\Phi} = \mathbf{X}'V\Sigma^{-1}W$ or $\mathbf{\Phi} = \mathbf{H}'V\Sigma^{-1}W$; the DMD modes are used to predict the state **x** at any time k with $\mathbf{x}_k = \mathbf{\Phi}\Lambda^{k-1}\mathbf{b}$, where $\mathbf{b} = \mathbf{\Phi}^{\dagger}\mathbf{x}_1$ or $\mathbf{b} = \mathbf{\Phi}^{\dagger}\mathbf{h}_1$. Thus, this method will also perfectly reconstruct the first m_o temporal states, where $m_o > 1$ for time-delay embedded systems and 1 otherwise. The algorithm is more fully explained on pgs. 275 – 279 of the text.

Before focusing on SINDy, we briefly investigate the Lotka-Volterra equations $\dot{x} = (b - py)x$ and $\dot{y} = (rx - d)y$, a set of nonlinear differential equations used for modeling predator-prey dynamics in biological systems. From plotting the given data, we estimate values for parameters b, p, r, and d and then test a range of parameters with values centered around those. This is a case where we are given some dynamical model $d\mathbf{x}/dt = f(\mathbf{x},t,\beta)$ and then try to understand its mathematical properties in order to gain some insight into the real world.

Whereas DMD uses the data to develop a linear model that best fits the data and predicts future states, and in contrast to the Lotka-Volterra model, where a dynamical system $d\mathbf{x}/dt = f(\mathbf{x},t,\beta)$ is prescribed and we work to understand it and match the data to it, SINDy uses the data to develop a sparse, nonlinear model for the dynamics by fitting the data to a library (usually large) of candidate functions. This method, necessarily based on regression methods, also lends itself well to algorithmic implementation. Starting as in DMD with m temporal samples of the state, we form the transpose of the data matrix $\mathbf{X}^T$ and either gather or compute from the data $\dot{\mathbf{X}}^T$, the derivatives. In

[1] I couldn't find more proper names for these terms in the textbook or in a brief online search, but what I have is descriptive enough.

general, the higher the quality of your derivatives $\dot{\mathbf{X}}^{\mathbf{T}}$, the better your results. We then form a library of candidate functions $\boldsymbol{\Theta}(\mathbf{X})$; reference Fig. 2.2 below, which shows a general $\boldsymbol{\Theta}(\mathbf{X})$,

$$\boldsymbol{\Theta}(\mathbf{X}) = \begin{bmatrix} \mathbf{1} & \mathbf{X} & \mathbf{X}^2 & \cdots & \mathbf{X}^d & \cdots & \sin(\mathbf{X}) & \cdots \end{bmatrix}$$

Fig. 2.2: A general library of nonlinear candidate functions $\boldsymbol{\Theta}(\mathbf{X})$ that may model the data

The weights $\boldsymbol{\Xi}$ are solved for by solving $\dot{\mathbf{X}}^{\mathrm{T}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}$. The weight for a given candidate function w for a given state variable k is $\xi_{w,k}$. Future states are then predicted by solving first for a state k's derivative at a given time as $\dot{x}_k = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}$ and then computing from the time step Δt the state at the next point in time; as with computing $\dot{\mathbf{X}}^{\mathbf{T}}$, there exist a variety of methods from pedagogical to cutting-edge for computing values from derivatives and time steps. The algorithm is more fully explained on pgs. 288 – 290 of the text.

With models in hand, we compute their KL divergences, which we seek to minimize; it is defined in Fig. 2.3 below:

$$I(f, g) = \int f(\mathbf{X}, \boldsymbol{\beta}) \log \left[\frac{f(\mathbf{X}, \boldsymbol{\beta})}{g(\mathbf{X}, \boldsymbol{\mu})} \right] d\mathbf{X}$$

Figure 2.3: KL divergence, a measure distance between two probability distributions

