ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CƠ KHÍ

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP

(Dành cho Giáo viên hướng dẫn)

Tên đề tài: Data Science in Identification of Mechanical Systems

Họ và tên SV: **Nguyễn Huy Công** Lớp: TT.CĐT.01 – K62

Chuyên ngành: Cơ điện tử
Giáo viên hướng dẫn: TS. Nguyễn Hải Sơn

NỘI DUNG NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

I. Tác phong làm việc

The student is diligent and proactive to fulfil the requirements of the graduation project.

II. Những kết quả đạt được

The student has researched and applied the Sparse Regression method in identifying mechanical systems.

III. Hạn chế của đồ án

The mechanical systems in the graduation project is quite simple.

IV. Kết luận

Người hướng dẫn đề nghị cho phép sinh viên được bảo vệ đề tài tốt nghiệp trước Hội đồng chấm đồ án tốt nghiệp.
Đánh giá: 9/10 điểm

Hà Nội, ngày 11 tháng 08 năm 2023
Giáo viên hướng dẫn

Nguyễn Hải Sơn

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF MECHANICAL ENGINEERING

Socialist Republic of Vietnam
Independence-Freedom-Happiness

GRADUATION THESIS EVALUATION

(For Thesis advisor)

Graduation thesis title: Data Science in the Identification of mechanical systems.
Full name: **Nguyễn Huy Công** Class: **TT.CĐT.01-K62**
Major: Mechatronics

EVALUATION CONTENT

I. Achieved results

Student had studied about the methods to identification of mechanic systems such as Sparse Regression, Lasso Regression and using data science and AI to solve Lorenz system, application for pendulum problems

II. Drawbacks of graduation thesis

Application only for simple case.

III. Conclusion

The dissertation committee approved the student to (not to) defend the graduation thesis topic before the graduation thesis dissertation committee council.

Evaluation: 10 points

Hanoi, August 10 2023
Thesis advisor committee

ph.D. Đặng Trọng Doanh

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

GRADUATION THESIS

Data Science in Identification of Mechanical Systems

NGUYỄN HUY CÔNG

cong.nh176570@sis.hust.edu.vn

Major Mechatronics

Instructor: Dr. Nguyễn Hải Sơn

_____

Signature of instructor

Specialized subject: Applied Mechanics

School: School of Mechanical Engineering

HÀ NỘI, 8/2023

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF MECHANICAL ENGINEERING

Socialist Republic of Vietnam
Independence-Freedom-Happiness

GRADUATION PROJECT TASK

(Specialized subject in Applied Mechanics)

1. **Student's information:**
   Full name: **Nguyễn Huy Công** Class: **TT.CĐT.01-K62**
   Tel.: 0977 211 864
   Email (presentative): cong.nh176570@sis.hust.edu.vn
   Training Type: Formal
   Major: Mechatronics
   Graduation thesis is executed at Specialized subject in Applied Mechanics
   Graduation thesis duration: From March 2023 to August 2023
2. **Thesis title:**
   Data science in the identification of mechanical systems.
3. **Design task:**
   - Study about Sparse Regression.
   - Learn the Lasso function in Matlab.
   - Simulating and using Sparse regression (The Lorenz system) with MATLAB.
   - Apply the Lasso Regression to the simple pendulum problem.

Hanoi, August ……2023
Instructor

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF MECHANICAL ENGINEERING

Socialist Republic of Vietnam
Independence-Freedom-Happiness

GRADUATION THESIS EVALUATION

(For Instructor)

Graduation thesis title: Data Science in the Identification of mechanical systems.
Full name: **Nguyễn Huy Công**
Class: **TT.CĐT.01-K62**
Major: Mechatronics
Instructor name: Dr. Nguyễn Hải Sơn

EVALUATION CONTENT

I. Working manners

.....
.....
.....

II. Achieved results

.....
.....
.....
.....

III. Drawbacks of graduation thesis

.....
.....
.....
.....
.....

IV. Conclusion

Instructor approved student to (not to) defend graduation thesis topic before graduation thesis dissertation committee council.
Evaluation: …… points

Hanoi, August ……2023
Instructor

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF MECHANICAL ENGINEERING

Socialist Republic of Vietnam
Independence-Freedom-Happiness

GRADUATION THESIS EVALUATION

(For Dissertation committee)

Graduation thesis title: Data Science in the Identification of mechanical systems.
Full name: **Nguyễn Huy Công**
Class: **TT.CĐT.01-K62**
Major: Mechatronics

EVALUATION CONTENT

I. Achieved results

.....
.....
.....
.....
.....

II. Drawbacks of graduation thesis

.....
.....
.....
.....
.....
.....
.....

III. Conclusion

The dissertation committee approved the student to (not to) defend the graduation thesis topic before the graduation thesis dissertation committee council.
Evaluation: …… points

Hanoi, August……2023
Dissertation committee

ACKNOWLEDGEMENTS

To complete this graduation project, I firstly would like to gratefully thank all lecturers and teachers at Hanoi University of Science and Technology on the whole, and to all lecturers and teachers of Applied Mechanics subject in particular, who have guided, taught, and supported me useful knowledge throughout all last five years with all their hearts. We especially would like to thank Dr. Nguyen Hai Son, who has guided, taught, and created all opportunities for me relentlessly throughout the time of doing this graduation project. And by this, I wish all of you always have good health, and enthusiasm to guide, teach, and help the next generations of students become good people in our country.

Last but not least, we would like to thank all families and friends who have encouraged, cheered, and given opinions throughout the time of learning and studying and also the time of doing the graduation project.

ABSTRACT

Extracting governing equations from data is a central challenge in many diverse areas of science and engineering. Data are abundant whereas models often remain elusive, as in climate science, neuroscience, ecology, finance, and epidemiology, to name only a few examples. In this work, we combine sparsity-promoting techniques and machine learning with nonlinear dynamical systems to discover governing equations from noisy measurement data. The only assumption about the structure of the model is that there are only a few important terms that govern the dynamics so that the equations are sparse in the space of possible functions; this assumption holds for many physical systems in an appropriate basis. In particular, we use sparse regression to determine the fewest terms in the dynamic governing equations required to accurately represent the data. This results in parsimonious models that balance accuracy with model complexity to avoid overfitting. We demonstrate the algorithm on a wide range of problems, from simple canonical systems, including linear and nonlinear oscillators and the chaotic Lorenz system, to the fluid vortex shedding behind an obstacle.

After a time of researching, and studying, the topic has completed following tasks:

- Learn about the Sparse Regression.
- Understanding the Lasso Regression
- Using data science and machine learning to solve the chaotic Lorenz system.
- Applying the theory to analyze the simple pendulum problem.

Students
Signature

TABLE OF CONTENTS

ACKNOWLEDGEMENTS 5
ABSTRACT 6
CHAPTER 1: GENERAL INTRODUCTION 10
1.1. MOTIVATION 10
1.2. PROBLEM OVERVIEW 10
1.3. SIGNIFICANCE 11
1.4. THESIS CONTENT 11
CHAPTER 2: SPARSE REGRESSION 15
2.1. LINEAR REGRESSION 15
2.1.1. LEAST-SQUARES ESTIMATION 16
2.1.2. PREPROCESSING 17
2.1.3. OVERFITTING 18
2.1.4. THEORETICAL ANALYSIS OF LINEAR REGRESSION 19
2.2. SPARSE REGRESSION 21
2.2.1. MODEL SELECTION 21
2.2.2. BEST SUBSET SELECTION AND FORWARD STEPWISE REGRESSION 21
CHAPTER 3: THE LASSO REGESSION 23
3.1. INTRODUCTION 23
3.1.1. OVERVIEW 23
3.1.2. HISTORY 23
3.2. BASIC FORM 24
3.2.1. LEAST SPARSES 24
3.2.2. ORTHONORMAL COVARIATES 25
3.2.3. CORRELATED COVARIES 26
3.3. LASSO REGRESSION FOR REGULARIZATION 26
CHAPTER 4: APPLICATION OF DATA-DRIVEN DYNAMICAL SYSTEMS FOR LORENZ SYSTEM AND SIMPLE PENDULUM 28
4.1. THE LORENZ SYSTEM 28
4.1.1. OVERVIEW 28
4.1.2. DYNAMICAL SYSTEM 29
4.1.3. SIMULATING THE CHAOTIC LORENZ SYSTEM IN MATLAB 30

4.1.4. SPARSE IDENTIFICATION OF NONLINEAR DYNAMICS 31
4.2. SIMPLE PENDULUM 35
4.2.1. OVERVIEW 35
4.2.2. OSCILLATION OF A SIMPLE PENDULUM 36
4.2.3. SIMPLE PENDULUM SIMULATION 37
4.2.4. THE LASSO FOR SIMPLE PENDULUM 38
CONCLUSION 41
REFERENCES 42

TABLE OF FIGURES

Figure 1.1 The matrix system of equation 12
Figure 1.2 The matrix of health outcome..... 13
Figure 1.3 The health outcome after using LASSO regression 14
Figure 2.1 Relative l_2 -norm error in estimating the response achieved using least-squares regression for different values of n (the number of training data). The training error is plotted in blue, whereas the test error is plotted in red. The green line indicates the training error of the true model used to generate the data. 19
Figure 3.1 The different between Lasso and Ridge Regression 27
Figure 4.1 Chaotic trajectory of the Lorenz system form. 29
Figure 4.2 The classic Lorenz butterfly form simulation..... 31
Figure 4.3 Schematic of the sparse identification of nonlinear dynamics (SINDy) algorithm. Parsimonious models are selected from a library of candidate nonlinear terms using sparse regression. This library $\Theta(X)$ may be constructed purely from measurement data. Modified from Brunton et al. 34
Figure 4.4 The output of the SINDy algorithm..... 35
Figure 4.5 The model of simple pendulum 36
Figure 4.6 The period of oscillation of simple pendulum..... 38
Figure 4.7 The output of Least-Square Regression..... 39
Figure 4.8 The output of LASSO function 40

Chapter 1 GENERAL INTRODUCTION

1.1. MOTIVATION

The world has entered to Industry 4.0, known as the fourth industrial revolutionary, a few years ago with new developments and inventions in science and technology. One of the biggest contributions to this revolutionary is the use of machine learning in all aspects of life. Hence, the study of machine learning and its application of it is vital and creates motivation to design more advanced models and software. This thesis aims to analyze governing equations from data by sparse regression and machine learning from the knowledge learned in HUST and to simulate a simple pendulum model to study that serve further purposes in the future.

