ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

MACHINE LEARNING ANALYSIS OF PULSAR TIMING DATA

M.Sc. THESIS

Esma HASANÇEBİ ESER

Department of Physics Engineering

Physics Engineering Programme

DECEMBER 2021

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

MACHINE LEARNING ANALYSIS OF PULSAR TIMING DATA

M.Sc. THESIS

Esma HASANÇEBİ ESER
(509191108)

Department of Physics Engineering

Physics Engineering Programme

Thesis Advisor: Prof. Dr. Muammer Altan ÇAKIR

DECEMBER 2021

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ATARCA ZAMAN VERİSİNİN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE ANALİZİ

YÜKSEK LİSANS TEZİ

Esma HASANÇEBİ ESER
(509191108)

Fizik Mühendisliği Anabilim Dalı

Fizik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Muammer Altan ÇAKIR

ARALIK 2021

Esma HASANÇEBİ ESER, a M.Sc. student of İTU Graduate School student ID 509191108 successfully defended the thesis/dissertation entitled "MACHINE LEARNING ANALYSIS OF PULSAR TIMING DATA", which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Muammer Altan ÇAKIR** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Kazım Yavuz EKŞİ** Istanbul Technical University	.....
	Prof. Dr. Tolga GÜVER Istanbul University	.....

Date of Submission : 13 November 2021
Date of Defense : 17 December 2021

To the pale blue dot...

FOREWORD

First of all, I would like to thank my supervisor Prof. Dr. Altan Çakır for giving me the opportunity to work with him. I am sincerely grateful to Prof. Dr. Kazım Yavuz Ekşi for his invaluable guidance and continuous encouragement throughout our study. His grace and scientific perspective inspire and enrich my progress as a physicist. Also, I thank him very much for his advice on the subject of the thesis. I am also indebted very much to Dr. Sinem Şaşmaz for her comments and suggestions. She was always there for me. Also, I would like to thank research assistant Aytül Filiz for her suggestions and kindness.

Most importantly, I would like to thank my precious family for believing in me and encouraging me to continue on the path of science. I am grateful to my mother Yasemin Hasançebi and my father Erhan Hasançebi for their unconditional love. I would like to thank my lovely grandmother Sezgin Gürbüz for her good wishes for me in every exam and project. Also, I would like to thank my dear grandfather Selami Hasançebi for teaching me courage and entrepreneurship.

Finally, I would like to thank my dear husband Baran Eser, for his unending patience and love. I have kept the promise I made to him regarding this thesis when we were in Mardin. I believe that with this thesis, I have created the first legacy of our family.

DECEMBER 2021

Esma HASANÇEBİ ESER
(Physics Engineer)

TABLE OF CONTENTS

Page

FOREWORD ..... **ix**
TABLE OF CONTENTS ..... **xi**
ABBREVIATIONS ..... **xiii**
SYMBOLS ..... **xv**
LIST OF TABLES ..... **xvii**
LIST OF FIGURES ..... **xix**
SUMMARY ..... **xxi**
ÖZET ..... **xxiii**
1. INTRODUCTION **1**
1.1 Problem Definition 4
1.1.1 Crab-like pulsars 4
1.2 Literature 6
1.3 Hypothesis 6
2. THEORETICAL BACKGROUND **7**
2.1 Dynamical Systems 7
2.1.1 Mathematical motivation of dynamical systems 7
2.1.1.1 Discrete time dynamical systems 8
2.1.1.2 Spectral decomposition 9
2.2 Sparse Identification of Nonlinear Dynamics from Data 10
2.2.1 PySINDy : A Python package for the sparse identification of nonlinear dynamics from data 12
2.3 Hankel Matrix 14
2.4 Singular Value Decomposition 14
2.5 Long Short-Term Memory 15
3. RESULTS **19**
3.1 Vela and Crab Pulsar Data Sets 19
3.2 Detrending Data Sets 20
3.3 Application of Singular Value Decomposition 23
3.4 Application of Sparse Identification of Nonlinear Dynamics from Data 25
3.4.1 Model score 27
3.5 Application of Long Short-Term Memory 28
4. CONCLUSION **33**
REFERENCES **35**
APPENDICES **39**
APPENDIX A 41
APPENDIX B 43

CURRICULUM VITAE **47**

ABBREVIATIONS

DMD	**:** Dynamical Mode Decomposition
SVD	**:** Singular Value Decomposition
SINDy	**:** Sparse Identification of Nonlinear Dynamical Systems
pySINDy	**:** Python Package of Sparse Identification of Nonlinear Dynamical Systems
RMS	**:** root-mean-squares
LSTM	**:** Long-Short Term Memory
PL	**:** Power-Law
PTA	**:** Pulsar Timing Array
RNN	**:** Recurrent Neural Network
NN	**:** Neural Network
SNR	**:** Supernova Remnant
NS	**:** Neutron Star
MDR	**:** Magnetic Dipole Radiation
SN	**:** Supernova
PWN	**:** Pulsar Wind Nebula

SYMBOLS

$\mathbf{M_{\odot}}$	**:** Solar mass
P	**:** Pulsar spin period
$\boldsymbol{\tau}$	**:** The age of a pulsar
m	**:** Magnetic moment
I	**:** Moment of inertia
n	**:** Pulsar braking index
$\mathbf{B_p}$	**:** Polar magnetic field
$\mathbf{E_{rot}}$	**:** Rotational energy
$\mathbf{\dot{E}_{rot}}$	**:** Energy loss rate
$\boldsymbol{\Omega}$	**:** Angular frequency
$\boldsymbol{\alpha}$	**:** The angle between the rotation axis and the magnetic field axis

LIST OF TABLES

Page

Table 1.1 **:** Braking indices of some young pulsars. 5
Table 3.1 **:** Model scores of SINDy. 27
Table 3.2 **:** Error analysis of LSTM method. 29

LIST OF FIGURES

Page

Figure 1.1 : Sky survey of Arecibo telescope for pulsars **2**
Figure 1.2 : $P - \dot{P}$ diagram of pulsars **3**
Figure 1.3 : X-ray picture of Crab Nebula, taken by Chandra X-ray Observatory . **4**
Figure 2.1 : Chaotic trajectory of the Lorenz system **8**
Figure 2.2 : Logistic map **9**
Figure 2.3 : Schematic of the SINDy algorithm **12**
Figure 2.4 : The submodules of PySINDy **13**
Figure 2.5 : Hankel matrix **14**
Figure 2.6 : Architecture of Recurrent Neural Network **17**
Figure 2.7 : Long Short-Term Memory architecture **17**
Figure 3.1 : Line trend of Crab pulsar data set **20**
Figure 3.2 : Parabolic trend of Crab pulsar data set **21**
Figure 3.3 : Residuals of Crab pulsar data set **22**
Figure 3.4 : Line trend of Vela pulsar data set **22**
Figure 3.5 : Residuals of Vela pulsar data set **23**
Figure 3.7 : Vela pulsar - variance of the modes of singular value decomposition **24**
Figure 3.6 : Crab pulsar - variance of the modes of singular value decomposition **24**
Figure 3.8 : Time delay coordinates embedding for Crab pulsar data set **25**
Figure 3.9 : Time delay coordinates embedding for Vela pulsar data set **25**
Figure 3.10 : Crab pulsar data set SINDy application **26**
Figure 3.11 : Vela pulsar data set SINDy application **27**
Figure 3.12 : Model loss of Crab pulsar data set **29**
Figure 3.13 : Model loss of Vela pulsar data set : **30**
Figure 3.14 : Crab pulsar data set : Comparison on predicted training and test data vs. observation data **30**
Figure 3.15 : Vela pulsar data set : Comparison on predicted training and test data vs. observation data **31**

MACHINE LEARNING ANALYSIS OF PULSAR TIMING DATA

SUMMARY

In 1967, radio pulsations from a celestial body were discovered by a graduate student Jocelyn Bell and her advisor Antony Hewish. This was the first sample of about three thousand similar sources, called pulsars, to be discovered in our galaxy to date. It has been understood that pulsars are rapidly spinning, strongly magnetized neutron stars. Neutron stars are very dense objects formed by the collapse of the cores of massive stars at the end of their life.

Each pulsar has its characteristic pulse shape and rotational frequency. The rotational frequency of pulsars can be measured very precisely. The rotation frequencies of pulsars are observed to decrease in time. Pulsars tap their radiative energy from their rotational kinetic energy. The mechanisms by which pulsars achieve this energy conversion is not well-understood. According to a prominent model, the pulsars convert their kinetic energy into radiation by emission of magnetic dipole radiation (MDR). However, studies with young pulsar data show that the MDR model does not fully explain the observations and there should be other mechanisms assisting the spin-down. The ejection of high-energy particles, the growth of the dipole magnetic field over time, interaction with a supernova debris disc, increasing inclination angle between the rotation and magnetic axis, and gravitational wave emission are some of the processes proposed to affect spin-evolution.

Occasionally, some pulsars suffer sudden increases in their spin, also known as "glitches" which decay in the following weeks or months. When they were first discovered, it was thought that glitches result from the breaking of the crust and hence they were called "stellar quakes". Today, it is conceived that this model can only account for the smallest glitches or that it could be a triggering mechanism for the main cause of the glitches. According to the more favoured view, the glitches are caused by the dynamics of the crustal superfluid. Sometimes a new glitch occurs before the previous glitch decayed. The presence of glitches in the pulsar data complicates the understanding of the spin evolution.

The aim of the thesis is to contribute to the understanding of the spin evolution of pulsars by machine learning methods. To this end, long-term time-dependent spin frequency data of Crab and Vela pulsars are used. These are the two best-known pulsars that have been studied the most.

Since the frequency changes by a small fraction throughout the time-span of the observations, we have eliminated the basic trend by fitting the data with a polynomial

function. By subtracting this basic trend from the data, we obtained the residuals that clearly show the complicated features of spin evolution such as glitches. We tested the performance of two machine learning methods in reproducing the evolution of the residuals.

The first method is called the *sparse identification of nonlinear dynamics* (SINDY). Given a time-series data, SINDY can identify the governing system of ordinary differential equations. We thus used this method to find the governing equations for the evolution of the residuals. The SINDy method gave information about the order of the equations and their coefficients.

In addition, we used a *recurrent neural network* (RNN) architecture called *long short-term memory* (LSTM) method on the same data sets. We found that LSTM can predict the dates of glitches in the test data.

The results show that SINDy and LSTM applications can contribute to the studies on the spin evolution of pulsars and may take place more in studies related to pulsars in the future.

ATARCA ZAMAN VERİSİNİN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE ANALİZİ

ÖZET

1967'de doktora öğrencisi Jocelyn Bell ve danışmanı Antony Hewish tarafından radyo bandında atımlı ışıma yapmakta olan bir gökcismi keşfedildi. Bu, günümüze dek galaksimizde keşfedilen ve *pulsar* (atarca) olarak adlandırılan üç bin civarında benzer kaynağın ilk örneğiydi. Pulsarların hızlı dönen ve yüksek manyetik alana sahip nötron yıldızları olduğu anlaşılmıştır. Nötron yıldızları büyük kütleli yıldızların yaşamının sonunda çekirdeklerinin çökmesi ile oluşan çok yoğun nesnelerdir.

Her pulsarın karakteristik bir atım şekli ve dönüş frekansı vardır. Pulsarların dönme frekansı çok hassas biçimde ölçülebilmektedir. Bu dönme frekanslarının zaman içinde küçüldüğü gözlenmektedir. Pulsarlar ışıdıkları enerjiyi dönme kinetik enerjilerinden devşirirler. Pulsarların bu enerji dönüşümünü hangi süreçlerle yaptıkları tam olarak anlaşılamamıştır. Öne çıkan bir modele göre atarcalar dönme kinetik enerjilerini manyetik dipol ışıması yolu ile ışımaya dönüştürürler. Bununla birlikte, genç atarcalar ile yapılan çalışmalar, MDR modelinin yetersiz kaldığını, yavaşlamaya yardımcı olan başka süreçlerin olması gerektiğine göstermektedir. Örneğin, yüksek enerjili parçacıkların atımı, dipol manyetik alanın zamanla büyümesi, pulsar manyetosferinin bir süpernova kalıntısı disk ile etkileşmesi, dipol ve dönme eksenleri arasındaki eğim açısının artması, ve kütleçekimsel dalga emisyonu gibi süreçler pulsarların yavaşlamasını etkileyebilecek mekanizmalar olarak önerilmiştir.

Kimi pulsarların dönüş hızları ara sıra "sıçrama" adı verilen ani artışlar göstermekte sonrasında haftalar veya aylar içerisinde sönümlenmektedir. İlk keşfedildiklerinde bunların nötron yıldızının kabuğunun kırılması ile gerçekleşen "yıldız depremleri" oldukları düşünülmüştür. Bugün bu modelin ancak en küçük genlikli sıçramaları açıklayabileceği veya sadece asıl sıçrama nedenini tetikleyebilecek bir unsur olabileceği düşünülmektedir. Daha çok kabul gören bir görüşe göre sıçramaların asıl nedeni kabuk bölgesindeki süperakışkanın dinamiğidir. Kimi zaman bir sıçramanın sönümlenmesi tamamlanmadan yenisi gerçekleşmektedir. Sıçramaların varlığı pulsarların uzun dönemli spin evrimlerinin anlaşılmasını güçleştiren unsurlardan biridir.