The KL divergence measures the distance between two probability distribution functions. When f is the data, then the KL divergence may be thought of as the amount of information lost when a model g is used to represent data f; if we make the philosophical leap that the collected data f represents reality, then the KL divergence essentially measures how good g is at modeling and predicting the world around us. In practice, however, full knowledge of the statistics of f is not in hand and the KL divergence cannot be computed; thus was developed the AIC and BIC scores, which we seek to minimize; reference Fig. 2.4 below:

$$AIC = 2K - 2\log\left[\mathcal{L}(\hat{\boldsymbol{\mu}}|\mathbf{x})\right]$$

$$BIC = \log(n)K - 2\log\left[\mathcal{L}(\hat{\boldsymbol{\mu}}|\mathbf{x})\right]$$

Figure 2.4: AIC (*top*) and BIC (*bot.*) scores, an estimate of the distance from the developed model to the true model

The AIC and BIC scores are higher for models with a higher number of terms, a penalization for lacking sparsity, and lower for models that better match the data. They are only slightly different in form and in properties; the principal difference is that a lower BIC score will always be associated with the correct model from which the data came if it is included in the host of candidate models, whereas the AIC score may not, even when it is included.

In the final part of this study, we apply DMD and SINDy to images from the BZ experiment; first we normalize the data to all be between 0 and 1, then we reform the images at each time into column vectors. Thus our vector of state variables is $\mathbf{x}(t) \in \mathbb{R}^{m*n}$, where m and n are the length and width, in pixels, of the images. We then arrange the vectors as before and apply the algorithms as previously. KL divergences and AIC and BIC scores will be computed.

Sec. III Algorithm Implementation and Development

The algorithms for the methods used in this study were described mathematically in the previous section; their implementation is straightforward and code samples are provided in the text: DMD on pg. 280, SINDy on pgs. 289 – 290, computing KL divergences on pg. 174, and computing AIC and BIC scores on pg. 176. The code developed for this study made direct use of the SINDy code *sparsifyDynamics* provided in the textbook on pg. 289 and followed many of the other provided code samples in the text closely.

Sec. IV Computational Results

IV.0

Moving to the computational results of this study, reference Fig. 4.0.1 below, which plots the provided hare and lynx population data through the years.

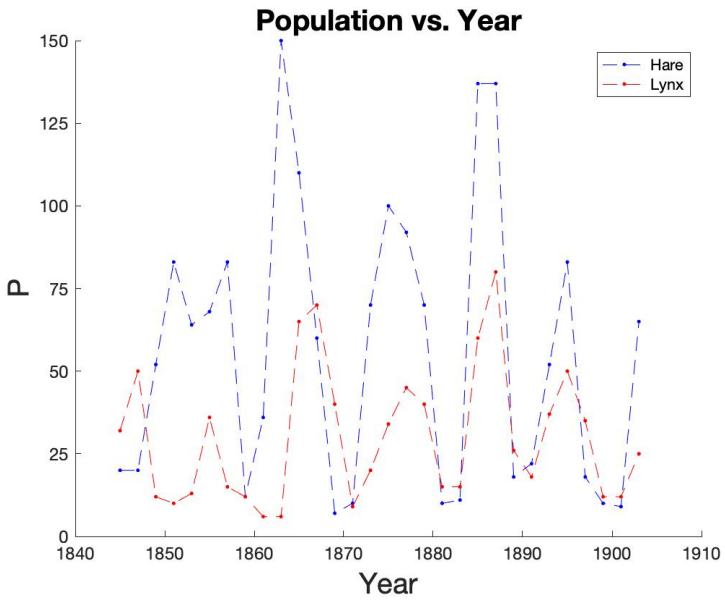

Figure 4.0.1: Hare and Lynx Population Data for 1840 – 1910

While a detailed analysis of this data is the main subject of this report, we may start with some general comments. First, we see that there is clearly some oscillatory behavior in both populations, as well as possibly some interdependence; natural intuition tells us, of course, that there is more than just *some*. Second, the oscillations in the hare population are much more pronounced than in the lynx population. Finally, both populations rise quickly from local minima to local maxima before precipitously declining again to local minima. This is seen in the hare populations in 1849–1859, 1859–1869, 1869–1881, and 1883–1889;[2] the lynx population, meanwhile, is steadier but displays similar rise-and-fall behavior in the years 1863–1869 and 1883–1891. except for ranges of years such as 1849 – 1857 for hares and 1849 – 1863 for lynxes, the populations rarely stay steady.