1.2. PROBLEM OVERVIEW

Dynamical systems provide a mathematical framework to describe the world around us, modeling the rich interactions between quantities that co-evolve in time. Formally, dynamical systems concern the analysis, prediction, and understanding of the behavior of systems of differential equations or iterative mappings that describe the evolution of the state of a system. This formulation is general enough to encompass a staggering range of phenomena, including those observed in classical mechanical systems, electrical circuits, turbulent fluids, climate science, finance, ecology, social systems, neuroscience, epidemiology, and nearly every other system that evolves in time.

In the past, people have been doing dynamical systems for hundreds of years. We might even say that Newton's Second Law is a dynamical system, it can give us the trajectories of ballistics, and we can also use it to simulate pendulum dynamics or the planetary motion which are examples of dynamical systems. Traditionally, we would take essentially a physics-based approach where we write down Newton's Law, the Lagrangian, or the Hamiltonian for a system and we would derive the equations of motion from first principles using physics. But today, the systems that we actually want to understand like the brain systems, climate science, or the financial market, and there are no first principles physics that we can write down in an easy-to-understand simulate and control framework. There is no master equation for our brain, climate, or financial market but we are increasingly getting more and more data from recording and measuring, and it gives us larger volumes of data that we can use to build models by using machine learning and regression method. We can build satellites, go to the moon with models from first principles but the next generation of problems for the next hundred years like controlling turbulence or understanding the brain, we are not going to be able to use first principles physics models all the time so we have to derive these from data. That is a huge transition as we are going from the first principles to data-driven techniques.

1.3. SIGNIFICANCE

Understanding dynamic constraints and balances in nature has facilitated the rapid development of knowledge and enabled technology, including aircraft, combustion engines, satellites, and electrical power. This work develops a novel framework to discover governing equations underlying a dynamical system simply from data measurements, leveraging advances in sparsity techniques and machine learning. The resulting models are parsimonious, balancing model complexity with descriptive ability while avoiding overfitting. There are many critical data-driven problems, such as understanding cognition from neural recordings, inferring climate patterns, determining the stability of financial markets, predicting and suppressing the spread of disease, and controlling turbulence for greener transportation and energy. With abundant data and elusive laws, data-driven discovery of dynamics will continue to play an important role in these efforts.

1.4. THESIS CONTENT

Advances in machine learning and data science have promised a renaissance in the analysis and understanding of complex data, extracting patterns in vast multimodal data that are beyond the ability of humans to grasp. However, despite the rapid development of tools to understand static data based on statistical relationships, there has been slow progress in distilling physical models of dynamic processes from big data. This has limited the ability of data science models to extrapolate the dynamics beyond the attractor where they were sampled and constructed.

An analogy may be drawn with the discoveries of Kepler and Newton. Kepler, equipped with the most extensive and accurate planetary data of the era, developed a data-driven model for planetary motion, resulting in his famous elliptic orbits. However, this was an attractor-based view of the world, and it did not explain the fundamental dynamic relationships that give rise to planetary orbits, or provide a model for how these bodies react when perturbed. Newton, in contrast, discovered a dynamic relationship between momentum and energy that described the underlying processes responsible for these elliptic orbits. This dynamic model may be generalized to predict behavior in regimes where no data were collected. Newton's model has proven remarkably robust for engineering design, making it possible to land a spacecraft on the moon, which would not have been possible using Kepler's model alone.

A seminal breakthrough by Bongard and Lipson and Schmidt and Lipson has resulted in a new approach to determine the underlying structure of a nonlinear dynamical system from data. This method uses symbolic regression [i.e., genetic programming] to find nonlinear differential equations, and it balances complexity of the model, measured in the number of terms, with model accuracy. The resulting model identification realizes a long-sought goal of the physics and engineering communities to discover dynamical systems from data. However, symbolic

regression is expensive, does not scale well to large systems of interest, and may be prone to overfitting unless care is taken to explicitly balance model complexity with predictive power.

Almost all the matters in data science, it is down to a math problem, as we have seen before. The regression problem often be written as some matrix system of equation:

$$Ax = b \qquad \textit{Eq 1.1}$$

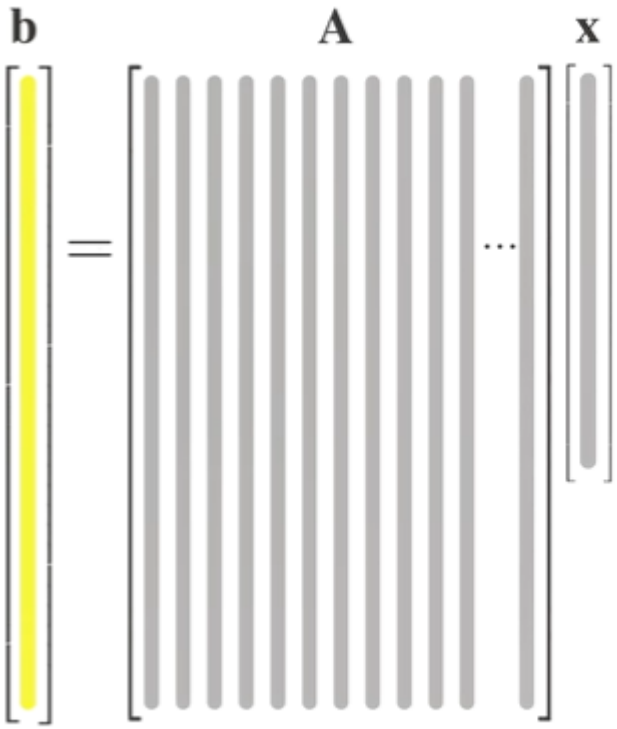

Figure 1.1 The matrix system of equation

Where we have the measurement of vector *b*, we also know what matrix *A* is and we are trying to solve for some unknown *x* vector. And that are regression problem would be. But in many cases, this matrix system of equations will not have exactly the same number of equations as the unknown variable in *x*. In general, we are not expecting to find a single unique *x* vector that exactly satisfies all the equations in *A* matrix because there are more equations than the unknown.

For more detail performance, I am going to use the example of health outcome.

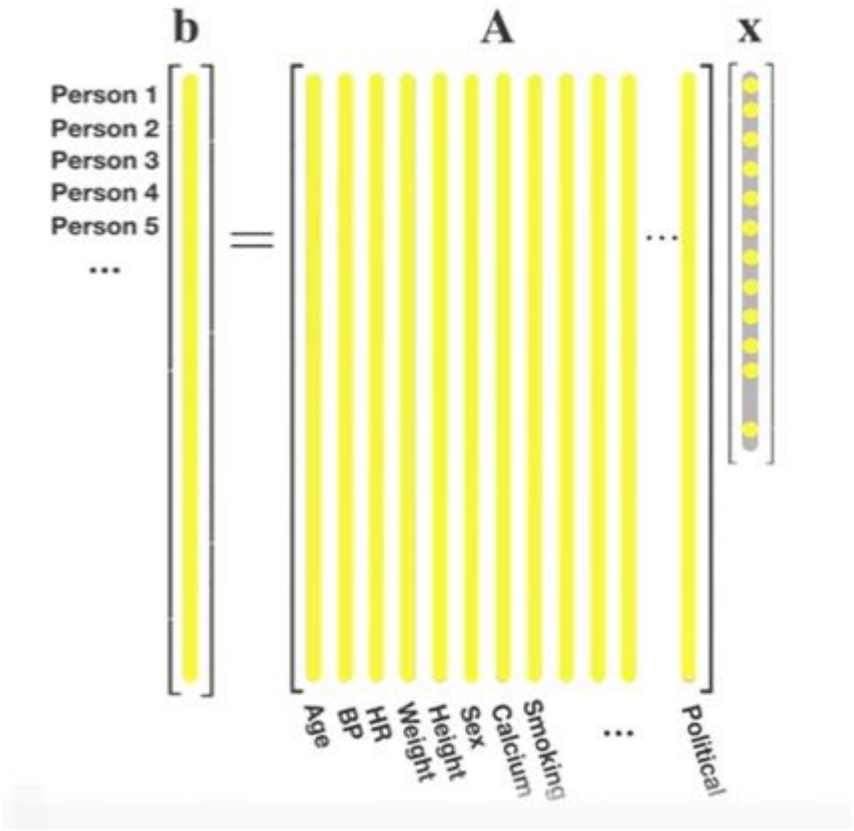

Figure 1.2 The matrix of health outcome

Let say we have measurement of hundreds of thousand people. So the *b* vector can be the probability of developing heart disease, and the column of *A* will be descripted features that we measure for all of the individual people that we have collected data (Their Age, Height, Weight, Blood Pressure, etc.). Some of these absolutely be relevant for the health outcome of developing heart disease, while others might be less relevant (Their Political, Pet, Financial Capability, etc.). All the information that we have collected, but many of them are not really matter with the heath outcome, so this is the big matrix regression problem.

The basic idea here is we are trying to do the regression to solve for *x* to satisfy the equation: $Ax = b$, and that is where the LASSO comes in. We are trying to minimize the error of the fit, we still want a good model fit $Ax = b$, but we want *x* to be as sparse as possible and have many zero entries as possible. The LASSO regression has a form:

$$L = \|Ax - b\|_2 + \lambda \|x\|_1 \qquad \textit{Eq 1.2}$$

And this does a pretty amazing thing. If you look at the model, you will get when you run the LASSO regression.