Tezin amacı makina öğrenmesi yöntemleriyle pulsarın spin evriminin anlaşılmasına katkıda bulunmaktır. Tez çalışması kapsamında, Crab ve Vela pulsarlarının uzun döneme yayılmış dönme frekans verileri kullanıldı. Bunlar, üzerinde en çok çalışılmış, en iyi bilinen iki pulsardır.

Frekans, gözlemlerin zaman aralığı boyunca küçük bir kesir kadar değiştiğinden, verileri bir polinom fonksiyonuyla uydurarak temel eğilimi ortadan kaldırdık. Bu

temel eğilimi verilerden çıkararak, dönme evriminin, sıçramalar gibi, karmaşık özelliklerini açıkça gösteren artıkları elde ettik. Artıkların evrimini yeniden üretmede iki makine öğrenimi yönteminin performansını test ettik.

İlk yöntem *doğrusal olmayan dinamik sistemlerin seyrek tanımlanması* (SINDy) olarak adlandırılır. Bir zaman serisi verisi verildiğinde, SINDY adi diferansiyel denklemlerin yönetim sistemini tanımlayabilir. Böylece artıkların evrimi için temel denklemleri bulmak için bu yöntemi kullandık. SINDy yöntemi, denklemlerin mertebesi ve katsayıları hakkında bilgi verdi.

Ayrıca, aynı veriye *uzun kısa-süreli bellek* (LSTM) yöntemi adı verilen bir *yineleyen sinir ağı* (RNN) mimarisini uyguladık. LSTM'in test verilerindeki sıçramaların gününü tahmin edebildiğini bulduk.

Sonuçlar, SINDy ve LSTM uygulamalarının pulsarların spin evrimi ile ilgili çalışmalara katkı sağlayabileceğini ve gelecekte pulsarlar ilgili çalışmalarda daha fazla yer alabileceğini göstermektedir.

1. INTRODUCTION

Like human beings, stars are born, shine, and die one day. The mass of a star determines how long it will live and how it will die. Thermonuclear reactions in the core of a star will terminate when the iron is formed. The core of a star of mass 8 to 20 solar masses, at the end of its life, collapses and the energy released due to the collapse leads to a strong explosion where the outer layers of the star are thrown into the interstellar medium. The compact object that is formed by the collapse is a *neutron star* and the explosion is observed as a *supernova*.

The mass of a neutron star is about 1.5 - 2 solar masses and its radius is 10-12 km. As such neutron stars are very dense objects with densities reaching $10^{15}\,\mathrm{g/cm^3}$. The most outstanding feature of neutron stars is that they have the highest known magnetic fields in the universe.

During the collapse, the conservation of angular momentum leads to rotational periods as short as milliseconds. For a rotating sphere, the rotational kinetic energy is

$$E_{\mathrm{rot}} = \frac{1}{2} I \Omega^2 \, . \tag{1.1}$$

where I is the moment of inertia and Ω is the angular frequency of rotation. The moment of inertia depends on the equation of state and, approximately, is given by

$$I \approx 0.33 M R^2 = 1.4 \times 10^{45} \left(\frac{M}{1.5 M_{\odot}} \right) \left(\frac{R}{12\,\mathrm{km}} \right)^2 \mathrm{g\ cm^2} \, . \tag{1.2}$$

By using the numerical value of the moment of inertia the rotational kinetic energy can be written as:

$$E_{\mathrm{rot}} \approx 2.8 \times 10^{48} \left(\frac{P}{100\,\mathrm{ms}} \right)^{-2} \left(\frac{M}{1.5 M_{\odot}} \right) \left(\frac{R}{12\,\mathrm{km}} \right)^2 \mathrm{erg} \tag{1.3}$$

where $P = 2\pi/\Omega$ is the spin period.

Rotationally powered pulsars (RPPs) are a subclass of neutron stars that are powered by tapping energy from their rotational kinetic energy. The spin-down power of such an object ($L_{\mathrm{sd}} = -\dot{E}_{\mathrm{rot}}$)

$$L_{\mathrm{sd}} = -I \Omega \dot{\Omega} = 4\pi^2 I \frac{\dot{P}}{P^3} . \tag{1.4}$$

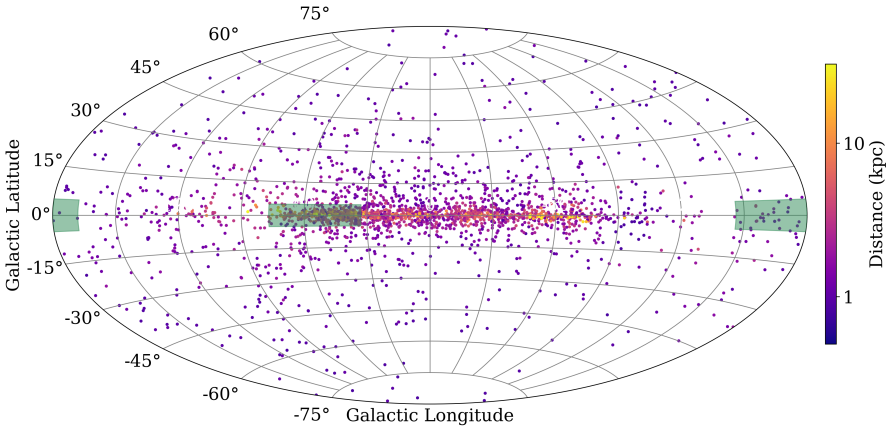

Figure 1.1 : Sky map in Galactic coordinates showing the portions of the Galactic plane being targeted by the PALFA survey. This region corresponds to most of the Galactic plane visible with the Arecibo telescope. The dots correspond to known pulsars, their color indicates their inferred distance. Figure courtesy: Emilie Parent[1].

would give the maximum radiative output of such a source. RPPs were first discovered as sources of pulsed radio emission, but some are discovered in other wavelengths. Rotational energy is the fundamental energy source for particle flow and radiation of RPPs. They accelerate particles by their strong magnetic fields. Using the numerical values, the spin-down power can be written as

$$L_{\rm sd} \approx 5.5 \times 10^{34} \left(\frac{P}{100\,{\rm ms}} \right)^{-3} \left(\frac{\dot{P}}{10^{-15}} \right) {\rm erg\,s^{-1}} \, . \tag{1.5}$$

If the rotation and magnetic axes of a neutron star are not aligned, the radiation emitted from the magnetic poles, if in the line of sight, can be observed periodically as the pulsar rotates. This *lighthouse effect* is the cause of the pulsed emission of RPPs.

The *magnetic dipole radiation* is accepted as the main process by which RPPs convert their rotational energy to radiation. The energy loss rate for a rotating dipole is given by

$$\dot{E} = -\frac{2m^2\Omega^4 \sin^2\alpha}{3c^3} \tag{1.6}$$

where m is the magnetic dipole moment and α is the angle between the rotation and the magnetic dipole axis. The magnetic dipole moment is related to the polar magnetic field by $m = B_{\rm p}R^3/2$. The motion of the charged particles along the magnetic fields generates the observed radio emission.

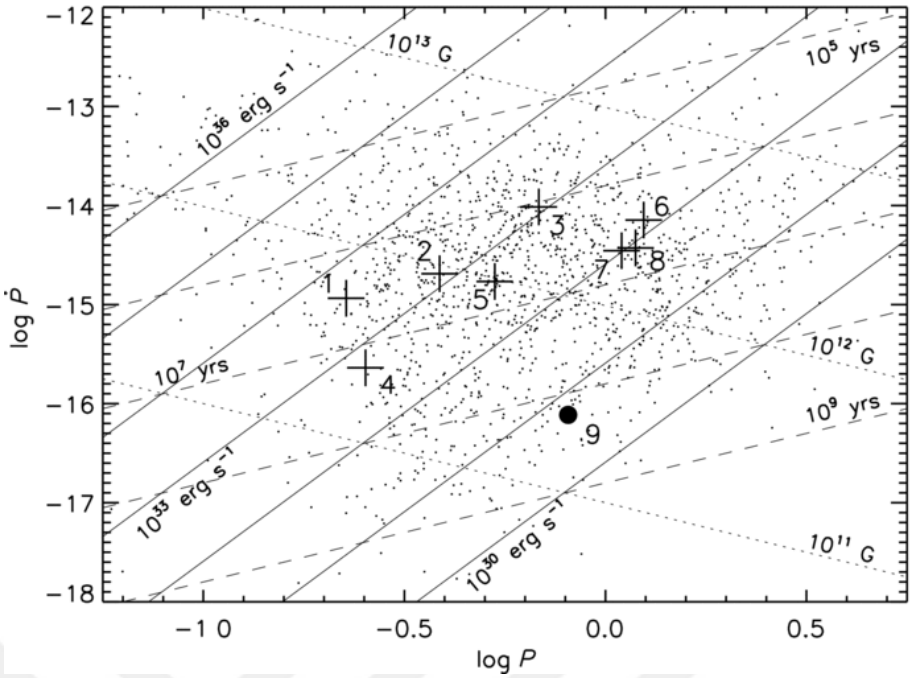

Figure 1.2 : $P-\dot{P}$ diagram of pulsars from the ATNF Pulsar Catalogue [1]

We use the equations (**1.4**) and (**1.6**) to obtain the polar magnetic field in terms of the P and $\dot{P}$:

$$B_{\rm p}^2 = \frac{3c^3}{2\pi^2}\frac{1}{\sin^2\alpha}\frac{I}{R^6}P\dot{P}\,. \tag{1.7}$$

For $I = 1.4\times10^{45}\,\mathrm{gcm}^2$, $R = 12\,\mathrm{km}$ and $\sin^2\alpha = 1/2$, the dipole magnetic field of a pulsar becomes,

$$B_{\rm p} \approx 6.2\times10^{19}\sqrt{P\dot{P}}\,\mathrm{G}\,. \tag{1.8}$$

The *characteristic age* of a pulsar is defined as

$$\tau \equiv \frac{P}{2\dot{P}}\,. \tag{1.9}$$

This is a crude estimation for the true age of a pulsar. It assumes magnetic dipole radiation do dominate and that the initial period P_0 is much shorter that the present period.

For most pulsars, the period, P and its derivative, $\dot{P}$ are the only measurements available to predict the properties of the pulsar. Usually, the $P-\dot{P}$ diagram is used to visualize pulsar populations as shown in figure **1.2**.

The braking index of a pulsar is defined as

$$n \equiv \frac{\Omega\ddot{\Omega}}{\dot{\Omega}^2}\,. \tag{1.10}$$

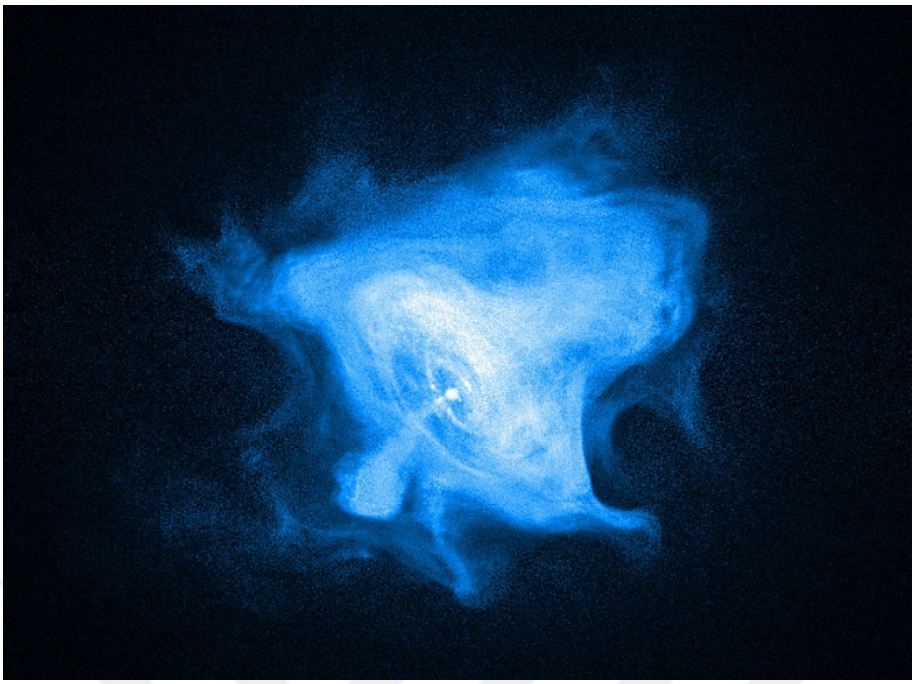

Figure 1.3 : X-ray picture of Crab Nebula, taken by Chandra X-ray Observatory. Figure courtesy [2].