IV.1

Moving now to our computational analysis, our first attempt at generating a model and forecasting the future is a naive implementation of DMD without any time-delay embedding. Reference Fig. 4.1.1 on the following page, which shows (*left*) the hare population and (*right*) the lynx population data against their DMD predictions; the DMD predictions are further extended to the year 1943.[3] These are clearly very poor predictions; while they match the first data point (as expected, recalling the discussion of § II) they decay to zero very quickly, completely failing to model even the data it was based on. While not ideal, this is in fact expected; we may gain some intuition by investigating the eigenvalues of $\tilde{A}$: $\lambda_1 = 0.7248$ and $\lambda_2 = 0.3784$. These eigenvalues are related to the frequencies of the solution

[2] One can't help but feel a tinge of remorse for all those poor little rabbits.

[3] The original data set had 30 data samples two year apart; predicting until 1943 is 50 data samples, still all two years apart. However, World War II probably had some effect on the populations, as more than a million Canadians served in the war, almost a tenth of its total population then.

by $\omega_i = \log(\lambda_i) / \Delta t$; $0 < \lambda_i < 1 \rightarrow \log(\lambda_i) < 0$, so our solution $\mathbf{x}(t) = \mathbf{\Theta}\exp(\Omega t)\mathbf{b}$ decays exponentially with time. This is in fact the behavior we observe in the cyan and magenta lines of Fig. 4.1.1. Thus our DMD solution has completely failed to capture the clear oscillatory behavior present in our data; in order to improve our model, we time-delay embed our system and construct $\mathbf{H}$ as in order to obtain $\lambda_i < 0 \rightarrow \log(\lambda_i) \in \mathbb{C} \rightarrow \mathbf{x}(t) = \mathbf{\Theta}\exp(\Omega t)\mathbf{b}$ will oscillate with reasonable amplitude before either decaying to zero or blowing up, depending on if any $\lambda_i > 0$ or $\lambda_i > 1$. Time-delay embedding essentially allows us to build a matrix that will decompose whatever way we choose.

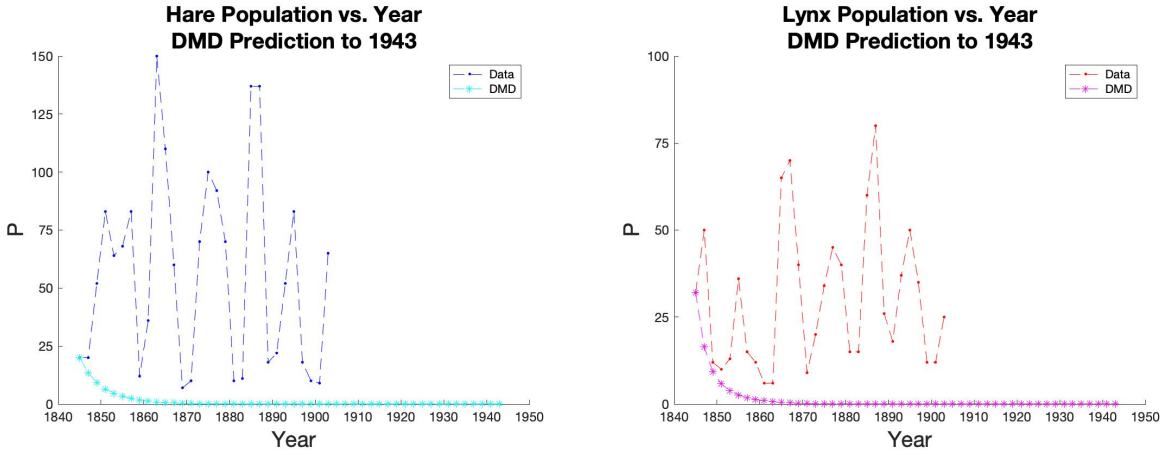

Fig. 4.1.1: DMD Model Forecasting to 1943 the Future Hare (*left*) and Lynx (*right*) Populations