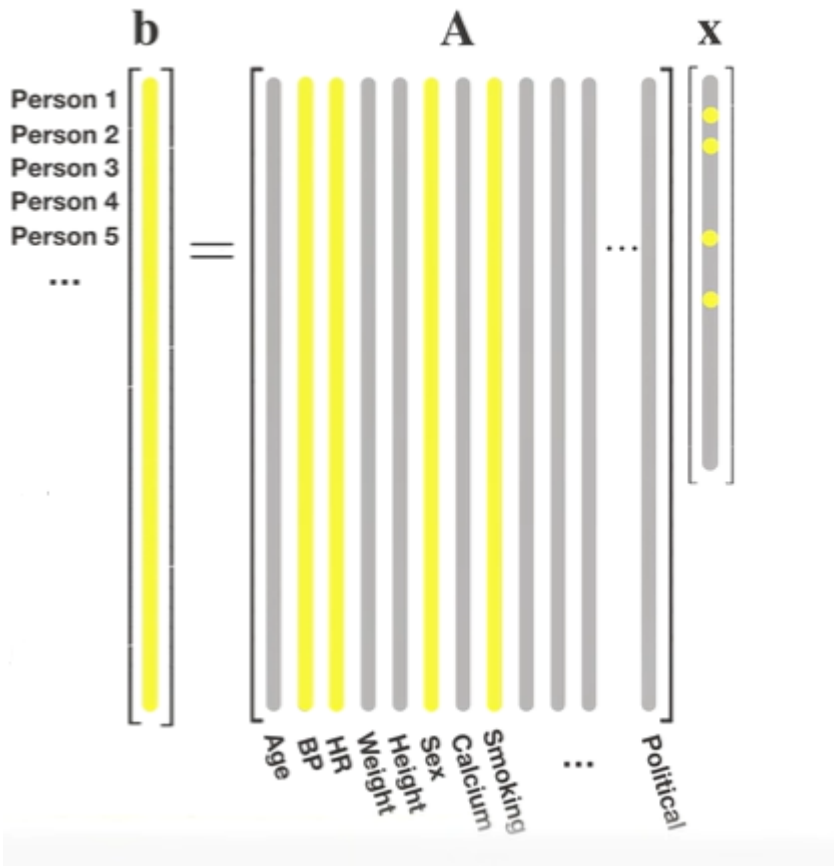

Figure 1.3 The health outcome after using LASSO regression

When you run the LASSO regression, you will find that *x* vector is very sparse and it highlights only a few columns of the *A* matrix that are most correlated with your desired health outcome which in this case. And it might pull out factors such as blood pressure, heart rate, if you are a smoker, etc. So the basic idea is that the LASSO regression will highlight a few columns of the *A* matrix, and it has really big impact in your resulting model. And the LASSO regression also prevents you from overfitting because it is only selecting variable factors that are most important to the desired outcomes.

Chapter 2 SPARSE REGRESSION

2.1. LINEAR REGRESSION

In statistics, the problem of regression is that of learning a function that allows to estimate a certain quantity of interest, the response or dependent variable, from several observed variables, known as covariates, features or independent variables. For example, we might be interested in estimating the price of a house based on its extension, the number of rooms, the year it was built, etc. The function that models the relation between the response and the predictors is learned from training data and can then be used to predict the response for new examples.

In linear regression, we assume that the response is well modeled as a linear combination of the predictors. The model is parametrized by an intercept $\beta_0 \in \mathbb{R}$ and a vector of weights $\beta \in \mathbb{R}^p$, where p is the number of predictors. Let us assume that we have n data points consisting of a value of the response y_i and the corresponding values of the predictors $X_{i1}, X_{i2}, \ldots, X_{ip}$, $1 \leq i \leq n$. The linear model is given by:

$$y_i \approx \beta_0 + \sum_{j=1}^{p} \beta_j X_{ij}, \quad 1 \leq i \leq n. \qquad \textit{Eq 2.1}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix} \approx \begin{bmatrix} 1 \\ 1 \\ \ldots \\ 1 \end{bmatrix} \beta_0 + \begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1p} \\ X_{21} & X_{22} & \ldots & X_{2p} \\ & & \ldots & \\ X_{n1} & X_{n2} & \ldots & X_{np} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \ldots \\ \beta_p \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \ldots \\ 1 \end{bmatrix} \beta_0 + X\beta \qquad \textit{Eq 2.2}$$

We have already encountered linear models that arise in inverse problems such as compressed sensing or super-resolution. A major difference between statistics and those applications is that in statistics, the ultimate aim is to predict y accurately for new values of the predictors, not to estimate β. The role of β is merely to quantify the linear relation between the response and the predictors. In contrast, when solving an inverse problem the main objective is to determine β, which has a physical interpretation: an image of a 2D slice of a body in MRI, the spectrum of multi-sinusoidal signal in spectral super-resolution, reflection coefficients of strata in seismography, etc.

2.1.1. LEAST-SQUARES ESTIMATION

To calibrate the linear regression model, we estimate the weight vector from the training data. By far the most popular method to compute the estimate is to minimize the l_2 norm of the fitting error on the training set. In more detail, the weight estimate β_{ls} is the solution of the least-squares problem:

$$\text{minimize} \quad \left\| y - X\tilde{\beta} \right\|_2 \qquad \textit{Eq 2.3}$$

The least-squares cost function is convenient from a computational view, since it is convex and can be minimized efficiently (in fact, as we will see in a moment it has a closed-form solution). In addition, as detailed in ***Proposition 1.3*** below, it has a reasonable probabilistic interpretation. The following proposition, proved in ***Section 2.1.1***, shows that the solution to the least-squares problem has a closed-form solution.

Proposition 1.1 (Least-squares solution). For $p \geq n$, if X is full rank then the solution to the least-squares problem is:

$$\beta_{ls} := \left(X^T X \right)^{-1} X^T y \qquad \textit{Eq 2.4}$$

A corollary to this result provides a geometric interpretation for the least-squares estimate of y: it is obtained by projecting the response onto the column space of the matrix formed by the predictors.

Corollary 1.2. For $p \geq n$, if X is full rank then $X\beta_{ls}$ is the projection of y onto the column space of X.

Let $X = U\Sigma V^T$ be the singular-value decomposition of X. Since X is full rank and $p \geq n$ we have $U^TU = I$, $V^TV = I$ and Σ is a square invertible matrix, which implies

$$X\beta_{ls} = X\left(X^T X \right)^{-1} X^T y \qquad \textit{Eq 2.5}$$

$$= U\sum V^T \left(V\sum U^T U\sum V^T \right) V\sum U^T y \qquad \textit{Eq 2.6}$$

$$= UU^T y. \qquad \textit{Eq 2.7}$$

Proposition 1.3 (Least-squares solution as maximum-likelihood estimate). If we model y *as*

$$y = X\beta + z \qquad \textit{Eq 2.8}$$

where $X \in \mathbb{R}^{n*p}$, $p \geq n$, $\beta \in \mathbb{R}^p$ and $y \in \mathbb{R}^n$ are fixed and the entries of $z \in \mathbb{R}^n$ are iid Gaussian random variables with mean zero and the same variance, then the maximum-likelihood estimate of β given y is equal to β_{ls}.

2.1.2. PREPROCESSING

The careful reader might notice that we have not explained how to fit the intercept β_0. Before fitting a linear regression model, we typically perform the following preprocessing steps.

1. Centering each predictor column X_i subtracting its mean

$$\mu_j := \sum_{i=1}^{n} X_{ij} \qquad \text{Eq 2.9}$$

so that each column of X has mean zero.

2. Normalizing each predictor column X_j by dividing by

$$\sigma_j := \sqrt{\sum_{i=1}^{n} \left(X_{ij} - \mu_j \right)^2} \qquad \text{Eq 2.10}$$

so that each column of X has l_2 norm equal to one. The objective is to make the estimate invariant to the units used to measure each predictor.

3. Centering the response vector y by subtracting its mean

$$\mu_y := \sum_{i=1}^{n} y_i \qquad \text{Eq 2.11}$$

By the following lemma, the intercept of the linear model once we have centered the data is equal to zero.

Lemma 1.4 (Intercept). If the mean of y and of each of the columns of X is zero, then the intercept in the least-squares solution is also zero.

Once have solved the least-squares problem using the centered and normalized data to obtain β_{ls}, we can use the model to estimate the response corresponding to a new vector of predictors $x \in \mathbb{R}^p$ by computing

$$f(x) := \mu_y + \sum_{i=1}^{p} \beta_{ls,i} \frac{x_i - \mu_i}{\sigma_i} \qquad \text{Eq 2.12}$$

2.1.3. OVERFITTING

Imagine that a friend tells you:

I found a cool way to predict the temperature in New York: It's just a linear combination of the temperature in every other state. I fit the model on data from the last month and a half and it's perfect!

Your friend is not lying, but the problem is that she is using a number of data points to fit the linear model that is roughly the same as the number of parameters. If $n = p$ we can find a β such that $y = X\beta$ exactly, even if y and X have nothing to do with each other! This is called overfitting and is usually caused by using a model that is too flexible with respect to the number of data that are available. To evaluate whether a model suffers from overfitting we separate the data into a training set and a test set. The training set is used to fit the model and the test set is used to evaluate the error. A model that overfits the training set will have a very low error when evaluated on the training examples but will not generalize well to the test examples. Figure 1 shows the result of evaluating the training error and the test error of a linear model with $p = 50$ parameters fitted from n training examples. The training and test data are generated by fixing a vector of weights β and then computing

$$y_{train} = X_{train}\beta + z_{train} \qquad \text{Eq 2.13}$$

$$y_{test} = X_{test}\beta \qquad \text{Eq 2.14}$$

where the entries of X_{train}, X_{test}, z_{train} and β are sampled independently at random from a Gaussian distribution with zero mean and unit variance. The training and test errors are defined as

$$error_{train} = \frac{\left\| X_{train}\beta_{ls} - y_{train} \right\|_2}{\left\| y_{train} \right\|_2} \qquad \text{Eq 2.15}$$

$$error_{test} = \frac{\left\| X_{test}\beta_{ls} - y_{test} \right\|_2}{\left\| y_{test} \right\|_2} \qquad \text{Eq 2.16}$$

Note that even the true β does not achieve zero training error because of the presence of the noise, but the test error is actually zero if we manage to estimate β exactly. The training error of the linear model grows with n. This makes sense as the model has to fit more data using the same number of parameters. When n is

close to p (50), the fitted model is much better than the true model at replicating the training data (the error of the true model is shown in green). This is a sign of overfitting: the model is adapting to the noise and not learning the true linear structure. Indeed, in that regime the test error is extremely high. At larger n, the training error rises to the level achieved by the true linear model and the test error decreases, indicating that we are learning the underlying model.