This predicts $n = 3$ for an object spinning down by magnetic dipole radiation. The observed values of braking indices are usually smaller than 3 and several models are suggested to address this difference. The braking index is hard to measure because of the timing noise. As a result, it is measured only for a few pulsars. The characteristic age for a pulsar with a measured braking index can be defined as

$$\tau_{\rm ch} \equiv \frac{P}{(n-1)\dot{P}} \,. \tag{1.11}$$

1.1 Problem Definition

1.1.1 Crab-like pulsars

Pulsars continuously spin-down by releasing electromagnetic energy. The presence of a neutron star at the center of the Crab nebula was shown by [2] and its pulsations with a frequency of 30.2 Hz were discovered by [3]. The spin frequency of the object decreased by 0.5 Hz from 1968 to 1984 [4, 5].

Lyne et al. [6] measured the braking index of the Crab pulsar as 2.51 ± 0.01 between two glitches. It is less than the value expected in the MDR model. Some of the measured braking indices are shown in table **3.1**.

[2] https://chandra.harvard.edu/photo/2018/crab/

Table 1.1 : Braking indices of some young pulsars.

PSR	n	REFERENCE
B0531+21(Crab)	2.51	Lyne et al. (1993) [7]
B0540-69	2.14	Ferdman et al. (2015)[8]
B0833-45 (Vela)	1.4	Lyne et al. (1996) [9]
J1833-1034	1.857	Roy, Gupta and Lewandowski (2012) [10]

We can easily see from table **3.1** that the MDR model alone is not sufficient to explain the deceleration process. There are many factors that can affect the spin evolution of pulsars. This makes it difficult to model and understand the spin-evolution of young pulsars.

Some of the factors that have been suggested to affect the slowdown are as follows:

- Ejection of relativistic particles as indicated by the presence of pulsar wind nebula (PWN) around some young pulsars.
- Increasing inclination angle between rotation axes and the dipole as observed in the Crab pulsar [11] possibly due to current-loss in the magnetosphere [12].
- An increasing magnetic dipole field strength [13]
- Interaction with an supernova debris disc [14]
- Gravitational wave emission.

In addition to all these processes the glitch events may also affect the measured braking indices since it takes several weeks to months until the pulsar relaxes to its pre-glitch spin-down trend. The presence of timing-noise also complicates the understanding of the pulsar timing data set.

1.2 Literature

With the developing technology, the amount of data produced in the field of astronomy is increasing. Machine learning methods that can understand the features of a data set are very useful for interpreting such large data sets.

A recent theme that has emerged is the "data-driven science". This approach has culminated with the introduction of "sparse identification of nonlinear dynamics" (SINDy) [15]. `pySINDy` is a Python implementation of this algorithm to identify non-linear dynamical system models from time-series data [16]. It can infer the governing dynamical equations from measurement data. For the first time, we have used this method to infer the equations of pulsar dynamics from measured spin history of pulsars.

Another method employed in this thesis is the long short-term memory (LSTM) [17] method which is an artificial recurrent neural network architecture. The significant feature that distinguishes LSTM from other neural networks is that it has feedback connections. It can process entire sequences of data set. LSTM is convenient for classifying and making predictions based on time-series data.

1.3 Hypothesis

We propose in the thesis that SINDy and LSTM can be suitable methods to get insight into pulsar spin evolution.

A set of ordinary differential equations was obtained from the observational spin-frequency data of Vela and Crab pulsars by using `pySINDy`. Comparative graphs and residues of the observation data and solution data were plotted by performing the numerical solution of the obtained model solution. In addition, LSTM is applied to the same data sets. We find that LSTM method can be useful for predicting the time of upcoming glitch events.

2. THEORETICAL BACKGROUND

2.1 Dynamical Systems

The dynamic systems study the behaviour of differential equations that characterize the evolution of a system. This definition applies to many systems that depend on time, such as classical mechanics, economics, and weather forecasting.

For centuries, many scientists have researched to understand dynamical systems using differential equations, linear algebra, numerical analysis and topology. Recently, especially with the development of machine learning algorithms, data-driven analysis methods contribute to our understanding of dynamic systems.

2.1.1 Mathematical motivation of dynamical systems

In this section, I followed the chapters 1 and 7 of the book by Steven L. Brunton and J. Nathan Kutz [18]. Dynamical systems can be defined in the form as:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t; \mathbf{c}) \ , \tag{2.1}$$

where $\mathbf{x}$ is the state of the system and $\mathbf{f}$ is probably a vector field that depends on the state $\mathbf{x}$, time t, and a set of parameters $\mathbf{c}$.

For instance the Lorenz system which is shown in figure **2.1** is among the simplest and best studied dynamical systems and can be expressed by the following equations

$$\dot{x} = \sigma(y - x) \ , \tag{2.2}$$

$$\dot{y} = x(\rho - z) - y \ , \tag{2.3}$$

$$\dot{z} = xy - \beta z \ . \tag{2.4}$$

In this instance, the state vector $\mathbf{x}$ equals to $[x, y, z]^T$ and $\mathbf{c}$ parameter vector equals to $[\sigma, \rho, \beta]^T$.

The basic case of a dynamical system without explicit time-dependence or parameters can be defined as:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)) \ . \tag{2.5}$$

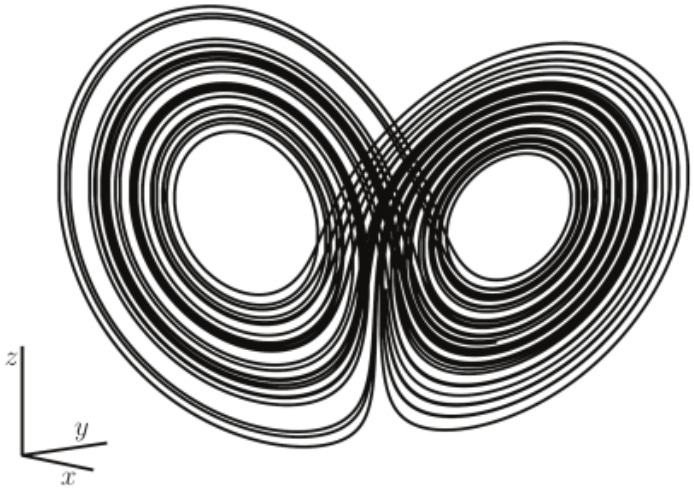

Figure 2.1 : Chaotic trajectory of the Lorenz system [18].

Principally, $\mathbf{x}(t)$ is an element of an n-dimensional state that lives on a manifold and $\mathbf{f}(\mathbf{x}(t))$ is element of tangent bundle of manifold. In simpler case $\mathbf{x}$ is a vector, $\mathbf{f}$ is a Lipschitz function [19] and manifold in R^n. These conditions ensure that the solution exists and is unique to (**2.5**).

2.1.1.1 Discrete time dynamical systems

There is no discontinuity in time in the definition of a continuous-time dynamic system. Discrete time systems, on the other hand, have a more general expression that also defines discontinuous times in the system. In a discrete system model, we examine the state of the system in a certain time interval, for example, in years or nanoseconds, or even in a plot that occurs at irregular intervals. Discrete-time dynamical system model is expressed by difference or recurrence equations

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, t) \tag{2.6}$$

where $\mathbf{x}_t$ is state variables of the dynamical system at time t and $\mathbf{F}$ is a function that defines the rules that characterize the behavior of the system.

For example, discrete time systems can be examined through logistic maps. It is a one-dimensional discrete-time map which is shown in figure **2.2** and defined by the following equation

$$x_{t+1} = \lambda x_t (1 - x_t) \ . \tag{2.7}$$

Points that remain unchanged in an iterated map are called fixed points. Usually these special points are tried to be found first. It is decided whether the points are attractors or

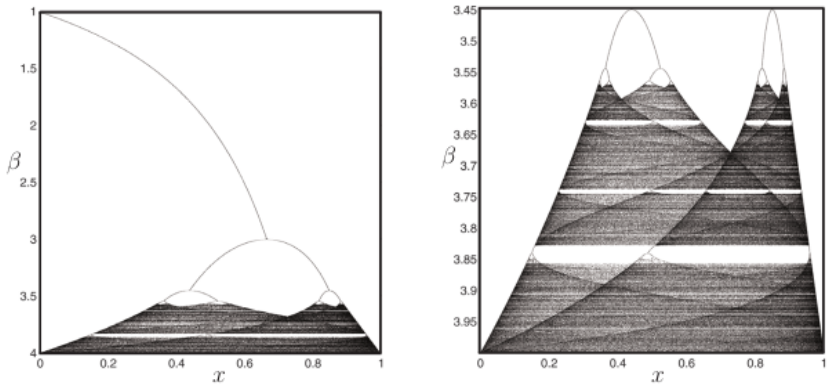

Figure 2.2 : Logistic map. [18]

repellers by examining the x values at the fixed points. If the parameter λ is increased, the attractors show chaotic behavior.

Discrete-time dynamical systems can be derived from continuous-time dynamical systems. x_t can be obtained from equation (**2.5**) in the form of Δt .

2.1.1.2 Spectral decomposition

A linear dynamic system can be described as,

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} \; . \tag{2.8}$$

where $\mathbf{A}$ is a square matrix. The solution of equation (**2.8**) is

$$\mathbf{x}(t_0 + t) = e^{\mathbf{A}t}\mathbf{x}(t_0) \; . \tag{2.9}$$

The dynamics of the system is characterized by the eigenvalues and eigenvectors of the matrix $\mathbf{A}$. Spectral decomposition is also called as eigen-decomposition. Spectral decomposition of $\mathbf{A}$ is shown below

$$\mathbf{A}\mathbf{T} = \mathbf{T}\mathbf{\Lambda} \; . \tag{2.10}$$

When matrix $\mathbf{A}$ has n different eigenvalues, $\mathbf{\Lambda}$ becomes a diagonal matrix containing eigenvalue Λ_j and $\mathbf{T}$ matrix's columns are the linearly independent eigenvectors related to these eigenvalues. $\mathbf{A} = \mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}$ relation can be established. The solution given in equation (**2.9**) becomes

$$\mathbf{x}(t_0 + t) = \mathbf{T}\mathrm{e}^{\mathbf{\Lambda}t}T^{-1}\mathbf{x}(t_0) \; . \tag{2.11}$$

Time dependent continuous systems can cause discontinuous time dynamics. The discrete time eigenvalues are $e^{\mathbf{\Lambda} t}$.

$$\mathbf{z} = \mathbf{T}^{-1}\mathbf{x} \tag{2.12}$$

where $\mathbf{T}^{-1}$ defines a transformation $\mathbf{z}$ is eigenvector coordinates.

$$\frac{d}{dt}\mathbf{z} = \mathbf{\Lambda}\mathbf{z} \tag{2.13}$$

where dynamics diverge. That is, each coordinate is only dependent on itself with simple dynamics

$$\frac{d}{dt}z_j = \Lambda_j z_j \ . \tag{2.14}$$

At the points where the dynamics are decoupled, the system coordinates can be converted to eigenvectors. Therefore, it is highly desirable to work with linear systems.

2.2 Sparse Identification of Nonlinear Dynamics from Data

Analyzing systems with classical analysis methods is insufficient due to the increase in the number of data in most experiments and observations. Exploring the dynamic system models and obtaining the governing equations from data is a challenge in many fields of science. Machine learning and regression methods have gained more importance today. In 2016 Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz have introduced a method called Sparse Identification of Nonlinear Dynamics [15]. In order to obtain the governing equations of the data set including the noise, machine learning and sparsity increasing methods were combined and applied to nonlinear systems. The basic assumption in this model is that there are only a few prominent terms governing the dynamics. To find these prominent terms, sparse regression has been added to the method. The advantage of sparse regression is that it prevents overfitting while preserving the accuracy of the model.

We have previously defined dynamical systems in the form as equation (**2.5**). The vector $\mathbf{x}(t)$ in this equation represents the states of the dynamical system at a time t. The $\mathbf{f}(\mathbf{x}(t))$ function contains the dynamic constraints that form the system's equations of motion. The important feature in SINDy is that the function $\mathbf{f}$ in most cases consists of a small number of terms i.e. it is sparse in the space of possible functions.