IV.2

In pursuit of this, reference Fig. 4.2.1 on the following page, which shows the DMD model forecasting with time-delay embedding for a variety of (m_c, m_o); note that we require $m_c + m_o \leq m = 30$, otherwise we would be indexing for more measurements than we have. We see that time-delay embedding significantly improves the DMD model, recreating the oscillatory behavior in the data. While $(m_c, m_o)_1 = (5, 25)$ and $(m_c, m_o)_2 = (25, 5)$ appear to have significant exponential decay terms, capturing the oscillatory behavior only for a limited number of cycles before being damped out, $(m_c, m_o)_3 = (15, 15)$ propagates the oscillatory behavior into the future at reasonable amplitudes without significant damping. While the population does become negative at some points, it is possible that unmodeled dynamics, such as different animal behavior or human intervention, could take over at very low population levels.

IV.3

We next move to an exploration of the Lotka-Volterra model. We being by estimating reasonable values of the parameters *b*, *p*, *r*, and *d* and numerically propagate the solution to the system forward in time using MATLAB's in-built *ode23*; reference Fig. 4.3.1 on the following page, which shows the data plotted against the Lotka-Volterra model for $(b, p, r, d) = (0.5, 0.04, 0.02, 1)$. We see that, with these parameters, the Lotka-Volterra model does a reasonably good job of approximating the data: both population models have about as many oscillations as they should, and they appear to be in phase with the data as well; further, the amplitude of the oscillations in the model appear to approximate the average amplitude of the oscillations of the data, though the lynx amplitude may be a little low. Overall, this model approximates the hare and lynx populations passably well but fails to capture the decade-to-decade change in the amplitude of the oscillation. We thus move to a more exhaustive study using SINDy.

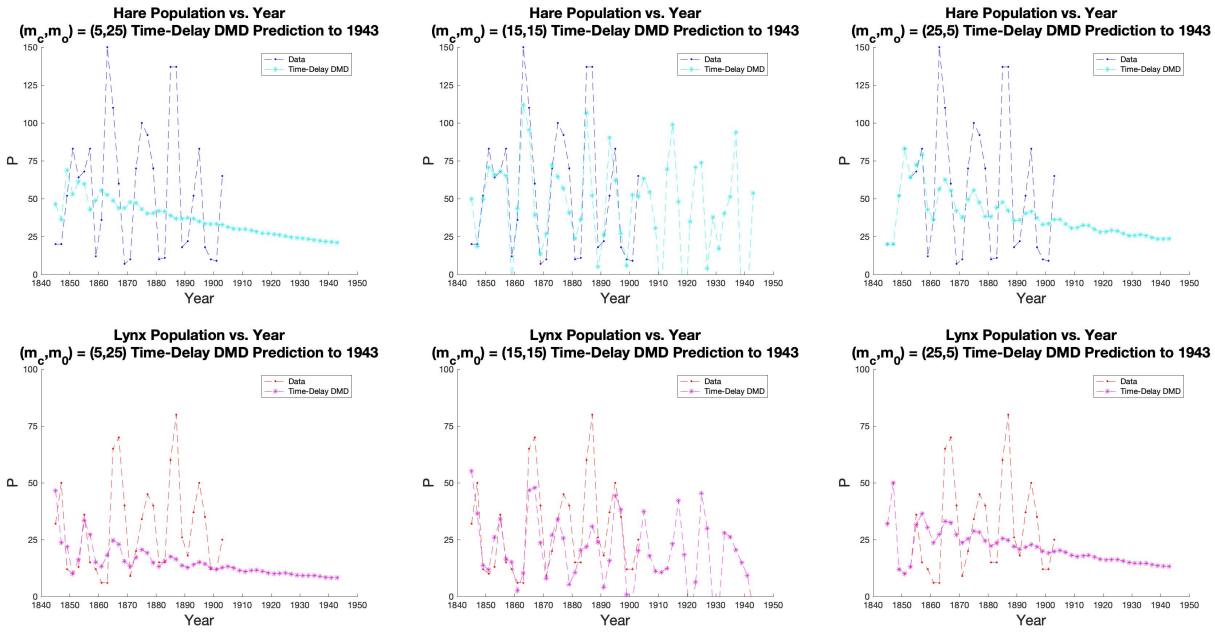

Fig. 4.2.1: Time-Delay Embedded DMD Forecasting to 1943 the Future Hare (*top*) and Lynx (*bottom*) Populations