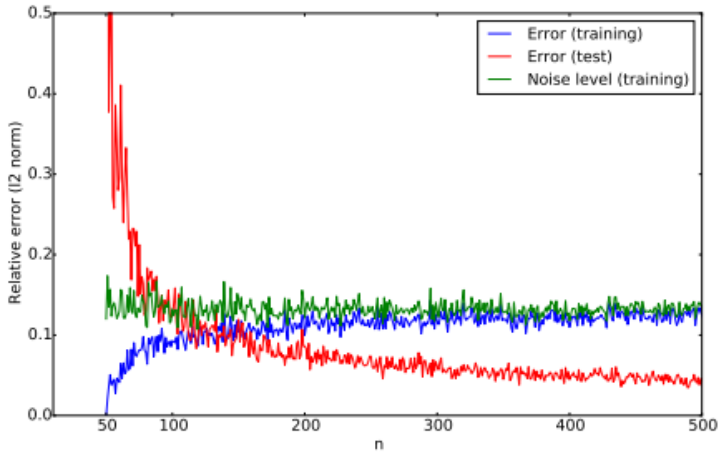

Figure 2.1 Relative l_2 -norm error in estimating the response achieved using least-squares regression for different values of n *(the number of training data). The training error is plotted in blue, whereas the test error is plotted in red. The green line indicates the training error of the true model used to generate the data.*

2.1.4. THEORETICAL ANALYSIS OF LINEAR REGRESSION

In this section we analyze the solution of the least-squares regression fit to understand its dependence with the number of training examples. The following theorem, characterizes the error in estimating the weights under the assumption that the data are indeed generated by a linear model with additive noise.

Theorem 1.5. Assume the data y are generated according to a linear model with additive noise,

$$y = X\beta + z \qquad \textit{Eq 2.17}$$

where $X \in \mathbb{R}^{n*p}$ and $\beta \in \mathbb{R}^{p}$, and that the entries of $z \in \mathbb{R}^{n}$ are drawn independently at random from a Gaussian distribution with zero mean and variance σ_{z^2} . The least-squares estimate

$$\beta_{ls} := \arg\min_{\beta} \left\| y - X\tilde{\beta} \right\|_2 \qquad \textit{Eq 2.18}$$

satisfies

$$\frac{p\sigma_z^2(1-\varepsilon)}{\sigma_{\max}^2} \leq \left\|\beta - \beta_{ls}\right\|_2^2 \leq \frac{p\sigma_z^2(1+\varepsilon)}{\sigma_{\min}^2}$$ *Eq 2.19*

with probability $1 - 2\exp(-\frac{p\varepsilon^2}{8})$. $\sigma_{\min}$ and $\sigma_{\max}$ *denote the smallest and largest singular value of X respectively.*

The bounds in the theorem are in terms of the singular values of the predictor matrix $X \in \mathbb{R}^{n*p}$. To provide some intuition as to the scaling of these singular values when we fix *p* and increase *n*, let us assume that the entries of *X* are drawn independently at random from a standard Gaussian distribution. Then both $\sigma_{\min}$ and $\sigma_{\max}$ are close to $\sqrt{n}$ with high probability as long as $n > C_p$ for some constant *C*. This implies that if the variance of the noise *z* equals one,

$$\left\|\beta - \beta_{ls}\right\|_2 \approx \sqrt{\frac{p}{n}}$$ *Eq 2.20*

If each of the entries of β has constant magnitude the l_2 norm of β is approximately equal to $\sqrt{p}$. In this case, the theoretical analysis predicts that the normalized error

$$\frac{\left\|\beta - \beta_{ls}\right\|_2}{\left\|\beta\right\|_2} \approx \frac{1}{\sqrt{n}}$$ *Eq 2.21*

the entries of *X,* β and *z* are all generated by sampling independently from standard Gaussian random variables. The relative error scales precisely as $1/\sqrt{n}$.

For a fixed number of predictors, this analysis indicates that the least-squares solution converges to the true weights as the number of data *n* grows. In statistics jargon, the estimator is *consistent*. The result suggests two scenarios in which the least-squares estimator may not yield a good estimate: when *p* is of the same order as *n* and when some of the predictors are highly correlated, as some of the singular values of *X* may be very small.

2.2. SPARSE REGRESSION

2.2.1. MODEL SELECTION

We establish that linear regression allows to learn a linear model when the number of available examples n is large with respect to the number of predictors *p*. However, in many modern applications, the number of predictors can be extremely large. An example is computational genomics, where the predictors may correspond to gene-expression measurements from thousands of genes, whereas *n* is the number of patients which might only be in the hundreds. It is obviously impossible to fit a linear model when $p > n$, or even when $p \approx n$ without overfitting (depending on the noise level), but it may still be possible to fit a sparse linear model that only depends on a subset of $s < p$ predictors. Selecting the relevant predictors to include in the model is called *model selection* in statistics.

2.2.2. BEST SUBSET SELECTION AND FORWARD STEPWISE REGRESSION

A possible way to select a small number of relevant predictors from a training set is to fix the order of the sparse model $s < p$ and then evaluate the least-squares fit of all possible *s*-sparse models in order to select the one that provides the best fit. This is called the best-subset selection method. Unfortunately it is computationally intractable even for small values of *s* and *p*. For instance, there are more than 1013 possible models if $s = 10$ and $p = 100$. An alternative to an exhaustive evaluation of all possible sparse models is to select the predictors greedily in the spirit of signal-processing methods such as orthogonal matching pursuit. In forward stepwise regression we select the predictor that is most correlated with the response and then project the rest of predictors onto its orthogonal complement. Iterating this procedure allows to learn an *s*-sparse model in *s* steps.

Algorithm 2.1 (Forward stepwise regression). Given a matrix of predictors $X \in \mathbb{R}^{n*p}$ and a response $y \in \mathbb{R}^n$, we initialize the residual and the subset of relevant predictors *S* by setting,

$$j_0 := \arg\max \left| \langle y, X_j \rangle \right| \qquad \textit{Eq 2.22}$$

$$S_0 := \{ j_0 \} \qquad \textit{Eq 2.23}$$

$$\beta_{ls} := \arg\min \left\| y - X_{S_0} \tilde{\beta} \right\|_2 \qquad \textit{Eq 2.24}$$

$$r^{(0)} := y - X_{S_0} \beta_{ls} \qquad \textit{Eq 2.25}$$

Then for k = 2, 3, . . . , s we compute

$$j_k := \arg\max_{j \notin S_{j-1}} \left| \left\langle y, P_{col(X_{S_{j-1}}} X_j \right\rangle \right| \qquad \text{Eq 2.26}$$

$$S_j := S_{j-1} \cup \{ j_k \} \qquad \text{Eq 2.27}$$

$$\beta_{ls} := \arg\min_{\tilde{\beta}} \left\| y - X_{S_j} \tilde{\beta} \right\|_2 \qquad \text{Eq 2.28}$$

$$r^{(0)} := y - X_{S_0} \beta_{ls} \qquad \text{Eq 2.29}$$

The algorithm is very similar to orthogonal matching pursuit. The only difference is the orthogonalization step in which we project the remaining predictors onto the orthogonal complement of the span of the predictors that have been selected already.

Chapter 3 THE LASSO REGESSION

3.1. INTRODUCTION

3.1.1. OVERVIEW

In statistics and machine learning, Lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. If a model uses the **L1 regularization** technique, then it is called lasso regression. It was originally introduced in geophysics, and later by Robert Tibshirani, who coined the term.

Lasso was originally formulated for linear regression models. This simple case reveals a substantial amount about the estimator. These include its relationship to ridge regression and best subset selection and the connections between lasso coefficient estimates and so-called soft thresholding. It also reveals that (like standard linear regression) the coefficient estimates do not need to be unique if covariates are colliner.

Though originally defined for linear regression, lasso regularization is easily extended to other statistical models including generalized linear models, generalized estimating equations, proportional hazards models, and M-estimators. Lasso's ability to perform subset selection relies on the form of the constraint and has a variety of interpretations including in terms of geometry, Bayesian statistics and convex analysis.

The LASSO is closely related to basis pursuit denoising.

3.1.2. HISTORY

Lasso was introduced in order to improve the prediction accuracy and interpretability of regression models. It selects a reduced set of the known covariates for use in a model. Lasso was developed independently in geophysics literature in 1986, based on prior work that used the ℓ_1 penalty for both fitting and penalization of the coefficients. Statistician Robert Tibshirani independently rediscovered and popularized it in 1996, based on Breiman's nonnegative garrote.

Prior to lasso, the most widely used method for choosing covariates was stepwise selection. That approach only improves prediction accuracy in certain cases, such as when only a few covariates have a strong relationship with the outcome. However, in other cases, it can increase prediction error. At the time, ridge regression was the most popular technique for improving prediction accuracy. Ridge regression improves prediction error by shrinking the sum of the squares of the regression coefficients to be less than a fixed value in order to reduce overfitting, but it does not perform covariate selection and therefore does not help to make the model more interpretable.

Lasso achieves both of these goals by forcing the sum of the absolute value of the regression coefficients to be less than a fixed value, which forces certain coefficients to zero, excluding them from impacting prediction. This idea is similar to ridge regression, which also shrinks the size of the coefficients; however, ridge regression does not set coefficients to zero (and, thus, does not perform variable selection).

3.2. BASIC FORM

3.2.1. LEAST SPARSES

Consider a sample consisting of *N* cases, each of which consists of *p* covariates and a single outcome. Let y_i be the outcome and $x_i := (x_1, x_2, ..., x_p)_i^T$ be the covariate vector for the i^{th} case. Then the objective of lasso is to solve

$$\min_{\beta_0, \beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - x_i^T \beta \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \left| \beta_j \right| \le t \qquad \textit{Eq 3.1}$$

Here β_0 is the constant coefficient, $\beta := (\beta_1, \beta_2, ..., \beta_p)$ is the coefficient vector, and t is a prespecified free parameter that determines the degree of regularization. Letting X be the covariate matrix, so that $X_{ij} = (x_i)_j$ and x_i^T is the i^{th} row of X, the expression can be written more compactly as

$$\min_{\beta_0, \beta} \left\{ \left\| y - \beta_0 - X\beta \right\|_2^2 \right\} \text{ subject to } \left\| \beta \right\|_1 \le t \qquad \textit{Eq 3.2}$$

where $\left\| u \right\|_p = (\sum_{i=1}^{N} |u_i|^p)^{1/p}$ is the standard ℓ^p norm.