In order to approximate the function **f**, it is sought that the generalized linear model equation has the least number of non-zero terms in ξ

$$\mathbf{f}(\mathbf{x}) \approx \sum_{k=1}^{p} \theta_k(x)\xi_k = \Theta(x)\xi \ . \tag{2.15}$$

Firstly, in order to obtain **f** the time series data of $\mathbf{x}(\mathbf{t})$ is extracted from equation (**2.5**) and converted to matrix form

$$\mathbf{X} = [\mathbf{x}(t_1) \quad \mathbf{x}(t_2) \quad \dots \quad \mathbf{x}(t_m)]^T \ . \tag{2.16}$$

In addition, the derivative of $\mathbf{x}(t)$ is calculated and expressed in the same form

$$\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1) \quad \dot{\mathbf{x}}(t_2) \quad \dots \quad \dot{\mathbf{x}}(t_m)]^T \ . \tag{2.17}$$

With the contribution of the $\mathbf{X}$ data, a library Θ containing nonlinear function candidates is created

$$\Theta(\mathbf{X}) = [1 \quad \mathbf{X} \quad \mathbf{X^2} \quad \dots \quad \mathbf{X}^d \quad \dots \quad \sin(\mathbf{X}) \quad \dots] \ . \tag{2.18}$$

The library can contain constants, polynomial and trigonometric terms. $\mathbf{X}^d$ represents the column vectors covering the time series of $d-$th degree polynomials in the state $\mathbf{x}$. That is, each column in the library matrix is a candidate function of the right hand side of equation (**2.5**) . In this way, it provides freedom of entry for the $\Theta(\mathbf{X})$ matrix containing nonlinear states. Very few nonlinear entries in each row of the $\Theta(\mathbf{X})$ matrix contribute to the **f** function. By integrating the sparse regression, it is aimed to find the sparse vectors of the coefficients that determine which nonlinearities in the dynamic equation (**2.5**) are active for one of the row equations $\dot{\mathbf{x}}_k = \mathbf{f}_k(\mathbf{x})$.

$$\Xi = [\xi_{\mathbf{1}} \quad \xi_{\mathbf{2}} \quad \dots \quad \xi_{\mathbf{n}}] \ . \tag{2.19}$$

The dynamic system which is defined in equation (**2.5**) can be rewritten as shown in the equation below

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi \ . \tag{2.20}$$

As can be seen in figure **2.3** [15], the time-dependent **x** states and derivatives of **x** of a Lorenz system are collected and expressed in matrix form. Then the nonlinear function library is prepared. The library is used to find the least number of nonlinear terms that

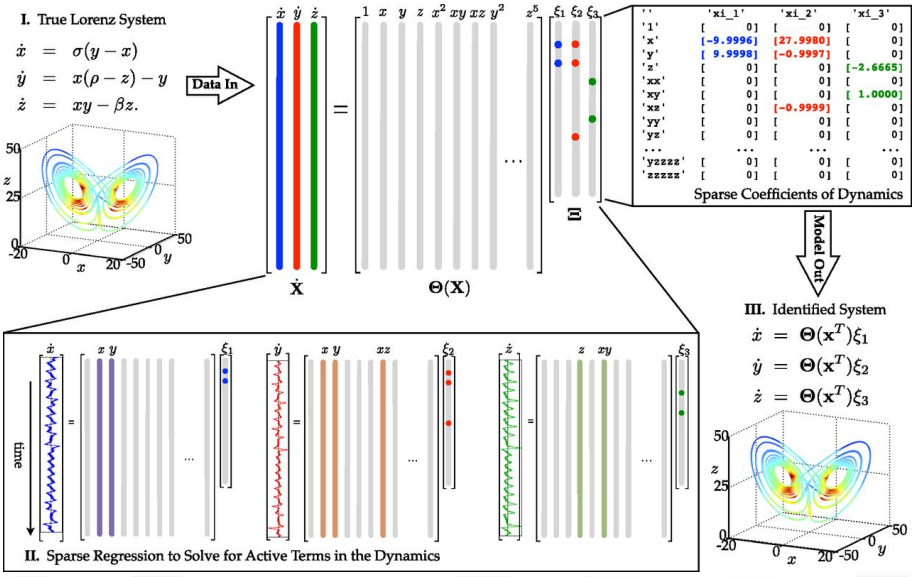

Figure 2.3 : Schematic of the SINDy algorithm, demonstrated on the Lorenz equations.

satisfy the condition given in equation (**2.20**). Active nonlinear terms are obtained by applying sparse regression to Ξ vectors.

Once Ξ is determined, each row of the governing equations can be modeled as

$$\dot{\mathbf{x}}_k = \mathbf{f}_k(\mathbf{x}) = \Theta(\mathbf{x}^T)\xi_k \ . \tag{2.21}$$

It is important to note the difference between the two definitions. $\Theta(\mathbf{X})$ is the data matrix and $\Theta(\mathbf{x})$ is a row vector of symbolic functions of $\mathbf{x}$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \Xi^T(\Theta(\mathbf{x}^T))^T \ . \tag{2.22}$$

2.2.1 PySINDy : A Python package for the sparse identification of nonlinear dynamics from data

In this subsection, I followed the paper [16] by Brian M. de Silva, K. Champion, M. Quade, J. C. Loiseau, J. N. Kutz and S. L. Brunton. PySINDy provides tools for applying the SINDy approach to model discovery from dynamical systems. Figure **2.4** shows the submodules of the `pySINDy` package. The SINDy model class acts as a scikit-learn [20] estimator and is the most crucial object in the `pySINDy` package.

The `pySINDy` integration includes three main steps that result in three modeling decisions:

1. To approximate $\dot{\mathbf{X}}$ from $\mathbf{X}$ *numerical differentiation* is used.

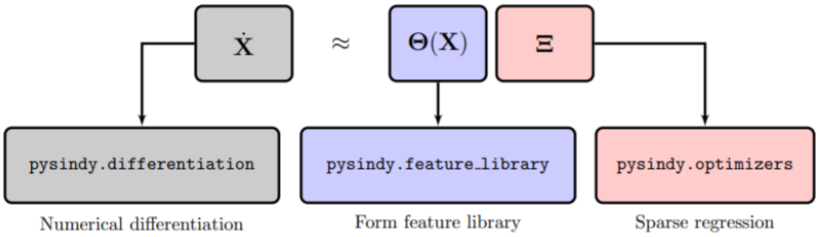

Figure 2.4 : The sparse regression problem solved by SINDy and the submodules of `pySINDy` [16].

2. The *feature library* created by candidate functions.

3. The *sparse regression* algorithm used to find $\mathbf{X}_i$ by solving equation (**2.20**).

The SINDy model class, which is the main object in the `pySINDy` implementation, enables three components to cooperate together :

1. Numerical differentiation
   (a) Finite difference : **FiniteDifference**
   (b) Smoothed finite difference : **SmoothedFiniteDifference**

2. Feature library
   (a) Trigonometric functions : **FourierLibrary**
   (b) Custom library that users add functions : **CustomLibrary**
   (c) Polynomials : **PolynomialLibrary**
   (d) Identity library : **IdentityLibrary**

3. Optimizer
   (a) Sequentially thresholded least-squares : **STLSQ**
   (b) Sparse relaxed regularized regression : **SR3**

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & \dots & \dots & a_{n-1} \\ a_1 & a_2 & & & & \vdots \\ a_2 & & & & & \vdots \\ \vdots & & & & & a_{2n-4} \\ \vdots & & & & a_{2n-4} & a_{2n-3} \\ a_{n-1} & \dots & \dots & a_{2n-4} & a_{2n-3} & a_{2n-2} \end{bmatrix}.$$

Figure 2.5 : Hankel matrix.

2.3 Hankel Matrix

Time series data of a dynamic system contains important information about the system. Hankel matrix method is a easy but very powerful way to understanding the dynamic structure. Embedology is a effective mathematical idea. It is based on the delay coordinates technique. The time delay coordinates was firstly proposed by Packard et al. [21] and Takens et al. [22]. The Hankel matrix is a square matrix obtained by shifting the data in each row by one step. Thus the elements throughout each diagonal are equal in the Hankel matrix. The Hankel matrix can be constructed by stacking shifted time-series values through desired delay coordinates. Hankel matrix is shown in figure **2.5**.

2.4 Singular Value Decomposition

The Singular Value Decomposition (SVD) method allows high-dimensional systems to be reduced to low-dimensional expressions with key features. Suppose we are analyzing a large data set $\mathbf{X} \in \mathbb{C}^{n \times m}$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \dots & x_m \\ | & | & & | \end{bmatrix} \tag{2.23}$$

The columns $\mathbf{x}_k \in \mathbb{C}$ may be measurements from experiments. The k index is a label denoting the k^{th} set of measurements. $\mathbf{X}$ represents a time dependent data set and $\mathbf{x}_k = \mathbf{x}(k\Delta t)$. Also, n is state-dimension and can be on the order of millions of degrees of freedom. Number of columns m is also called as snapshots of $\mathbf{X}$. SVD is a method that can be used to perform matrix decomposition for every complex-valued matrix

$\mathbf{X} \in \mathbb{C}^{n \times m}$ as

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{*} \tag{2.24}$$

where $\mathbf{U} \in \mathbb{C}^{n \times n}$, the left singular vector, and $\mathbf{V} \in \mathbb{C}^{m \times m}$, the right singular vector, are unitary matrices with orthonormal columns. Here, "$*$" denotes complex conjugate transpose, $\mathbf{\Sigma} \in \mathbb{R}$ is a matrix with zeros at off diagonal and containing decreasing, non-negative values (i.e.) on its diagonal called the singular values.

In Σ diagonal matrix, all columns are ordered by importance such as $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \geq \sigma_m \geq 0$. It means, 1^{st} column of $\mathbf{U}$ and $\mathbf{V}$ corresponding to σ_1 is more important than 2^{nd} columns $\mathbf{U}$ and $\mathbf{V}$ and 2^{nd} column of $\mathbf{U}$ and $\mathbf{V}$ corresponding to σ_2 is more important than 3^{th} columns of $\mathbf{U}$ and $\mathbf{V}$. This allows the model to ignore very small singular values. So, the approximate $\mathbf{X}$ matrix is expressed with only in terms of the first few dominant columns of $\mathbf{U}$ and $\mathbf{V}$ and the first dominant singular values of Σ. When the values are so small that they can be neglected, we truncate the process in rank r.

$$\mathbf{X} = \sigma_1 u_1 {v_1}^T + \sigma_2 u_2 {v_2}^T + \quad \ldots \quad \sigma_m u_m {v_m}^T + 0 \approx \tilde{U}\tilde{\Sigma}\tilde{V}^T \tag{2.25}$$

This is called rank r approximation. The basis of this approach is the Eckart-Young theorem. The theorem states that the best absolute approximation to the matrix $\mathbf{X}$ has rank r.

2.5 Long Short-Term Memory

Long short-term Memory (LSTM) is a special variant of recurrent neural network (RNN) which is capable of learning long-term dependencies from the training data. So it is necessary to examine RNN first. Recurrent neural networks were first mentioned in David Rumelhart's 1986 article [23]. To understand RNN, we should first review the working principle of a feedforward network. It can be said that it is a structure that produces output by applying some mathematical operations to the information coming to the neurons. Incoming information in the feedforward running structure is only forwarded. An output value is obtained by passing the input data through the network. Error is found by comparison of the obtained output value and the correct values. Depending on the error the weight values are changed. So, a model is created that can produce the most accurate result. RNN's inputs and outputs work

sequentially. Each output of RNN progresses depending on the previous step. It keeps the results calculated in the previous steps in its memory. If a very long input data is given to the RNN, it is incapable of remembering past information and therefore has difficulty in making predictions. Therefore, RNN can not work well for some problems because it has a short-term memory. The LSTM variant was designed by Hochreiter and Schmidhuber in 1997 [24] to eliminate the short-term memory problem.

All RNNs are in the form of a chain of repeating RNN cells. As can be seen in figure **2.6** the repeating cell in a standard RNN contains a single tanh layer.

The structure of the LSTM method is the same as the RNN structure. The important difference is that there are four interacting layers in repeating cell as shown in figure **2.7**. The basic concept of LSTM consists of the cell state and various gates it uses. Cell state can be explained as a communication line and network memory that carries meaningful information across cells to make predictions. The LSTM can add or subtract information from the cell state, which is carefully controlled by structures known as gates. Gates decides what information will be remembered or forgotten.

An LSTM has three of these gates to control the cell status. “Forget gate layer” uses the sigmoid activation function, which compresses the data to a range of 0 to 1. The information from the previous cell h_t and the current information x_t is inserted into the gate and if the result of sigmoid activation is 0, the information is forgotten and if it is 1, the information continues to progress with the cell state. Another sigmoid layer is the “input gate layer". This gate decides what information to update. It also includes a tanh layer. The tanh layer creates a vector for new values that can be added to the state cell. Then it is decided which information to update by multiplying the results of the sigmoid and tanh functions, it is decided which information to update. “Output gate layer" runs a sigmoid layer first to decide which parts of the cell state is going to be output. Then the tanh activation function is used to compress the data to the range of -1 and 1. [25]

[1] https://colah.github.io/posts/2015-08-Understanding-LSTMs
[2] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

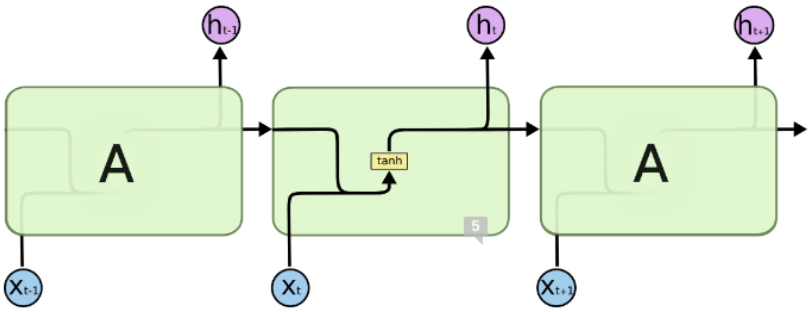

Figure 2.6 : The repeating cell in a RNN contains a single layer. Figure courtesy [1].