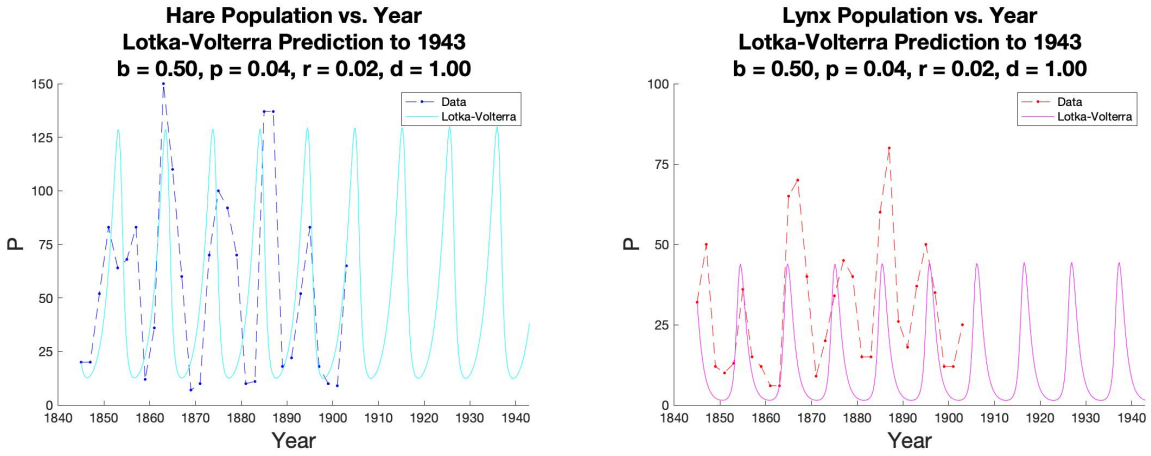

Figure 4.3.1: Lotka-Volterra Model Forecasting to 1943 the Future Hare (*Left*) and Lynx (*Right*) Populations

IV.4

Before finding a nonlinear model with SINDy, we must construct a library of candidate functions. While mathematically this library may be as large as we wish, including even the most exotic of functions, we should limit ourselves to those our understanding of the underlying physics at work deem reasonable; we are further limited by the computing restraints of the author. Our candidate functions include all combinations of products of our state variables from zeroth to third power as well as sines and cosines of both populations with frequencies B = 1, 2, and 3; a full list is given in Appx. A. We carry out the SINDy algorithm through the provided *sparsifyDynamics.m* code, with which we can specify a sparsity-promoting factor λ, higher values of which set increasingly more terms to 0; reference Fig. 4.4.1 below, which shows the weights of the functions ξ_w for both populations for various λ.

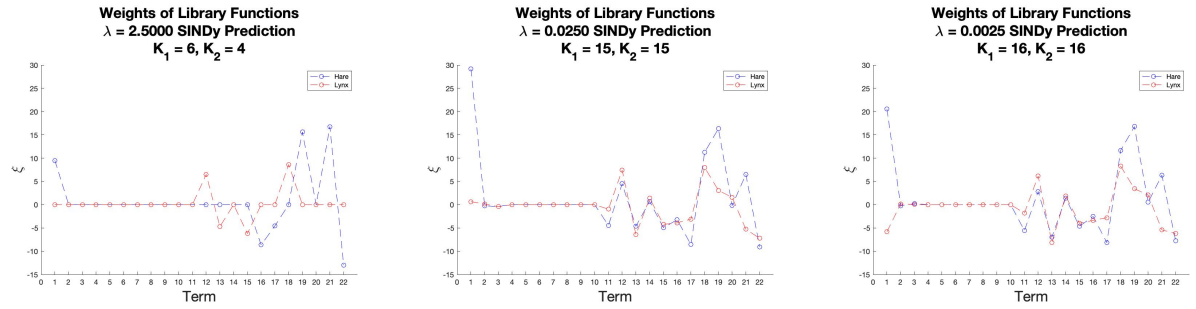

Fig. 4.4.1: Weights of Library Functions for $\lambda \in \{0.0025, 0.025, 2.5\}$