Denoting the scalar mean of the data points x_i by $\bar{x}$ and the mean of the response variables y_i by $\bar{y}$, the resulting estimate for β_0 is $\hat{\beta}_0 = \bar{y} - \bar{x}^T \beta$, so that

$$y_i - \hat{\beta}_0 - x_i^T \beta = y_i - \left(\bar{y} - \bar{x}^T \beta \right) - x_i^T \beta = \left(y_i - \bar{y} \right) - \left(x_i - \bar{x} \right)^T \beta \qquad \text{Eq 3.3}$$

and therefore it is standard to work with variables that have been made zero-mean. Additionally, the covariates are typically standardized ($\sum_{i=1}^{N} x_i^2 = 1$) so that the solution does not depend on the measurement scale.

It can be helpful to rewrite

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 \right\} \text{ subject to } \|\beta\|_1 \leq t \qquad \textit{Eq 3.4}$$

in the so-called Lagrangian form

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\} \qquad \textit{Eq 3.5}$$

where the exact relationship between t and λ is data dependent.

3.2.2. ORTHONORMAL COVARIATES

Some basic properties of the lasso estimator can now be considered.

Assuming first that the covariates are orthonormal so that $x_i^T x_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta, or, equivalently, $X^T X = I$, then using subgradient methods it can be shown that

$$\hat{\beta}_j = S_{N\lambda}\left(\hat{\beta}^{OLS}\right) = \hat{\beta}_j^{OLS} \max\left(0, 1 - \frac{N\lambda}{\left|\hat{\beta}_j^{OLS}\right|}\right) \qquad \textit{Eq 3.6}$$

$$\text{where } \hat{\beta}^{OLS} = \left(X^T X\right)^{-1} X^T y = X^T y$$

S_α is referred to as the *soft thresholding operator*, since it translates values towards zero (making them exactly zero if they are small enough) instead of setting smaller values to zero and leaving larger ones untouched as the *hard thresholding operator*, often denoted H_α.

In ridge regression the objective is to minimize

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \right\} \qquad \textit{Eq 3.7}$$

Using $X^T X = I$ and the ridge regression formula: $\hat{\beta} = ((X^T X) + N\lambda I)^{-1} X^T y$ it yields:

$$\hat{\beta}_j = \left(1 + N\lambda\right)^{-1} \hat{\beta}_j^{OLS} \qquad \textit{Eq 3.8}$$

Ridge regression shrinks all coefficients by a uniform factor of $(1 + N\lambda)^{-1}$ and does not set any coefficients to zero.

It can also be compared to regression with best subset selection, in which the goal is to minimize

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_0 \right\} \qquad \text{Eq 3.9}$$

3.2.3. CORRELATED COVARIES

In one special case two covariates, say j and k, are identical for each observation, so that $x_{(j)} = x_{(k)}$, where $x_{(j),i} = x_{(k),i}$. Then the values of β_j and β_k that minimize the lasso objective function are not uniquely determined. In fact, if some $\hat{\beta}$ in which $\hat{\beta}_j \hat{\beta}_k \geq 0$, then if $s \in [0,1]$ replacing $\hat{\beta}_j$ by $s\left(\hat{\beta}_j + \hat{\beta}_k\right)$ and $\hat{\beta}_k$ by $(1-s)(\hat{\beta}_j + \hat{\beta}_k)$, while keeping all the other $\hat{\beta}_i$ fixed, gives a new solution, so the lasso objective function then has a continuum of valid minimizers. Several variants of the lasso, including the Elastic net regularization, have been designed to address this shortcoming.

3.3. LASSO REGRESSION FOR REGULARIZATION

In this shrinkage technique, the coefficients determined in the linear model from which are shrunk towards the central point as the mean by introducing a penalization factor called the alpha α (or sometimes lambda) values.

$$L_{lasso}\left(\hat{\beta}\right) = \sum_{i=1}^{n}\left(y_i - x_i\hat{\beta}\right)^2 + \lambda\sum_{j=1}^{m}\left|\beta_j\right| \qquad \text{Eq 3.10}$$

Alpha (α) is the penalty term that denotes the amount of shrinkage (or constraint) that will be implemented in the equation. With alpha set to zero, you will find that this is the equivalent of the linear regression model, and a larger value penalizes the optimization function. Therefore, lasso regression shrinks the coefficients and helps to reduce the model complexity and multi-collinearity.

Alpha (α) can be any real-valued number between zero and infinity; the larger the value, the more aggressive the penalization is.

Lasso Regression for Model Selection

Due to the fact that coefficients will be shrunk towards a mean of zero, less important features in a dataset are eliminated when penalized. The shrinkage of these coefficients based on the alpha value provided leads to some form of automatic feature selection, as input variables are removed in an effective approach.

Ridge Regression

Similar to the lasso regression, ridge regression puts a similar constraint on the coefficients by introducing a penalty factor. However, while lasso regression takes the magnitude of the coefficients, ridge regression takes the square.

$$L_{hridge}\left(\hat{\beta}\right) = \sum_{i=1}^{n}\left(y_i - x_i\hat{\beta}\right)^2 + \lambda\sum_{j=1}^{m}\omega_j\hat{\beta}_j^2 \qquad \textit{Eq 3.11}$$

Ridge regression is also referred to as **L2 Regularization**.

The reason that Lasso can be Used for Model Selection, but not Ridge Regression

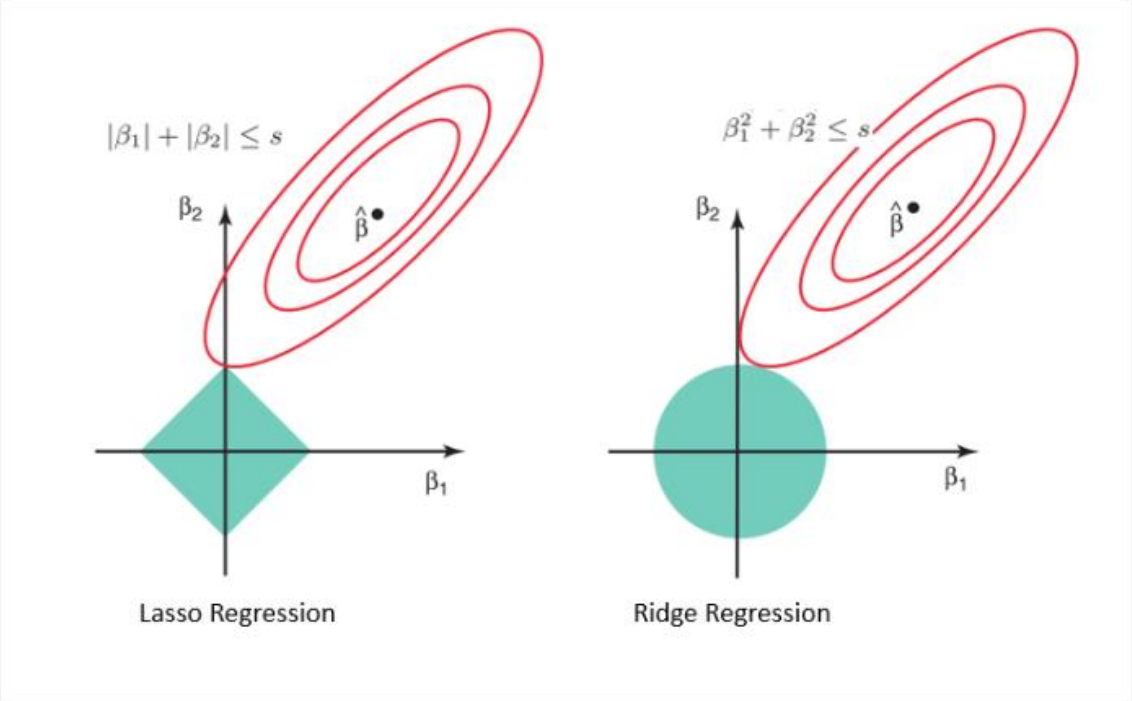

Figure 3.1 The different between Lasso and Ridge Regression

Considering the geometry of both the lasso (left) and ridge (right) models, the elliptical contours (red circles) are the cost functions for each. Relaxing the constraints introduced by the penalty factor leads to an increase in the constrained region (diamond, circle). Doing this continually, we will hit the center of the ellipse, where the results of both lasso and ridge models are similar to a linear regression model.

However, both methods determine coefficients by finding the first point where the elliptical contours hit the region of constraints. Since lasso regression takes a diamond shape in the plot for the constrained region, each time the elliptical regions intersect with these corners, at least one of the coefficients becomes zero. This is impossible in the ridge regression model as it forms a circular shape and therefore values can be shrunk close to zero, but never equal to zero.

Chapter 4 APPLICATION OF DATA-DRIVEN DYNAMICAL SYSTEMS FOR LORENZ SYSTEM AND SIMPLE PENDULUM

4.1. THE LORENZ SYSTEM

4.1.1. OVERVIEW

The **Lorenz system** is a system of ordinary differential equations first studied by mathematician and meteorologist Edward Lorenz. It is notable for having chaotic solutions for certain parameter values and initial conditions. In particular, the Lorenz attractor is a set of chaotic solutions of the Lorenz system. In popular media the "butterfly effect" stems from the real-world implications of the Lorenz attractor, namely that several different initial chaotic conditions evolve in phase space in a way that never repeats, so all chaos is unpredictable. This underscores that chaotic systems can be completely deterministic and yet still be inherently unpredictable over long periods of time. Because chaos continually increases in systems, we cannot predict the future of systems well. E.g., even the small flap of a butterfly's wings could set the world on a vastly different trajectory, such as by causing a hurricane. The shape of the Lorenz attractor itself, when plotted in phase space, may also be seen to resemble a butterfly.