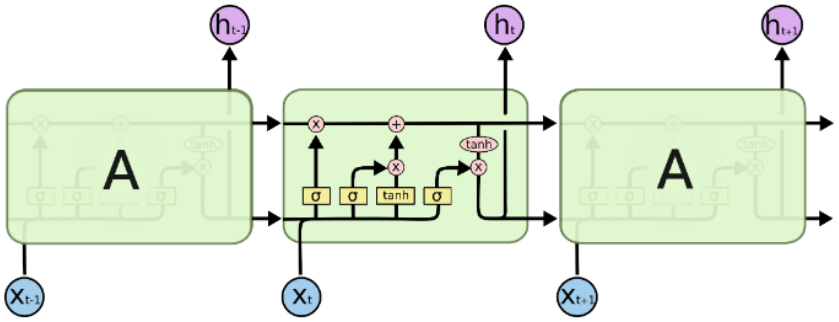

Figure 2.7 : The repeating cell in an LSTM contains four interacting layers. Figure courtesy [2].

The LSTM architecture may seem more complex than other networks. Its architecture allows it to be used in state of art deep learning applications such as natural language processing and speech recognition. In this thesis, we defend that LSTM will be advantageous for predicting the dates of glitches that occur in spin evolution of pulsars.

3. RESULTS

Within the scope of this thesis, the equations governing the spin evolution of pulsars are obtained by applying machine learning algorithms to the pulsar timing data sets. In this chapter we present the results we obtained by SINDy and LSTM.

Various open source software is used while applying machine learning methods, visualization and scientific calculations. The software used are as follows, `pandas` [26], `seaborn`[27], `matplotlib` [28], `numpy` [29], `scipy` [30], `pySINDy`[16] and `keras` [31]. Also, `jupyter notebook` [1] was used as development environment.

This chapter includes, the data collection process in section **3.1**, detrending of data sets process in section **3.2** construction of Hankel matrix and application of singular value decomposion method in section **3.3**, application of *SINDy* to Crab and Vela pulsar timing data set in section **3.4** and application of *LSTM* to Vela pulsar timing data set in section **3.5**. You can find the code of SINDy model in appendix A and LSTM model in appendix B.

3.1 Vela and Crab Pulsar Data Sets

The spin period P and the first time derivative of the spin period $\dot{P}$ are the two most important features used to understand the spin evolution of pulsars. While deriving the governing equations of pulsar spin evolution, we wanted to apply the `pySINDy` model to two well known pulsars. For this purpose, we used an 11-year data set of Vela pulsar [32] and a 39-year data set of Crab pulsar [6] [2]. Modified Julian Date (MJD), frequency ν and first time derivative of frequency $\dot{\nu}$ columns were used from both data sets. MJD is defined as the Julian Date - 2400000.5. The Crab pulsar has 403 rows and has MJD values between 47296 and 58917. Vela pulsar data set has 197 rows and MJD values between 54692 and 58839.

[1] `https://jupyter.org/`

[2] `http://www.jb.man.ac.uk/pulsar/crab.html`

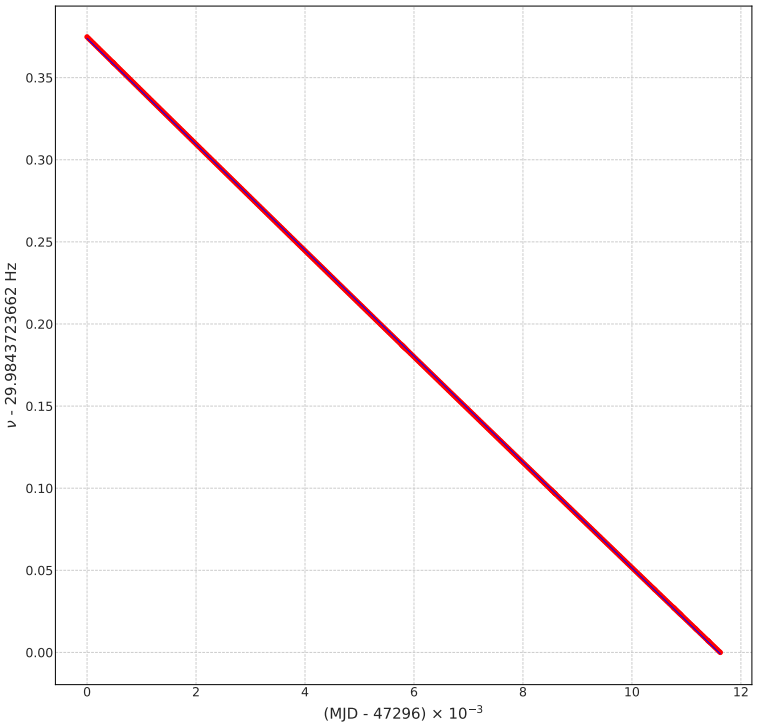

Figure 3.1 : Crab pulsar data set: The observational values of spin frequency ν (red dots) and the trend of data (blue solid line).

3.2 Detrending Data Sets

It is seen that there is a linearly decreasing trend in the data sets. In order to focus on the smaller changes, a linear fit

$$\bar{\nu} = \bar{\dot{\nu}} t + \bar{\nu}_0 \tag{3.1}$$

was applied to the data set of the Crab pulsar to obtain

$$\bar{\dot{\nu}} = -0.03224646$$

$$\bar{\nu}_0 = 0.37390197.$$

The Figure **3.1** shows the observational data and its trend. When we subtract this linear trend from the observation data, we find that there is still a quadratic trend as shown in Figure **3.2**.

We thus fitted the data by a quadratic equation $\nu - \bar{\nu} = at^2 + bt + c$ to obtain

$$a = 3.88173258 \times 10^{-5} \tag{3.2}$$

$$b = -4.55928446 \times 10^{-4} \tag{3.3}$$

$$c = 8.88413504 \times 10^{-5}. \tag{3.4}$$

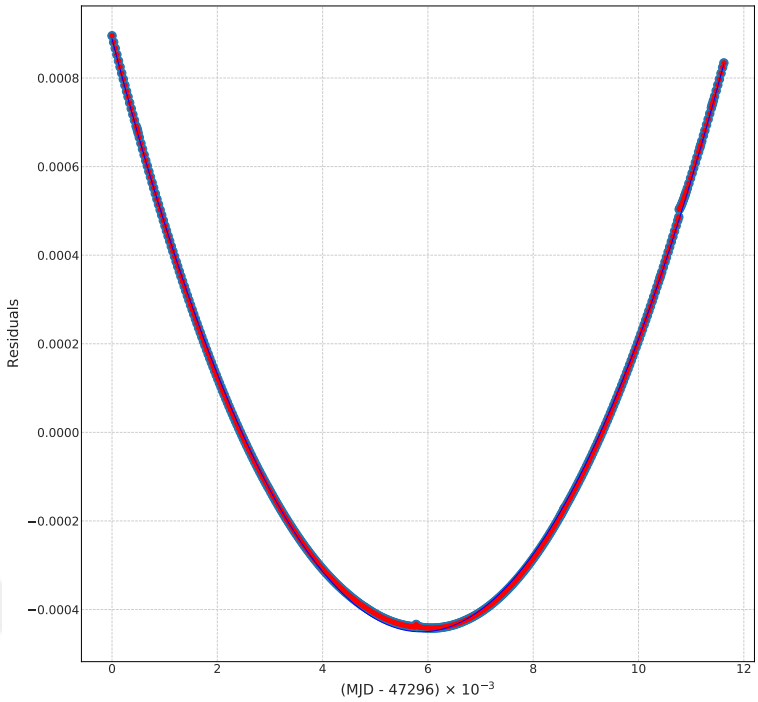

Figure 3.2 : Crab pulsar data set: The observational values of spin frequency ν (red dots) and the quadratic fit to the data (blue solid line).

The residuals obtained by subtracting the trend from the observation data are shown in Figure **3.3**.

The equation (**3.1**) was applied for data set of the Vela pulsar. The equation gave 5.48200441 for the intercept and -1.3236143 for the slope.

$$\bar{\nu} = -1.3236143t + 5.48200441 \tag{3.5}$$

The trend of the observation data is shown in Figure **3.4**. The residuals obtained by subtracting the trend from the observation data are shown in Figure **3.5**.

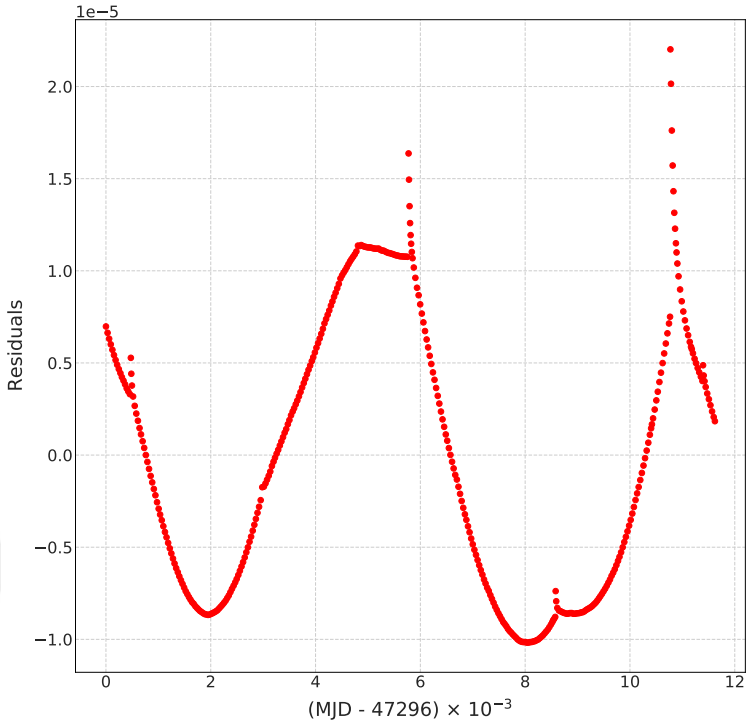

Figure 3.3 : Crab pulsar data set: Residuals after the quadratic trend is eliminated.

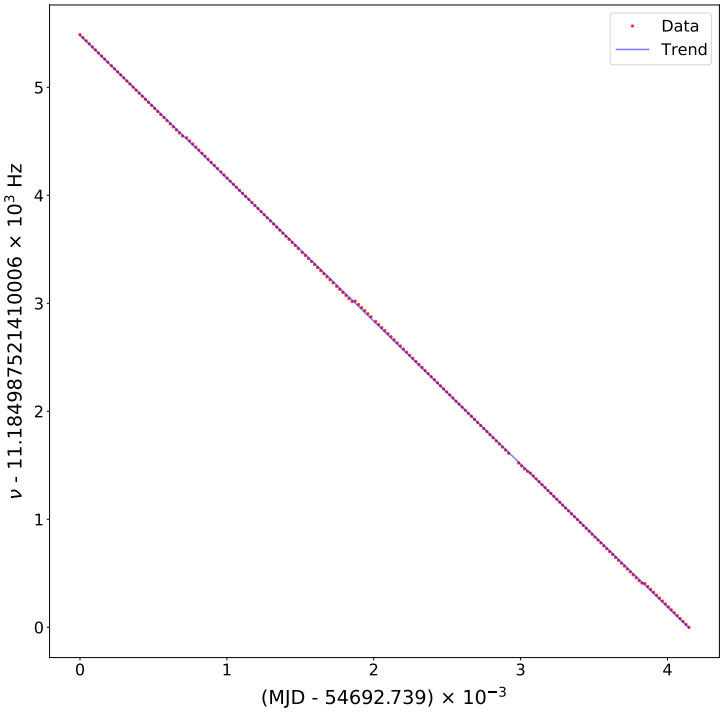

Figure 3.4 : Vela pulsar data set: The observational values of spin frequency ν (red dots) and the trend of data (blue solid line).

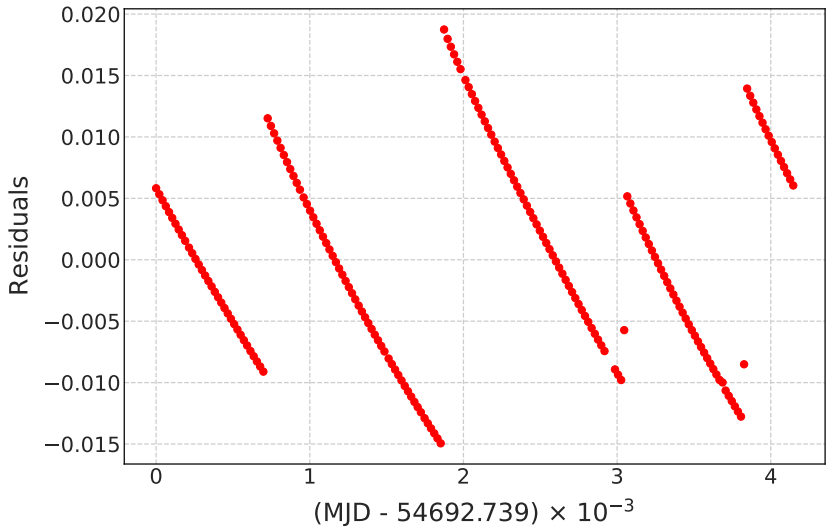

Figure 3.5 : Vela pulsar data set: Residuals after the linear trend is eliminated.

3.3 Application of Singular Value Decomposition

`scipy.linalg.hankel` and `scipy.linalg.svd` packages were imported to implement SVD to Crab and Vela data. Firstly, I constructed a Hankel Matrix from frequency data. The Hankel matrix is a square matrix obtained by shifting the data in each row by one step. Thus the elements throughout each diagonal are equal in the Hankel matrix [33]. Hankel matrix set up this way determines the covariance between the future and past stacked vectors [34].