For all values of λ, the entries for ξ_w for the different populations largely track each other and have almost exclusively the sinusoidal terms as nonzero; a notable exception and difference between the two populations is that the hare population has a significant constant value for all λ while the lynx population does not. For $\lambda = 2.5$, the sparsest model, the terms in the model for the hare population are $\{1, \cos(2x), \sin(2y), \sin(3x), \sin(3y), \cos(3y)\}$ while those for the lynx population are $\{\cos(x), \sin(y), \sin(2x), \cos(2y)\}$; while they are all sinusoidal (except for the constant term in the hare population); none of them are common between the two populations. One interpretation for the constant term in the hare model is perhaps that it reflects the predator-prey relationship; assuming a simplistic model where there is plenty of foliage for the (presumably herbivorous) hares to consume and no other prey than hares for the lynx to consume, then the hares survival is despite the lynx, but the lynx could not survive without the hare. Thus, its population depends entirely on functions of the populations and hence has no constant term.

Next, we use the weights for the functions to actually model and predict the data; reference Fig. 4.4.2 below, which shows the hare and lynx populations modeled and forecasted until 1943 with the same values of λ.

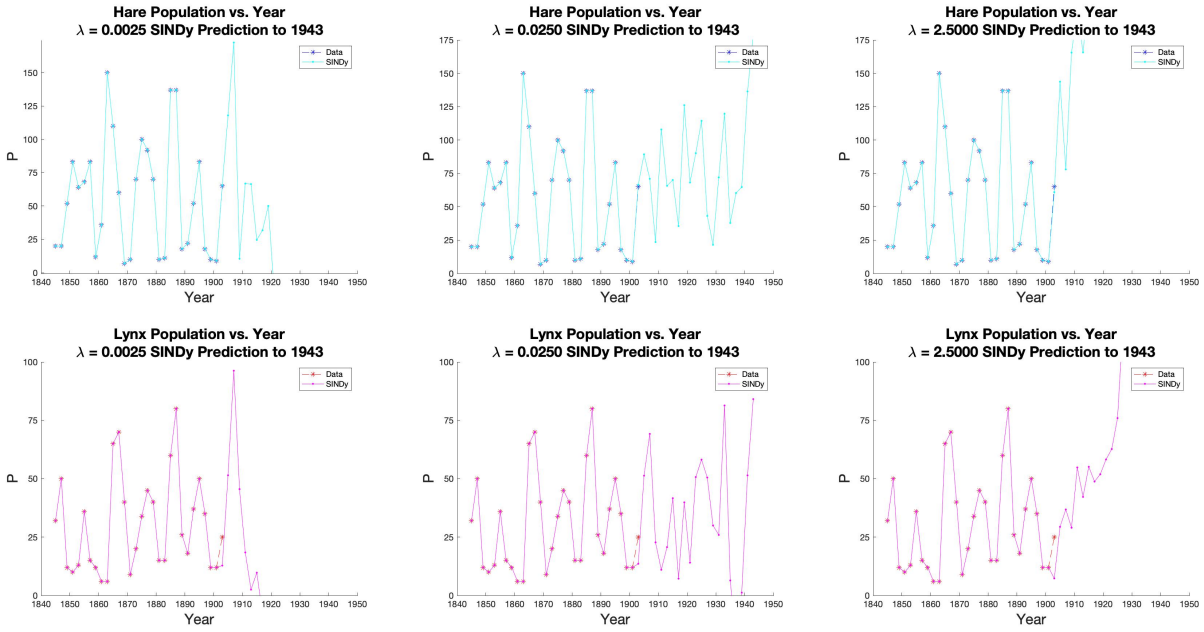

Fig. 4.4.2: SINDy Forecasting to 1943 the Future Hare (*top*) and Lynx (*bottom*) Populations