In 1963, Edward Lorenz, with the help of Ellen Fetter who was responsible for the numerical simulations and figures, and Margaret Hamilton who helped in the initial, numerical computations leading up to the findings of the Lorenz model, developed a simplified mathematical model for atmospheric convection. The model is a system of three ordinary differential equations now known as the Lorenz equations:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(p - z) - y \\ \dot{z} &= xy - \beta z \end{aligned} \qquad \textit{Eq 4.1}$$

The equations relate the properties of a two-dimensional fluid layer uniformly warmed from below and cooled from above. In particular, the equations describe the rate of change of three quantities with respect to time: x is proportional to the rate of convection, y to the horizontal temperature variation, and z to the vertical temperature variation. The constants σ, p, and β are system parameters proportional to the Prandtl number, Rayleigh number, and certain physical dimensions of the layer itself.

The Lorenz equations can arise in simplified models for lasers, dynamos, thermosyphons, brushless DC motors, electric circuits, chemical reactions and forward osmosis. The Lorenz equations are also the governing equations in Fourier

space for the Malkus waterwheel. The Malkus waterwheel exhibits chaotic motion where instead of spinning in one direction at a constant speed, its rotation will speed up, slow down, stop, change directions, and oscillate back and forth between combinations of such behaviors in an unpredictable manner.

From a technical standpoint, the Lorenz system is nonlinear, aperiodic, three-dimensional and deterministic. The Lorenz equations have been the subject of hundreds of research articles, and at least one book-length study.

4.1.2. DYNAMICAL SYSTEM

Throughout this chapter, we will consider dynamical systems of the form

$$\frac{d}{dt}x(t) = f\left(x(t);t;\beta\right) \qquad \text{Eq 4.2}$$

where x is the state of the system and f is a vector field that possibly depends on the state x, time t, and a set of parameters β.

Consider the Lorenz equations:

$$\begin{aligned}\dot{x} &= \sigma\left(y-x\right)\\ \dot{y} &= x\left(p-z\right)-y\\ \dot{z} &= xy-\beta z\end{aligned} \qquad \text{Eq 4.3}$$

For example, let the parameters σ = 10, ρ = 28, and β = 8/3. A trajectory of the Lorenz system is shown in Fig. 4.1. In this case, the state vector is $x=[x \quad y \quad z]^T$ and the parameter vector is $\beta=[\sigma \quad p \quad \beta]^T$.

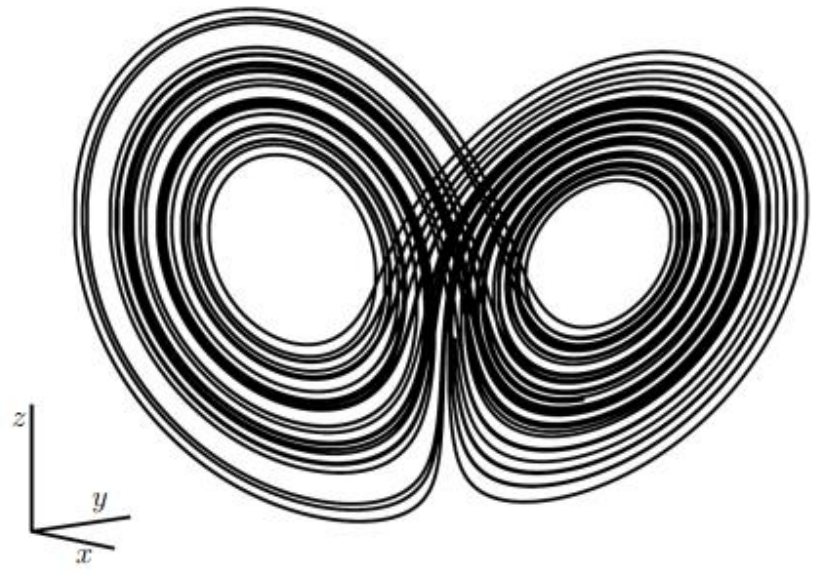

Figure 4.1 Chaotic trajectory of the Lorenz system form.

The Lorenz system is among the simplest and most well-studied dynamical systems that exhibit chaos, which is characterized as a sensitive dependence on initial conditions. Two trajectories with nearby initial conditions will rapidly diverge in behavior, and, after long times, only statistical statements can be made.

4.1.3. SIMULATING THE CHAOTIC LORENZ SYSTEM IN MATLAB

We will consider the chaotic case of Lorenz system where the parameters σ = 10, ρ = 28, and β = 8/3.

It is simple to simulate dynamical systems, such as the Lorenz system. First, the vector field $f(x,t,\beta)$ is defined in the function Lorenz in Code 4.1

```
function dx = lorenz(t,x,Beta)
dx = [
Beta(1)*(x(2)-x(1));
x(1)*(Beta(2)-x(3))-x(2);
x(1)*x(2)-Beta(3)*x(3);
];
```

Code 4.1 Define Lorenz vector field.

In Code 4.2, we define the system parameters β, initial condition x_0, and timespan, and simulate the equations with a fourth-order Runge–Kutta integration scheme with adaptive time-step; in MATLAB we use the **ode45** command.

```
clear all, close all, clc

Beta = [10; 28; 8/3]; % Lorenz's parameters (chaotic)

x0=[0; 1; 20];  % Initial condition
dt = 0.001;
tspan=dt:dt:50;
options = odeset('RelTol',1e-12,'AbsTol',1e-12*ones(1,3));

[t,x]=ode45(@(t,x) lorenz(t,x,Beta),tspan,x0,options);
plot3(x(:,1),x(:,2),x(:,3));
```

Code 4.2 Define Lorenz system parameters and simulate with Runge–Kutta integrator.

We will often consider the simpler case of an autonomous system without time dependence or parameters:

$$\frac{d}{dt}x(t) = f\left(x(t)\right) \qquad \text{Eq 4.4}$$

In general, $x(t) \in M$ is an *n*-dimensional state that lives on a smooth manifold *M*, and *f* is an element of the tangent bundle *TM* of *M* so that $f\left(x(t)\right) \in T_{x(t)}M$ However, we will typically consider the simpler case where x is a vector, $M = \mathbb{R}^n$, and *f* is a Lipschitz continuous function, guaranteeing existence and uniqueness of solutions to (Eq 4.4) .

And that is all we have done, we have defined a right-hand side vector field, we put in our parameters, initial condition *dt* in the time span (*tspan*), and we have given some *ode* options so we can dial down the tolerance so it is very accurate.

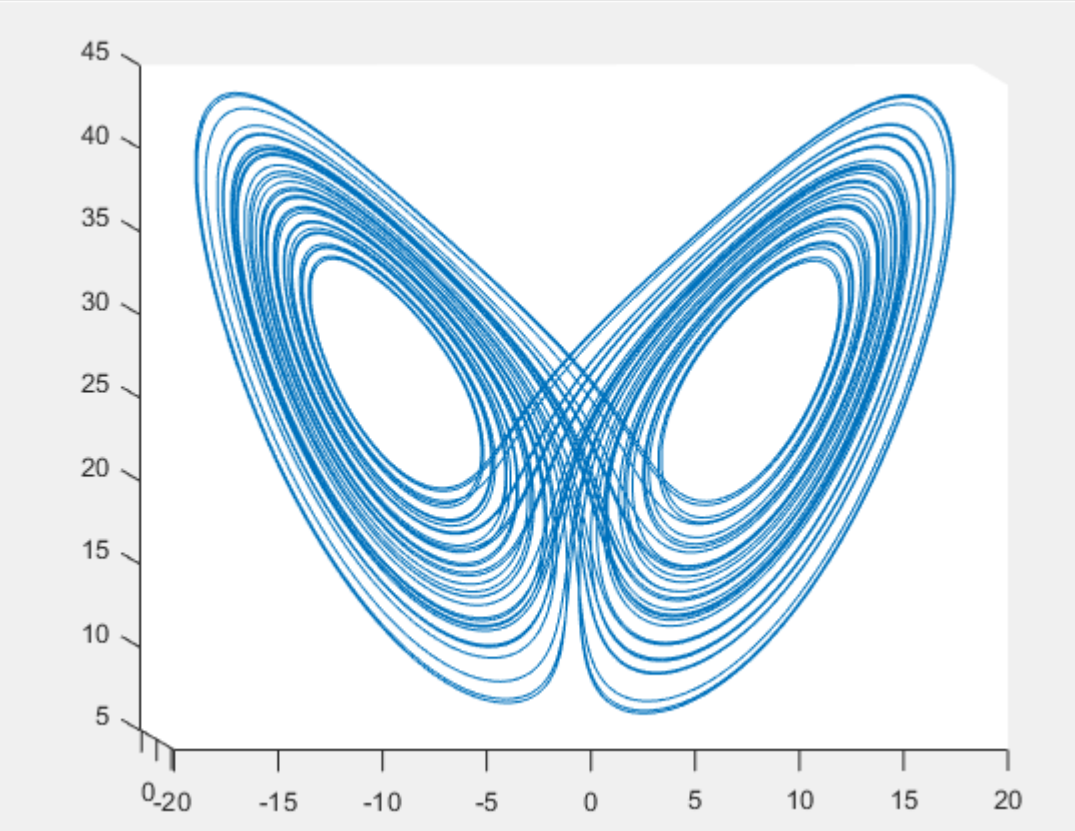

Figure 4.2 The classic Lorenz butterfly form simulation.

According to this chapter, we knew this Lorenz system is the chaotic dynamical system, so what happen is any uncertainly in our initial conditions grows exponentially in time and eventually mixes on to this butterfly attractor. These initial conditions are integrating through, stretching and folding through these dynamics.

4.1.4. SPARSE IDENTIFICATION OF NONLINEAR DYNAMICS

Discovering dynamical systems models from data is a central challenge in mathematical physics, with a rich history going back at least as far as the time of Kepler and Newton and the discovery of the laws of planetary motion. Historically, this process relied on a combination of high-quality measurements and expert intuition. With vast quantities of data and increasing computational power, the *automated* discovery of governing equations and dynamical systems is a new and exciting scientific paradigm.