The Hankel matrix's singular value magnitudes are useful in determining the dimension of the dynamic model. As shown in Equation (**2.24**), singular value decomposition (SVD) expresses an $m \times n$ matrix $\mathbf{X}$. $\mathbf{S}$ is an $m \times n$ diagonal matrix with singular values of $\mathbf{X}$ on its diagonal. The columns of the $m \times m$ matrix $\mathbf{U}$ are the left singular vectors for corresponding singular values. The columns of the $n \times n$ matrix $\mathbf{V}$ are the right singular vectors for corresponding singular values. $\mathbf{V}^T$ is the Hermitian transpose of $\mathbf{V}$. The Σ diagonal matrix is formed as a vector of singular values. To compute the singular value decomposition of a matrix I used the `svd()` function in `scipy`. This function allows the calculation of singular values of a matrix individually or both singular values and singular vectors in a single function call [35]. The function takes Hankel matrix and returns the $\mathbf{U}$, Σ and $\mathbf{V}^T$ elements.

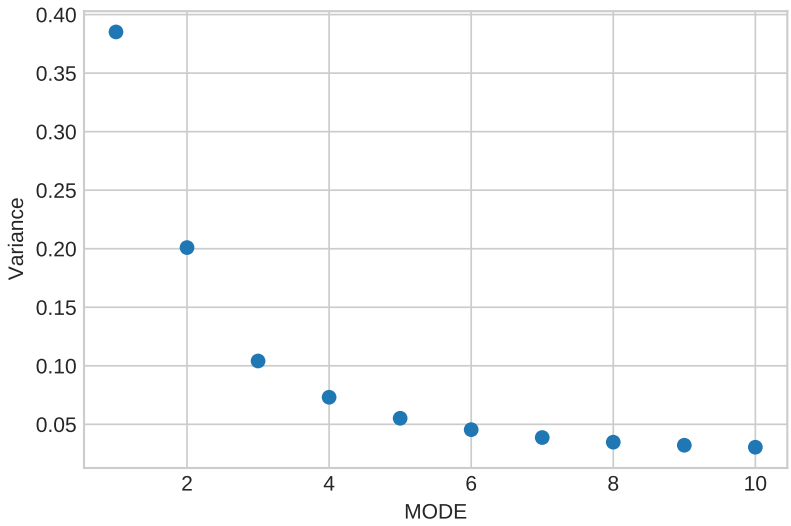

Figure 3.7 : Variance of the modes of SVD for Vela pulsar

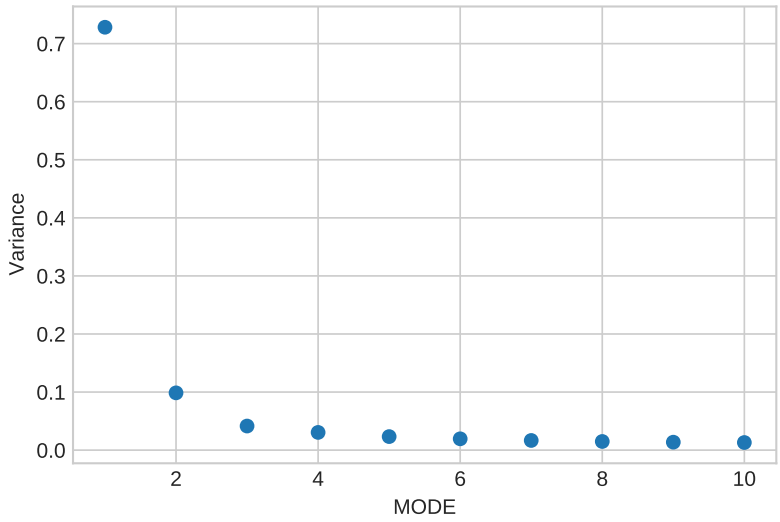

Figure 3.6 : Variance of the modes of SVD for Crab pulsar

The **V** matrix is formed in a transposed form. The graph of the variances of SVD modes was obtained as in Figure **3.7** for Vela pulsar and in Figure **3.6** for Crab pulsar data sets. As suggested by the SVD method, the initial modes have a greater effect on the system. Dimension number is chosen as 3 for the Crab pulsar and the Vela pulsar data sets.

Time delay coordinates are shown in Figure **3.8** for Crab pulsar. Time delay coordinates are shown in Figure **3.9** for Vela pulsar.

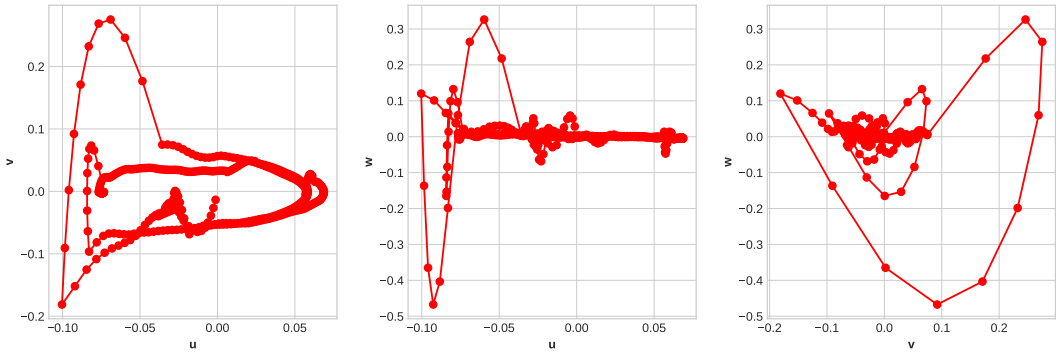

Figure 3.8 : Time delay coordinates embedding for Crab pulsar data set.

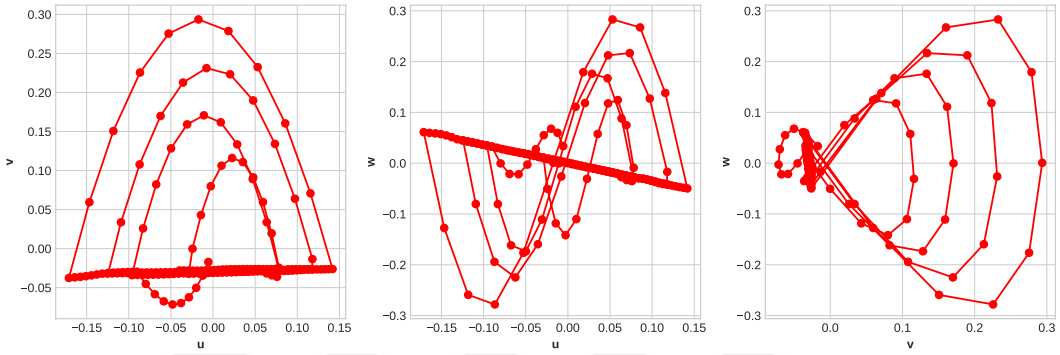

Figure 3.9 : Time delay coordinates embedding for Vela pulsar data. set.

3.4 Application of Sparse Identification of Nonlinear Dynamics from Data

As stated in subsection **2.2.1** `pySINDy` package revolves around the SINDy class which consists of three primary components which are (i) differentiation method that calculates $\dot{\mathbf{X}}$, (ii) feature library that specifies candidate basis functions and (iii) the optimizer that implements sparse regression to resolve x_i. The SINDy model is applied to Crab and Vela pulsars separately. Differentiation method is selected as `FiniteDifference` and 2^{nd} order, feature library is chosen as 4^{th} degree `PolynomialLibrary` and optimizer was selected as `STLSQ` with a threshold value of 0.4 for Crab pulsar and differentiation method is selected as `FiniteDifference` and 2^{nd} order, feature library is chosen as 2^{nd} degree `PolynomialLibrary` and optimizer was selected as `STLSQ` with a threshold value of 0.01 for Vela pulsar. Also, in `model.fit` function, the precision is set to 10 for both. When running the model, time delay data and normalized MJD data are used as inputs.

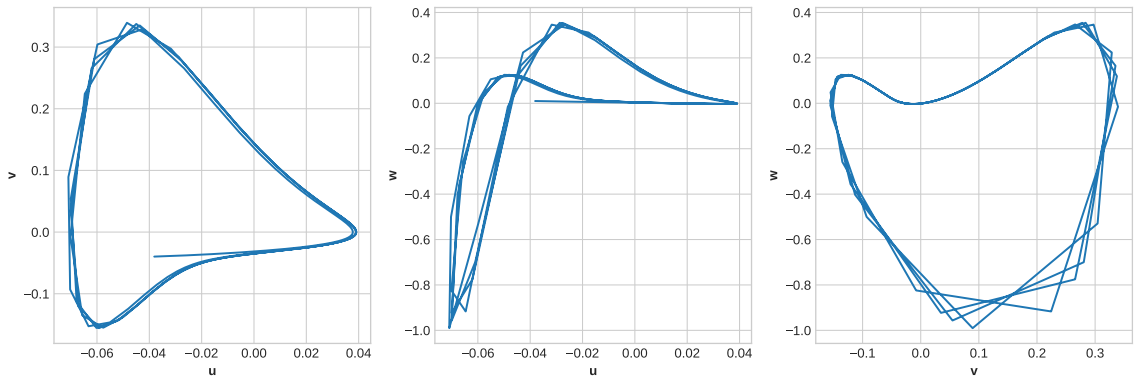

Figure 3.10 : Crab pulsar data Set: SINDy solutions of governing equations.

The dynamic system equations are obtained for the Crab pulsar as follows

$$\frac{dx}{dt} = -1.725y \tag{3.6}$$

$$\frac{dy}{dt} = 1.530x + 6.850z - 1.104z^2 \tag{3.7}$$

$$\frac{dz}{dt} = -0.067x + 4.358y - 15.388z + 50.309xy + 144.244y^2 + 160.964yz + 5.860z^2 - 1062.686y^3. \tag{3.8}$$

The `pySINDy` solutions of the governing equations of the Crab pulsar are shown in figure **3.10**.

The dynamic system equations are obtained for Vela pulsar as follows

$$\frac{dx}{dt} = 0.048 - 0.294x - 7.219y + 0.040z - 21.018x^2 - 5.195xy - 37.508xz + 4.879y^2 - 0.831yz + 7.715z^2 \tag{3.9}$$

$$\frac{dy}{dt} = 11.021x + 0.833y + 14.921z + 0.333x^2 + 103.367xy + 4.913xz - 60.731yz \tag{3.10}$$

$$\frac{dz}{dt} = 2.074x - 3.406y - 4.595z + 154.825x^2 + 114.452xy + 342.754xz - 94.749y^2 + 16.547yz - 45.238z^2. \tag{3.11}$$

The `pySINDy` solutions of the governing equations of the Vela pulsar are shown in figure **3.11**.

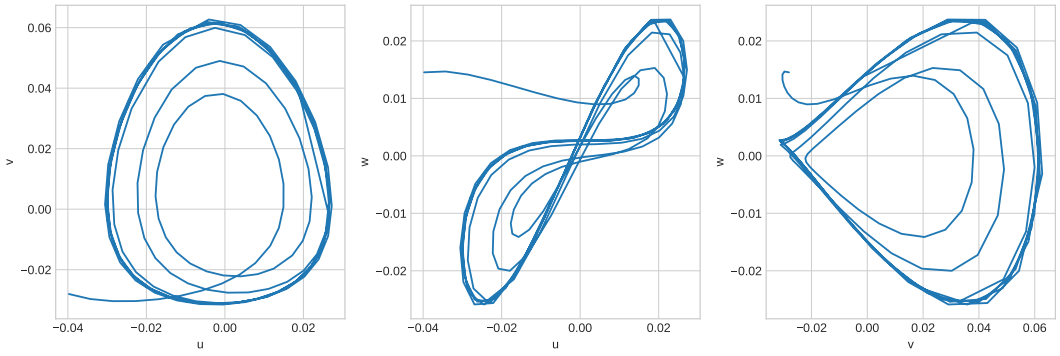

Figure 3.11 : Vela pulsar data set: SINDy solutions of governing equations.

Table 3.1 : Model scores of SINDy.

PSR	MODEL SCORE
B0531+21(Crab Data Set)	0.55
B0833-45 (Vela)	0.87

3.4.1 Model score

Calculation of the success of the model is provided by the R^2 metric. R^2 is a regression score function. Represents the ratio of the variance explained by the independent variables in the model. It shows the integrity of fit and thus provides a measure of how well-unseen pieces can be predicted by the model, through the ratio of defined variance. This metric is given as

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{3.12}$$

where,

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{3.13}$$

Parameter y_i is corresponding true value for total n samples and $\hat{y}_i$ is predicted value of the $i-$th sample. Best possible score is 1.0 in this metric. The evaluation score gained from R^2 metric is obtained for each pulsar as shown in figure **3.1**.