We see that, although $\lambda_3 = 2.5$ provided the sparsest model with almost exclusively oscillatory terms, $\lambda_2 = 0.025$ appears to model the data with the greatest fidelity. With $\lambda_2 = 0.025$, for both populations, all oscillatory terms are activated as well as constant and linear terms in both variables. It is perhaps these additional linear and constant terms that allow for greater richness in the results and realistic forecasting. The finite-difference method used to create $\mathbf{x}_{t+1}$ from $\mathbf{x}_t$ as well as to create $\dot{\mathbf{X}}^T$ in the initial step of the problem was a simple first-order finite difference method; with a more robust method, such as a second-order central difference, the quality of the results would rise.

Some interesting physical interpretations of each of the three models are: for $\lambda_1 = 0.0025$, human (in)action results in the destruction of the ecosystem and a collapse of both populations, assuming different populations dynamics don't take over; for $\lambda_2 = 0.025$, the ecosystem continues as it has for 60 years with minor changes in its population dynamics, the occasional negative values being explained again as before away as times when unmodeled population dynamics or (favorable) human intervention occurs; and for $\lambda_3 = 2.5$, the lynx and hare populations explode, perhaps as a result of an increase in resources, again assuming different population dynamics at this state.

We now move to computing the KL divergence of each model. The method mirrors that provided in the book, referenced in § III. Reference Table 4.5.1 on the following page, which tabulates the KL divergences of each of the models previously discussed for both populations. The three most successful models, as measured by the smallest KL divergence, are the SINDy models with different sparsity factors λ. $\lambda_1 = 0.0025$, which was the least sparse model, had the lowest KL divergences for its populations. That this model was the least sparse should penalize it somewhere, which leads us naturally to computing AIC and BIC scores.

Reference Table 4.5.2 on the following page, which tabulates the AIC and BIC scores of the SINDy models previously discussed for both populations. The method mirrors that provided in the book, referenced in § III. From the values, we see that the model with the lowest AIC and BIC scores is the SINDy model with $\lambda_3 = 2.5$, the highest sparsity-promoting value of the three. Thus, this model, out of all tested throughout this study, does the best job of recreating and predicting the states from the data.

Model	**KL Divergence**
1. DMD	$I_{hare} = 3.8377$ $I_{lynx} = 3.8477$
2. DMD with TDE $(m_c, m_o)_1 = (5, 25)$	$I_{hare} = 4.5482$ $I_{lynx} = 3.9074$
2. DMD with TDE $(m_c, m_o)_2 = (15, 15)$	$I_{hare} = 3.7543$ $I_{lynx} = 3.4243$
2. DMD with TDE $(m_c, m_o)_3 = (25, 5)$	$I_{hare} = 3.4129$ $I_{lynx} = 4.0746$
3. Lotka-Volterra	$I_{hare} = 3.6443$ $I_{lynx} = 4.2026$
4. SINDy, $\lambda_1 = 0.0025$	$I_{hare} = 0.1489$ $I_{lynx} = 0.1280$
4. SINDy, $\lambda_2 = 0.025$	$I_{hare} = 0.1489$ $I_{lynx} = 0.1489$
4. SINDy, $\lambda_3 = 2.5$	$I_{hare} = 0.1489$ $I_{lynx} = 0.1489$

Table 4.5.1: KL Divergences between each model and the data for both populations; winning models highlighted

Model	**AIC Score**	**BIC SCore**
4. SINDy, $\lambda_1 = 0.0025$	$AIC_{hare} = 40.0493$ $AIC_{lynx} = 39.6224$	$BIC_{hare} = 62.4685$ $BIC_{lynx} = 62.0415$
4. SINDy, $\lambda_2 = 0.025$	$AIC_{hare} = 38.0493$ $AIC_{lynx} = 37.6658$	$BIC_{hare} = 59.0673$ $BIC_{lynx} = 58.6838$
4. SINDy, $\lambda_3 = 2.5$	$AIC_{hare} = 20.0493$ $AIC_{lynx} = 15.6658$	$BIC_{hare} = 28.4565$ $BIC_{lynx} = 21.2706$

Table 4.5.2: AIC and BIC scores of the top three best-fit models for each population; winning model highlighted

As a final part of this study, we seek to perform DMD on the BZ.mat data set. We first reform the download variable BZ_tensor.mat into a matrix $\mathbf{X}$ whose dimensions are $(m*n,T) = (158301, 1200)$; previously, T was represented by m, but m now symbolizes the length of the image. We then proceed with the DMD algorithm as previously detailed.