Typically, either the form of a candidate model is constrained via prior knowledge of the governing equations, or a handful of heuristic models are tested and parameters are optimized to fit data. Alternatively, best-fit linear models may be obtained using DMD or ERA. Simultaneously identifying the nonlinear structure and parameters of a model from data is considerably more challenging, as there are combinatorically many possible model structures. The sparse identification of nonlinear dynamics (SINDy) algorithm bypasses the intractable combinatorial search through all possible model structures, leveraging the fact that many dynamical systems

$$\frac{d}{dt}x = f(x) \qquad \textit{Eq 4.5}$$

We have dynamics **f** with only a few active terms in the space of possible righthand side functions; for example, the Lorenz equations only have a few linear and quadratic interaction terms per equation.

We then seek to approximate **f** by a generalized linear model

$$f(x) \approx \sum_{k=1}^{p} \theta_k(x)\xi_k = \Theta(x)\xi \qquad \textit{Eq 4.6}$$

with the fewest non-zero terms in ξ as possible. It is then possible to solve for the relevant terms that are active in the dynamics using sparse regression that penalizes the number of terms in the dynamics and scales well to large problems.

First, time-series data is collected from (Eq 4.5) and formed into a data matrix:

$$X = \begin{bmatrix} x(t_1) & x(t_2) & \dots & x(t_m) \end{bmatrix}^T \qquad \textit{Eq 4.7}$$

A similar matrix of derivatives is formed:

$$X = \begin{bmatrix} \dot{x}(t_1) & \dot{x}(t_2) & \dots & \dot{x}(t_m) \end{bmatrix}^T \qquad \textit{Eq 4.8}$$

In practice, this may be computed directly from the data in *X*; for noisy data, the total-variation regularized derivative tends to provide numerically robust derivatives. Alternatively, it is possible to formulate the SINDy algorithm for discrete-time systems $x_{k+1} = F(x_k)$,as in the DMD algorithm, and avoid derivatives entirely.

A library of candidate nonlinear functions $\Theta(X)$ may be constructed from the data in *X*:

$$\Theta(X) = [1 \quad X \quad X^2 \quad \dots \quad X^d \quad \dots \quad \sin(X) \quad \dots] \qquad \textit{Eq 4.9}$$

Here, the matrix X_d denotes a matrix with column vectors given by all possible time series of *d*th-degree polynomials in the state *x*. In general, this library of candidate functions is only limited by one's imagination.

The dynamical system in (Eq 4.5) may now be represented in terms of the data matrices in (Eq 4.8) and (Eq 4.9) as

$$\dot{X} = \Theta(X)\Xi \qquad \textit{Eq 4.10}$$

Each column ξ_k in Ξ is a vector of coefficients determining the active terms in the *k*th row in (Eq 4.5). A parsimonious model will provide an accurate model fit in (Eq 4.10) with as few terms as possible in Ξ. Such a model may be identified using a convex ℓ_1-regularized sparse regression:

$$\xi_k = \arg\min_{\xi_k^1} \left\| \dot{X}_k - \Theta(X)\xi_k^1 \right\|_2 + \lambda \left\| \xi_k^1 \right\|_1 . \qquad \textit{Eq 4.11}$$

Here, $\dot{X}_k$ is the *k*th column of $\dot{X}$, and λ is a sparsity-promoting knob. Sparse regression, such as the LASSO or the sequential thresholded least-squares (STLS) algorithm used in SINDy , improves the numerical robustness of this identification for noisy over-determined problems, in contrast to earlier methods that used compressed sensing. We advocate the STLS (Code 4.3) to select active terms

```
function Xi = sparsifyDynamics(Theta,dXdt,lambda,n)
% compute Sparse regression: sequential least squares
Xi = Theta\dXdt;  % initial guess: Least-squares

% lambda is our sparsification knob.
for k=1:10
    smallinds = (abs(Xi)<lambda);   % find small coefficients
    Xi(smallinds)=0;                % and threshold
    for ind = 1:n                   % n is state dimension
        biginds = ~smallinds(:,ind);
        % Regress dynamics onto remaining terms to find sparse Xi
        Xi(biginds,ind) = Theta(:,biginds)\dXdt(:,ind);
    end
end
```

Code 4.3 [MATLAB] Sequentially thresholded least-squares.

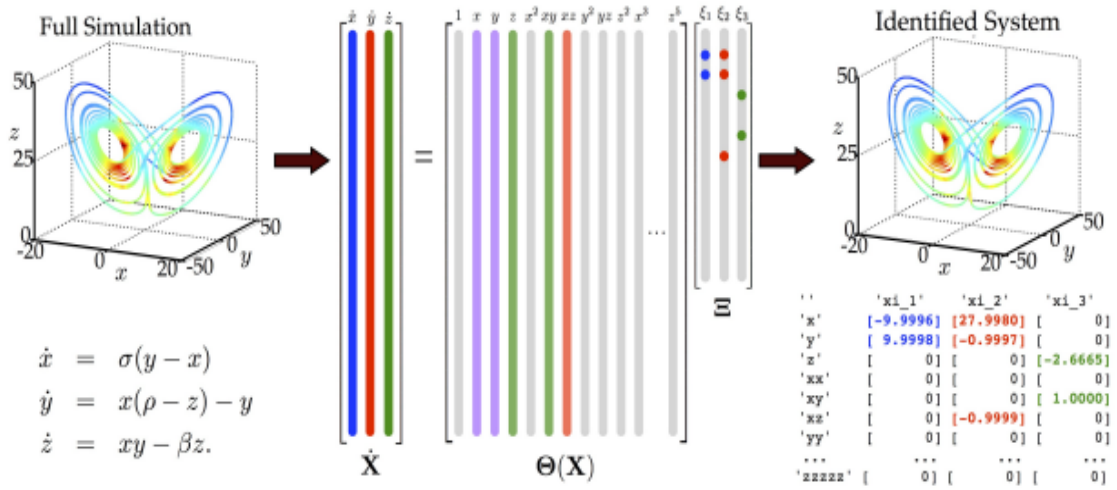

Figure 4.3 Schematic of the sparse identification of nonlinear dynamics (SINDy) algorithm. Parsimonious models are selected from a library of candidate nonlinear terms using sparse regression. This library $\Theta(X)$ *may be constructed purely from measurement data. Modified from Brunton et al.*

The sparse vectors ξ_k may be synthesized into a dynamical system:

$$\dot{x}_k = \Theta(x)\xi_k. \qquad \text{Eq 4.12}$$

Note that $\dot{x}_k$ is the *k*th element of $\dot{x}$ and $\Theta(x)$ is a row vector of symbolic functions of *x*, as opposed to the data matrix $\Theta(X)$. Figure 4.3 shows how SINDy may be used to discover the Lorenz equations from data. Code 4.4 performs the SINDy regression for the Lorenz system based on the data generated in Code 4.2.

```
%% Compute Derivative
for i=1:length(x)
    dx(i,:) = lorenz(0,x(i,:),Beta);
end
```

```
%% Build library and compute sparse regression
Theta = poolData(x,n,3);  % up to third order polynomials
lambda = 0.025;      % lambda is our sparsification knob.
Xi = sparsifyDynamics(Theta,dx,lambda,n)
```

Code 4.4 SINDy regression to identify the Lorenz system from data.

The output of the SINDy algorithm is a sparse matrix of coefficients Ξ:

	'xdot'	'ydot'	'zdot'
'1'	[0]	[0]	[0]
'x'	[-10.0000]	[28.0000]	[0]
'y'	[10.0000]	[-1.0000]	[0]
'z'	[0]	[0]	[-2.6667]
'xx'	[0]	[0]	[0]
'xy'	[0]	[0]	[1.0000]
'xz'	[0]	[-1.0000]	[0]
'yy'	[0]	[0]	[0]
'yz'	[0]	[0]	[0]
'zz'	[0]	[0]	[0]
'xxx'	[0]	[0]	[0]
'xxy'	[0]	[0]	[0]
'xxz'	[0]	[0]	[0]
'xyy'	[0]	[0]	[0]
'xyz'	[0]	[0]	[0]
'xzz'	[0]	[0]	[0]

Figure 4.4 The output of the SINDy algorithm

The result of the SINDy regression is a parsimonious model that includes only the most important terms required to explain the observed behavior. The sparse regression procedure used to identify the most parsimonious nonlinear model is a convex procedure. The alternative approach, which involves regression onto every possible sparse nonlinear structure, constitutes an intractable brute-force search through the combinatorically many candidate model forms. SINDy bypasses this combinatorial search with modern convex optimization and machine learning. It is interesting to note that, for discrete-time dynamics, if $\Theta(X)$ consists only of linear terms, and if we remove the sparsity-promoting term by setting $\lambda = 0$, then this algorithm reduces to the dynamic mode decomposition. If a least-squares regression is used, as in DMD, then even a small amount of measurement error or numerical round-off will lead to every term in the library being active in the dynamics, which is nonphysical. A major benefit of the SINDy architecture is the ability to identify parsimonious models that contain only the required nonlinear terms, resulting in interpretable models that avoid overfitting.

4.2. SIMPLE PENDULUM

4.2.1. OVERVIEW

A **pendulum** is a weight suspended from a pivot so that it can swing freely. When a pendulum is displaced sideways from its resting, equilibrium position, it is subject to a restoring force due to gravity that will accelerate it back toward the equilibrium position. When released, the restoring force acting on the pendulum's mass causes it to oscillate about the equilibrium position, swinging

back and forth. The time for one complete cycle, a left swing and a right swing, is called the period. The period depends on the length of the pendulum and also to a slight degree on the amplitude, the width of the pendulum's swing.

The simple gravity pendulum is an idealized mathematical model of a pendulum. This is a weight (or bob) on the end of a massless cord suspended from a pivot, without friction. When given an initial push, it will swing back and forth at a constant amplitude. Real pendulums are subject to friction and air drag, so the amplitude of their swings declines.

4.2.2. OSCILLATION OF A SIMPLE PENDULUM

A simple pendulum consists of a ball (point-mass) m hanging from a (massless) string of length L and fixed at a pivot point P. When displaced to an initial angle and released, the pendulum will swing back and forth with periodic motion.