3.5 Application of Long Short-Term Memory

Main aim in this part of the thesis is to try to predict the glitches that occur in the spin evolution of pulsars with the LSTM method. The glitches in the Vela pulsar are easily recognizable. For this reason, it was suitable for analysis with the LSTM method. I used the `Keras` [3] framework and `keras.layers.LSTM` layer to apply the LSTM method to Crab and Vela pulsar. Keras is a user-friendly neural network (NN) library written in Python. First, the data sets of Crab and Vela pulsar are inserted. For each pulsar a data set was created for the model's inputs and outputs. The created data set was divided as training and test data sets. Sixty percent of data set were kept for training and the remaining fourty percent for testing for Vela pulsar and seventy percent of data set were kept for training and the remaining thirty percent for testing for Crab pulsar. The sequential type model is used when implementing LSTM. First, an LSTM layer was used when constructing neural network. This layer has 128 outputs. A dropout layer with rate 0.3 has been added after the LSTM layer. Dropout layer tries to prevent the system from memorizing. Finally, a dense layer was added as an output layer. The `model.compile` function used to compile the model was defined with two parameters: `optimizer` and `loss`. "Adam" algorithm was chosen as optimizer and sets the learning rate during training. The learning rate determines how quickly optimal weights are calculated for the model. "meansquarederror" regression loss function was chosen as loss. Mean squared error is calculated as the average of the squared differences between the predicted and actual values. To train the model, we used `model.fit` function in our model with the following seven parameters: `X train`, `Y train`, `epochs`, `batch size`, `validation data`, `verbose` and `shuffle`. `X train` represents training data and `Y train` represents target data. *Verbose* is used to see outputs during training and it is set to 1 in the model. `Batch size` is the amount of data to be trained simultaneously and it is set to 1. `Shuffle` is defined to change the position of the data in each epoch. `Epochs` specifies how many times the data set will be trained on the model and it is chosen as 100. The error analysis of the model we obtained with the LSTM method for Crab and Vela pulsar is shown in figure **3.2**.

[3] `https://keras.io/`

Table 3.2 : Error analysis of LSTM method.

PSR	TRAIN MAE	TRAIN RMSE	TEST MAE	TEST RMSE
B0531+21(Crab)	0.09	0.11	0.08	0.11
B0833-45 (Vela)	0.04	0.11	0.04	0.08

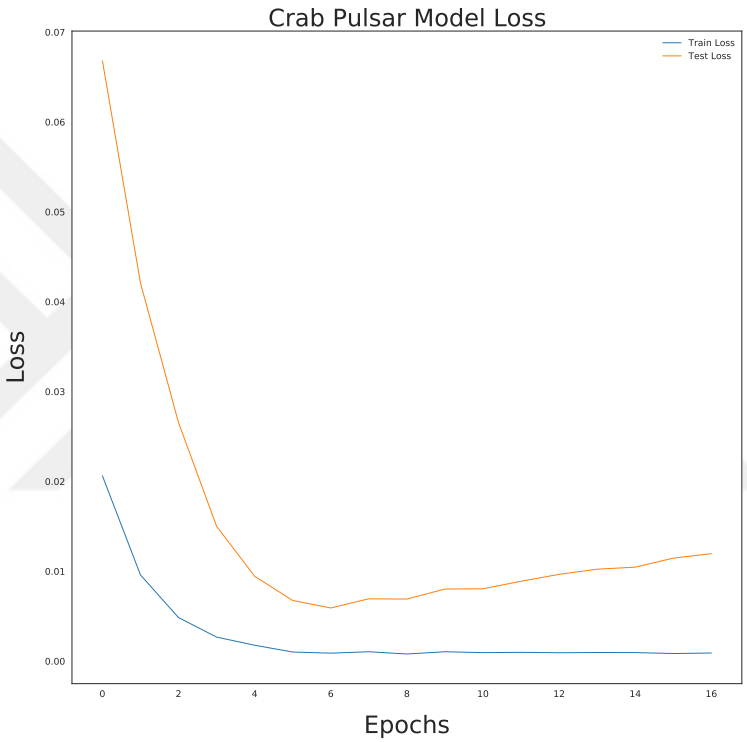

Figure 3.12 : Crab pulsar data set : Model loss.

Afterwards, the model loss graph containing the train loss and test loss information was obtained for Crab and Vela pulsars as in figure **3.12** and **3.13**. The comparison graph of predicted training and test data with Crab pulsar observation data is shown in figure **3.14**. The comparison graph of predicted training and test data with Vela pulsar observation data is shown in figure **3.15**.

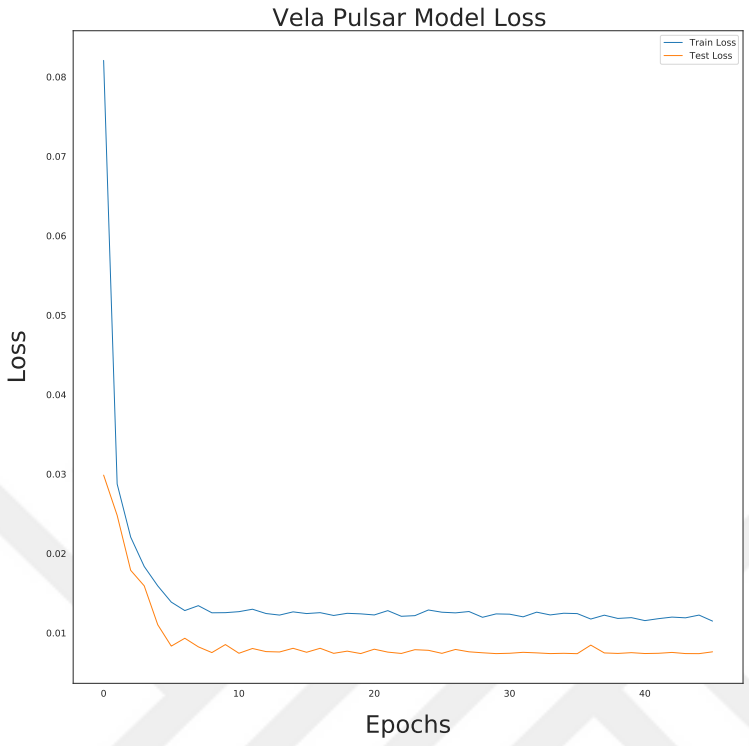

Figure 3.13 : Vela pulsar data set : Model loss.

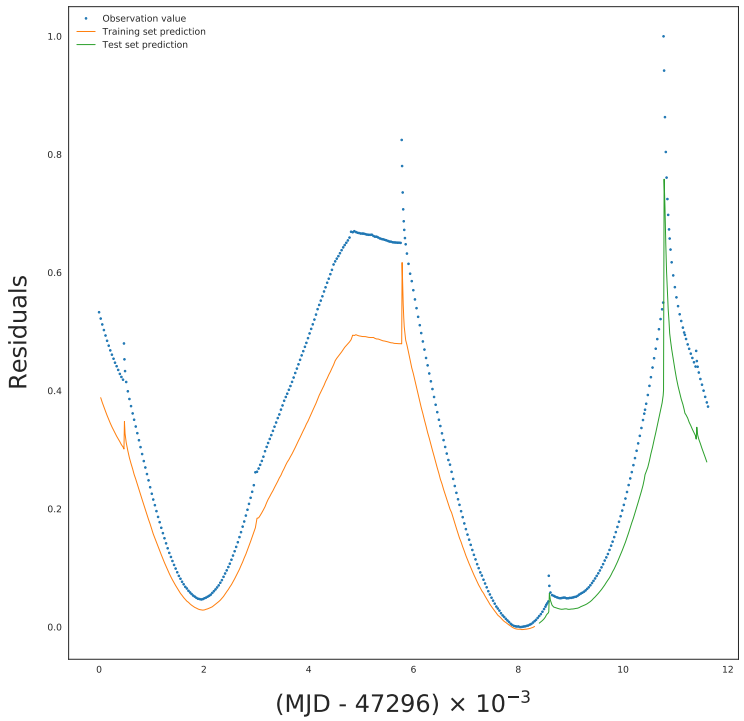

Figure 3.14 : Crab pulsar data set : Comparison on predicted training and test data vs. observation data.

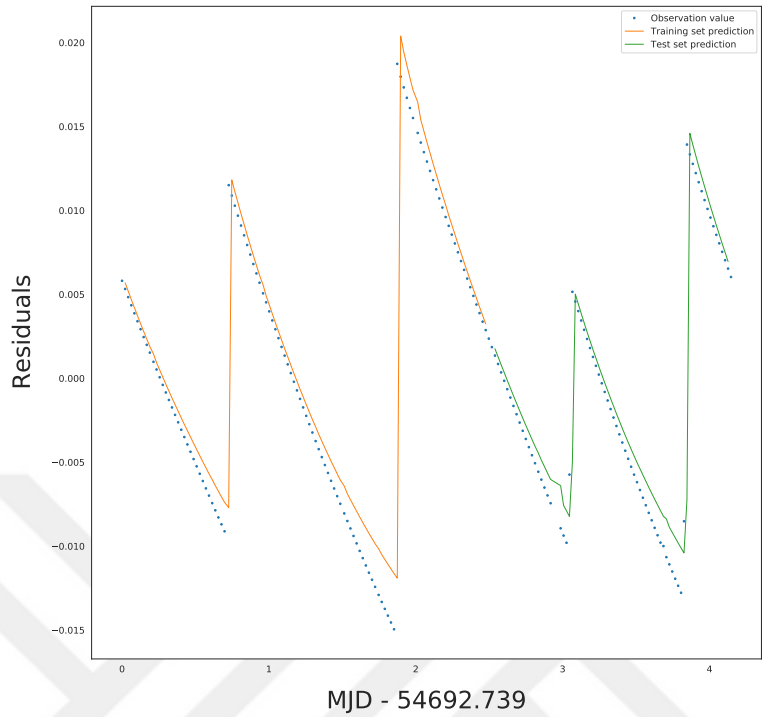

Figure 3.15 : Vela pulsar data set : Comparison on predicted training and test data vs. observation data.

4. CONCLUSION

In this thesis, we attempted to determine the spin evolution of two frequently observed pulsars by machine learning methods.We applied SINDy and LSTM methods to the observational data of Crab and Vela pulsars containing long-term spin frequencies.

First, we applied the SVD method to both pulsars using Hankel matrix to find the number of variables affecting the system. Then by using the SINDy method, we identified a set of equations that defined the spin evolution of these pulsars. The options featured in the SINDy method are the optimizer, the numerical differentiation method, and the feature library selection. An important parameter of the method is the "threshold value" on which the regression is based on. This parameter definition indicates that polynomials with weights less than the threshold are not taken into account. We chose the threshold value 0.01 for Vela pulsar and 0.4 for Crab pulsar. When we applied the SINDy model to detrended pulsar datasets, we obtained differential equations the solutions of which give the spin-frequency evolution. Thus, the equations obtained can be assumed to govern the spin evolution of pulsars.

We solved the differential equations numerically starting with the initial values. As a result, we obtained the spin frequency evolution with the SINDy model. Model scores obtained with the R^2 metric are 0.87 for Vela, 0.55 for Crab pulsar respectively. Model score is great for Vela pulsar and acceptable for Crab pulsar. We tried to understand the topology of the dynamic system with the comparative graphs of the equation solutions. Then we compared the SINDy solution graphs with time delayed graphs. Their topologies were similar in general terms. We have obtained dynamic system equations that will help us to understand pulsar spin evolution.

To apply the LSTM method, we used sixty percent of the spin frequency data of Vela pulsar for training and the remaining forty percent for testing. We have seen that there is no significant difference between the train loss and test loss parameters in the model loss graphs in **3.12** and **3.13**. In neural network (NN) structure, the model learns the structure from the training data which is then used for validation. If the validation loss

is increasing while the train loss is decreasing, the model is said to be overfitting. This is undesirable in NN models because it means that the method is not learning from the training data, but it is memorizing the data. When we examine the error analysis, we see that the train and test errors are close to both root mean square error (RMSE) and mean squared error (MSE). Our results prove that such a problem does not occur.

Our results show that the pulsar spin glitches causing abrupt changes in the pulsar spin evolution can be modeled by the LSTM method. Accordingly, the LSTM model can be employed to predict the date of pulsar glitches. This can be important in pulsar research since many instruments can target the pulsar on that date to achieve high time resolution data.

We think that the knowledge of pulsar spin evolution will be increased by applying the methods to more pulsar data.

REFERENCES

[1] **Manchester, R.N.**, **Hobbs, G.B.**, **Teoh, A. and Hobbs, M.** (2005). The Australia Telescope National Facility Pulsar Catalogue, *AJ*, *129*(4), 1993–2006, astro-ph/0412641.

[2] **Staelin, D.H. and Reifenstein, Edward C., I.** (1968). Pulsating Radio Sources near the Crab Nebula, *Science*, *162*(3861), 1481–1483.

[3] **Comella, J.M.**, **Craft, H.D.**, **Lovelace, R.V.E. and Sutton, J.M.** (1969). Crab Nebula Pulsar NP 0532, *Nature*, *221*(5179), 453–454.

[4] **Lyne, A.G.**, **Jordan, C.A.**, **Graham-Smith, F.**, **Espinoza, C.M.**, **Stappers, B. and Weltevrede, P.** (2014). 45 years of rotation of the Crab pulsar, *MNRAS*, *446*.