Unfortunately, the total number of pixels, 158301, is prohibitively large, and my machine could not reasonably handle the associated matrices. As such, we sampled a reduced version, taking only $M*N = 8*10 = 80$ pixels, taking every twentieth row and column of pixels. This was much more computationally feasible. Even still, with sampling such a reduced version of the full data-set, DMD with any time-delay embedding gives a KL divergence of just 0.0015, suggesting that we will do an extremely good job of modeling the given data (pixel intensities). This is surprising, given that we are using only a small number of the total available pixels, but then, chemical reactions are probably more predictable than population dynamics.

Sec. V Summary and Conclusions

In this study, we applied the DMD and SINDy algorithms to the classic snowshoe hare and Canadian lynx data from 1845 through 1903. DMD was found to be significantly better at modeling the data and forecasting future states when time-delay embedding was used, but, with a diverse enough library of candidate functions, SINDy was found to produce better models overall, with lower KL divergences and AIC and BIC scores.

Areas for future work are numerous, including:

- Exploring the singular values of the Hankel matrices $\mathbf{H}$ and the eigenvalues of the reduced matrix $\tilde{A}$ in the time-embedded DMD algorithm; we should ensure that they line up with the expectations laid out
- Using a higher-order finite-difference operator to compute $\dot{\mathbf{X}}^T$ and $\mathbf{x}_{t+1}$ in the SINDy algorithm; with a higher-order approximation to values central in the algorithm, the quality of the results will rise.
- Interpolating the hare and lynx populations between the two year periods in order to generate more data points; within that, there are a variety of ways one could interpolate between the data points. With more data points, DMD and SINDy may be able to reproduce higher quality results.
- Exploring a wider range of values of λ, plotting the AIC and BIC scores against a wide range of λ with very fine step sizes; although attempted by this author, this proved prohibitively computationally expensive. Also of interest would be increasing the number of measurements and observing the effect on the algorithm.
- Exploring a wider range of widths and degrees of time-delay embedding (m_c, m_o) and particularly exploring the space of $m_c + m_o < 30$. It is possible that the KL divergence and AIC and BIC scores vary significantly within the entire space of possible (m_c, m_o). Also of interest would be increasing the number of measurements and observing the effect on the algorithm.
- Performing DMD on the entire set of pixels from the BZ experiment; my machine could only handle a very reduced subset of pixels, but even with that reduced version, DMD gave a very small KL divergence.
- Performing DMD with time-delay embedding on a subset or the entire set of pixels from the BZ experiment. Unfortunately, I just didn't quite get enough time to try this (I write this at 11:55PM), although I think my code for it almost runs.

Appendix A

This appendix provides a brief explanation of how the code runs.

First, run the first sections of AMATH563_HW02_Main.m. Then you may run any or all of sections titled 1. through 4. Each corresponding section of 5. Depends on that part's section in AMATH563_HW02_Main, and 6. Depends on part 4. of 5. Different parts of the code reference various included helper functions. The code taken from the textbook was noted in § III.

See Table A.1 below for the full library of candidate functions used for the SINDy algorithm.

Function *w*	**θ(w)**
1	1
2	x
3	y
4	x^2
5	xy
6	y^2
7	x^3
8	x^2y
9	xy^2
10	y^3
11	sin(x)
12	sin(y)
13	cos(x)
14	cos(y)
15	sin(2x)
16	sin(2y)
17	cos(2x)
18	cos(2y)
19	sin(3x)
20	sin(3y)
21	cos(3x)
22	cos(3y)

Table A.1: The library of candidate functions used in this study

Appendix B

See GitHub repository:

https://github.com/williamsj0165/AMATH563_HW02.git