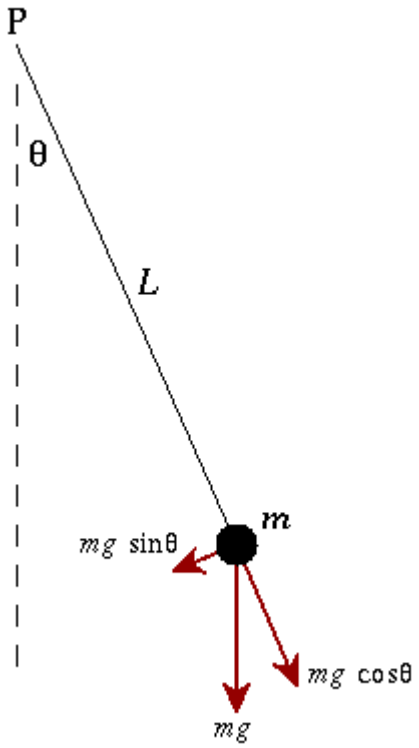

Figure 4.5 The model of simple pendulum

By applying Newton's second law for rotational systems, the equation of motion for the pendulum may be obtained

$$\tau = I\alpha \;\Rightarrow\; -mg\sin\theta L = mL^2\frac{d^2\theta}{dt^2} \qquad \text{Eq 4.13}$$

and rearranged as

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0 \qquad \text{Eq 4.14}$$

If the amplitude of angular displacement is small enough, so the small angle approximation holds true, then the equation of motion reduces to the equation of simple harmonic motion

$$\frac{d^2\theta}{dt^2} + \frac{g}{L} = 0 \qquad \textit{Eq 4.15}$$

The simple harmonic solution is

$$\theta(t) = \theta_0 \cos(\omega t) \qquad \textit{Eq 4.16}$$

where θ_0 is the initial angular displacement, and $\omega = \sqrt{g/L}$ the natural frequency of the motion. The period of this system (time for one oscillation) is

$$T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{L}{g}} \qquad \textit{Eq 4.17}$$

4.2.3. SIMPLE PENDULUM SIMULATION

It is simple to simulate dynamical systems, such as the simple pendulum. First, we have to verify the vector field of simple pendulum by following Code 4.5

```
function yp = nonlinear(t,y)
g=9.8;
l=1;
yp = [y(2);((-g/l) *sin(y(1)))];
```

Code 4.5 Define simple pendulum vector

In Code 4.6, we define the system parameters, initial condition, and timespan, and simulate the equations; in MATLAB we use the **ode45** command.

```
clear;
g=9.8;
l=1;
tspan = 0:0.01:15;
y0 = [1;0];
[t,y] = ode45('nonlinear',tspan,y0);
plot(t,y(:,1))
grid on;
xlabel('Time')
ylabel('Theta')
title('Theta Vs Time')
hold on;
```

Code 4.5 Define simple pendulum system parameters and simulate

And that is all we have done, we have defined a simple pendulum vector field, we put in our parameters, initial condition in the time span (*tspan*), and we have given

some *ode* options. After that, running the Code 4.6 then we got the simple pendulum simulation in Fig 4.6

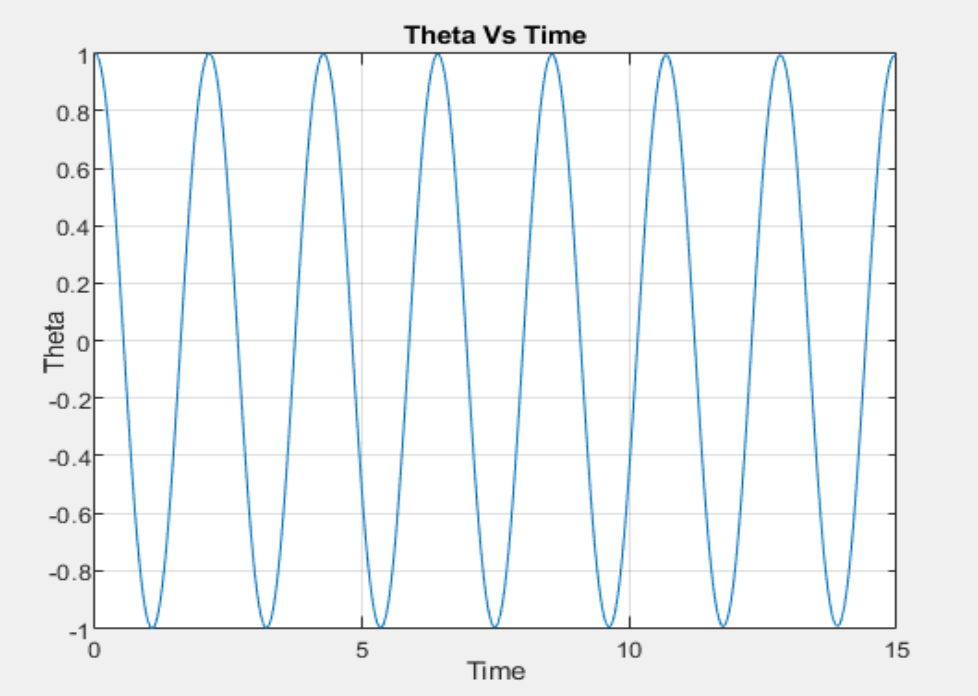

Figure 4.6 The period of oscillation of simple pendulum

4.2.4. THE LASSO FOR SIMPLE PENDULUM

After the simple pendulum system has been verified, we will try the Least-Square method then we use the LASSO Regression to see the different between those two methods. Firstly, we have to declare the input of *A* matrix and vector *y*, we can put any equations that we can think of. In my example, I will place 12 equations and vector *y* and the λ=0 in the Code below.

```
hold on;
theta = y(:,1);
dtheta = y(:,2);
ddtheta = -g/l*sin(theta);
cons = ones(length(tspan),1);
y = ddtheta;
A = [cons theta theta.^2 theta.^3 theta.*dtheta sin(theta) cos(theta) theta.*sin(theta) dtheta dtheta.^2 theta.^4 theta.^6];
lambda = 0;
x = lasso(A,y,'Lambda',lambda)
```

Code 46 The Least-Square method used in simple pendulum

And we will run the Code 4.7 to get the result of *x* vector in Fig 4.8

```
x =

         0
   -8.4976
    0.0026
    1.3318
    0.0000
   -1.2795
    0.0052
    0.0001
    0.0000
   -0.0000
   -0.0014
    0.0008
```

Figure 4.7 The output of Least-Square Regression

The solution will be very large with 9 objects which is not equal to zero. We can see that the problem of Least-Square method is that the *x* vector is extremely dense and we did not get many zero values. The Least-Square Regression is going to tell you that all the factors are important and all of them have non zero entries in *x* so *x* is not sparse anymore. But we know a lot of factors probably not collate the outcome.

If you get a model that is essentially dense, so all the entries *x* are non-zero, your model is still very complicated. In term of needing all of the explanatory factors to describe the outcome, so that is where the LASSO comes in. The ideal here is we generally makes as many entries of vector zero as possible. And so the LASSO regression still has error term, we are trying to minimize the error of the fit. We still want a good model but we want *x* to be as sparse as possible and have as many zero entries as possible. We used the LASSO into our simple pendulum in Code 4.7.

```
hold on;
theta = y(:,1);
dtheta = y(:,2);
ddtheta = -g/l*sin(theta);
cons = ones(length(tspan),1);
y = ddtheta;
A = [cons theta theta.^2 theta.^3 theta.*dtheta sin(theta) cos(theta) theta.*sin(theta) dtheta dtheta.^2 theta.^4 theta.^6];
lambda = 0.2;
x = lasso(A,y,'Lambda',lambda)
```

Code 48 The LASSO used in simple pendulum

After we used the LASSO function in Matlab, we get the result of *x* vector in the figure down below.

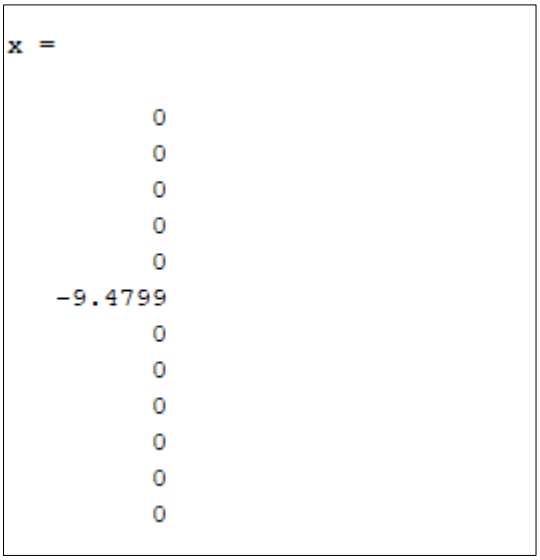

Figure 4.8 The output of LASSO function

We can see that the result of *x* in position 6 where *x* = sin(theta) in A. With the final result, we got only one acceptance which is nearly equal to initial constant $\frac{g}{L} \approx 9.81$. So we can consider the principle of simple pendulum is $\ddot{\theta} = -\frac{g}{L}\sin\dot{\theta}$

That is what we are trying to find by the data which we collected before.

CONCLUSION

The topic “Data Science in the Identification of Mechanical Systems” has completed as schedule and achieved following results:

- Have applied basic knowledge about data science knowledge to complete the tasks of this thesis.
- Using the Sparse identification of nonlinear dynamic onto the chaotic Lorenz System.
- Write and apply MATLAB software to simulate and make sparse regression for the Lorenz system.
- Design the simple pendulum model and observe the data from it.
- Design the LASSO program, simulate and make a regression successfully.
- The result of the thesis could be applied to practice, apply in basic tasks.

Through the above topic, we have known how to apply studied major knowledge which was taught in Hanoi University of Science and Technology for the last five years into real life, especially in industrial environment.

Due to the limit of time and also the knowledge in this project, we have just solved some of the basic problems in Data Science. There are much more problems need to be tackled to have improvements in near future. Therefore, I would like lecturers and teachers in dissertation committee give me opinions to improve this thesis more and more.

Best wishes!

REFERENCES

[1] J. L. P. a. J. N. K. Steven L. Brunton, Discovering governing equations from data by sparse idenfication of nonlinear dymical system, 2016.

[2] Steven L. Brunton, Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.

[3] R. Tibshirani, Regression Shrinkage and Selection via the Lasso.