[5] **Lovelace, R.V.E. and Tyler, G.L.** (2014). On the discovery of the period of the Crab Nebular pulsar, *The Observatory*, *132*, 186–188.

[6] **G., L.A.**, **Pritchard, R.S. and F., G.S.** (1993). 23 years of Crab pulsar rotational history, *MNRAS*, *265*, 1003–1012.

[7] **Lyne, A.G.**, **Taylor, J.H. and Manchester** (1993). Catalog of 558 Pulsars, *Astrophysical Journal Supplement*, *88*, 529.

[8] **Ferdman, R.D.**, **Archibald, R.F. and Kaspi, V.M.** (2015). Long-term Timing and Emission Behavior of the Young Crab-like Pulsar PSR B0540-69, *The Astrophysical Journal*, *812*, 10.

[9] **Lyne, A.G.**, **Pritchard, R.S.**, **Graham-Smith, F. and Camilo, F.** (1996). Very low braking index for the Vela pulsar, *Nature*, *381*, 497–498.

[10] **Jayanta Roy, Yashwant Gupta, W.L.** (2012). Observations of four glitches in the young pulsar J1833-1034 and study of its glitch activity, *Monthly Notices of the Royal Astronomical Society*, *424*, 2213–2221.

[11] **Lyne, A.**, **Graham-Smith, F.**, **Weltevrede, P.**, **Jordan, C.**, **Stappers, B.**, **Bassa, C. and Kramer, M.** (2013). Evolution of the Magnetic Field Structure of the Crab Pulsar, *Science*, *342*(6158), 598–601, 1311.0408.

[12] **Beskin, V.S. and Nokhrina, E.E.** (2007). On the role of the current loss in radio pulsar evolution, *ApSS*, *308*(1-4), 569–573, astro-ph/0608689.

[13] **Blandford, R.D. and Romani, R.W.** (1988). On the interpretation of pulsar braking indices., *MNRAS*, *234*, 57P–60.

[14] **Menou1, K.**, **Perna, R. and Hernquist, L.** (2001). Stability and Evolution of Supernova Fallback Disks, *The Astrophysical Journal*, *559*.

[15] **Brunton, S.L.**, **Proctor, J.L. and Kutz, J.N.** (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems, *PNAS*, *113*, 3932–3937.

[16] **de Silva, B.M.**, **Champion, K.**, **Quade, M.**, **Loiseau, J.C.**, **Kutz, J.N. and Brunton, S.L.** (2020). PySINDy: A Python package for the Sparse Identification of Nonlinear Dynamics from Data, *The Journal of Open Source Software*, *49*.

[17] **Hochreiter, S. and Schmidhuber, J.** (1997). Long Short Term Memory, *Neural Computation*, *9*, 1735–1780.

[18] **Brunton, S.L. and Kutz, J.N.** (2019). *Data-Science and Engineering*, Cambridge University Press.

[19] **MARX, I. and PIRANIAN, G.** (1953). Lipschitz Functions of Continuous Functions, *Pacific Journal of. Mathematics.*, *3*.

[20] **Pedregosa, F.**, **Varoquaux, G.**, **Gramfort, A.**, **Michel, V.**, **Thirion, B.**, **Grisel, O.**, **Blondel, M.**, **Prettenhofer, P.**, **Weiss, R.**, **Dubourg, V.**, **Vanderplas, J.**, **Passos, A.**, **Cournapeau, D.**, **Brucher, M.**, **Perrot, M. and Édouard Duchesnay** (2011). Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, *12*(85), 2825–2830.

[21] **Packard, N.H.**, **Crutchfield, J.P.**, **Farmer, J.D. and Shaw, R.S.** (1980). Geometry from a Time Series, *Phys. Rev. Lett.*, *45*, 712–716.

[22] **Takens, F.**, (1981). Detecting Strange Attractors in Turbulence, Dynamical Systems and Turbulence, Warwick 1980, Springer, pp.366–381.

[23] **Rumelhart, D.E.**, **Hinton, G.E. and Williams, R.J.** (1986). Learning Representations by Back-propagating Errors, *Nature*, *323*(6088), 533–536.

[24] **Hochreiter, S. and Schmidhuber, J.** (1997). Long Short-Term Memory, *Neural Computation*, *9*(8), 1735–1780.

[25] **Goodfellow, I.**, **Bengio, Y. and Courville, A.** (2016). *Deep Learning*, MIT Press, `http://www.deeplearningbook.org`.

[26] **Wes McKinney** (2010). Data Structures for Statistical Computing in Python, ***Stéfan van der Walt and Jarrod Millman**, editors, Proceedings of the 9th Python in Science Conference*, pp.56 – 61.

[27] **Waskom, M.**, **Botvinnik, O.**, **O'Kane, D.**, **Hobson, P.**, **Ostblom, J.**, **Lukauskas, S.**, **Gemperline, D.C.**, **Augspurger, T.**, **Halchenko, Y.**, **Cole, J.B.**, **Warmenhoven, J.**, **de Ruiter, J.**, **Pye, C.**, **Hoyer, S.**, **Vanderplas, J.**, **Villalba, S.**, **Kunter, G.**, **Quintero, E.**, **Bachant, P.**, **Martin, M.**, **Meyer, K.**, **Miles, A.**, **Ram, Y.**, **Brunner, T.**, **Yarkoni, T.**, **Williams, M.L.**, **Evans, C.**, **Fitzgerald, C.**, **Brian and Qalieh, A.** (2018). mwaskom/seaborn: v0.9.0 (July 2018).

[28] **Hunter, J.D.** (2007). Matplotlib: A 2D graphics environment, *Computing In Science & Engineering*, *9*(3), 90–95.

[29] **Harris, C.R.**, **Millman, K.J.**, **van der Walt, S.J.**, **Gommers, R.**, **Virtanen, P.**, **Cournapeau, D.**, **Wieser, E.**, **Taylor, J.**, **Berg, S.**, **Smith, N.J.**, **Kern, R.**, **Picus, M.**, **Hoyer, S.**, **van Kerkwijk, M.H.**, **Brett, M.**, **Haldane, A.**, **del R'ıo, J.F.**, **Wiebe, M.**, **Peterson, P.**, **G'erard-Marchant, P.**, **Sheppard, K.**, **Reddy, T.**, **Weckesser, W.**, **Abbasi, H.**, **Gohlke, C. and Oliphant, T.E.** (2020). Array programming with NumPy, *Nature*, *585*(7825), 357–362.

[30] **Jones, E.**, **Oliphant, T. and Peterson, P.**, (2001), SciPy: Open Source Scientific Tools for Python.

[31] **Chollet, F.** *et al.*, (2015), Keras, `https://github.com/fchollet/keras`.

[32] **Gugercinoglu, E.**, **Ge, M.Y.**, **Yuan, J.P. and Zhou, S.Q.** (2020). Glitches in four gamma-ray pulsars and inferences on the neutron star structure, *MNRAS, submitted*.

[33] **Partington, J.R.** (1989). *An Introduction to Hankel Operators*, London Mathematical Society Student Texts, Series Number 13.

[34] **Aoki, M.** (1990). *State Space Modeling of Time Series*, Springer.

[35] **Klema, V. and Laub, A.** (1980). The singular value decomposition: Its computation and some applications, *IEEE Transactions on Automatic Control*, *25*(2), 164–176.

APPENDICES

APPENDIX A : SINDy model code.

APPENDIX B : LSTM model code.

APPENDIX A

Listing A.1: SINDy model code.

```
import numpy as np
import pysindy as ps
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
from scipy.linalg import hankel, svd

# Load data

veladata = np.loadtxt('Velafreq.txt')
nu = (veladata[:,1])
tv = (veladata[:,0])
nudot = (veladata[:,3])

# Normalize data

nudot_norm = 1E-12
day = 8.64E4
t_norm = 1E3
nu_norm = 1E3
tn = (tv - min(tv))/t_norm
nun = (nu - min(nu))*nu_norm
nudotn = nudot*nudot_norm*day*t_norm*nu_norm

# Detrend

theta = np.polyfit(tn, nun, 1)
y_line = theta[1] + theta[0]*tn
diff = nun - y_line

# Processed data

ydata = diff.copy()

# Hankel

q = 10
xh = hankel(ydata)[0:q,:]

# SVD

u, s, vh = svd(xh)
V = vh.T

# Time delay coordinates

x1 = V[:,0]
x2 = V[:,1]
x3 = V[:,2]

# Plot time delay coordinates
```

```
fig, ax = plt.subplots(1, 3, figsize=(12, 4), constrained_layout=True)
fig.suptitle("Time delay coordinates", fontsize=16)
ax[0].plot(x1, x2)
ax[0].set_xlabel("u")
ax[0].set_ylabel("v")

ax[1].plot(x1, x3)
ax[1].set_xlabel("u")
ax[1].set_ylabel("w")

ax[2].plot(x2, x3)
ax[2].set_xlabel("v")
ax[2].set_ylabel("w")
#plt.show()


# SINDy

feature_library = ps.PolynomialLibrary(degree=2)
optimizer = ps.STLSQ(threshold=0.01, alpha=0.5)
x_model = np.column_stack((x1,x2,x3))

model = ps.SINDy(
    #differentiation_method=differentiation_method,
    feature_library=feature_library,
    optimizer=optimizer,
    feature_names=["x","y","z"]
)

model.fit(x_model, t=tn)
model.print()
model.score(x_model, t=tn)

# Simulate for preferred initial conditions

sim = model.simulate(x_model[0], t=tn)

# SINDy reconstruction of the system in time delay coordinates

fig, ax = plt.subplots(1, 3, figsize=(12, 4), constrained_layout=True)
fig.suptitle("SINDy reconstruction", fontsize=16)
ax[0].plot(sim[:,0], sim[:,1])
ax[0].set_xlabel("u")
ax[0].set_ylabel("v")

ax[1].plot(sim[:,0], sim[:,2])
ax[1].set_xlabel("u")
ax[1].set_ylabel("w")

ax[2].plot(sim[:,1], sim[:,2])
ax[2].set_xlabel("v")
ax[2].set_ylabel("w")
plt.show()
```

APPENDIX B

Listing B.1: LSTM model code

```
import os
import sys
module_path = os.path.abspath(os.path.join('..'))
import pandas as pd
if module_path not in sys.path:
    sys.path.append(module_path)
from io import StringIO
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
from pandas import read_csv

# Load data

tv, nu, nudot = np.loadtxt('crabdata.txt', float,
usecols=(0,1,2), unpack = True, skiprows=1)

# Normalize data

day = 8.64E4
t_norm = 1E3
nu_norm = 1E3
tn = (tv - min(tv))/t_norm
nun = nu - min(nu)
nudotn = nudot*1E-15*day*t_norm

#Detrend data

theta = np.polyfit(tn, nun, 1)
print(f'The parameters of the line: {theta}')
y_line = theta[1] + theta[0] * tn

# Plotting the data points and the best fit line

plt.figure(figsize = (15, 15))
plt.scatter(tn, nun)
plt.plot(tn,y_line, 'r')
plt.title('Best fit line using numpy.polyfit()')
plt.xlabel('mjd')
plt.ylabel('nu')
plt.show()

# Processed data

diff = nun - y_line

#Detrend
```

```
omega = np.polyfit(tn, diff, 2)
print(f'The parameters of the line: {omega}')
y_line2 = omega[2] + omega[1]*tn+omega[0]*tn**2
# Plotting the data points and the best fit line
plt.figure(figsize = (15, 15))
plt.scatter(tn, diff)
plt.plot(tn,y_line2, 'r')
plt.title('Best fit line using numpy.polyfit()')
plt.xlabel('mjd')
plt.ylabel('nu')
plt.show()

#processed data

diff2=diff-y_line2
print(diff2)

diff2 = diff2.astype('float32')
diff2 = np.reshape(diff2, (-1, 1))
scaler = MinMaxScaler(feature_range=(0, 1))
diff2 = scaler.fit_transform(diff2)
train_size = int(len(diff2) * 0.90)
test_size = len(diff2) - train_size
train, test = diff2[0:train_size,:], diff2[train_size:len(diff2),:]

def create_dataset(diff2, look_back=2):
    X, Y = [], []
    for i in range(len(diff2)-look_back-1):
        a = diff2[i:(i+look_back), 0]
        X.append(a)
        Y.append(diff2[i + look_back, 0])
    return np.array(X), np.array(Y)

# reshape into X=t and Y=t+1

look_back = 1
X_train, Y_train = create_dataset(train, look_back)
X_test, Y_test = create_dataset(test, look_back)

# reshape input to be [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))

#LSTM model

model = Sequential()
model.add(LSTM(128, input_shape=(1, look_back)))

model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

history = model.fit(X_train, Y_train, epochs=100, batch_size=1,
validation_data=(X_test, Y_test), callbacks=[EarlyStopping(monitor='val_loss',
patience=10)],verbose=1, shuffle=False)

# Training phase
```

```
model.summary()
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# invert predictions

train_predict = scaler.inverse_transform(train_predict)
Y_train = scaler.inverse_transform([Y_train])
test_predict = scaler.inverse_transform(test_predict)
Y_test = scaler.inverse_transform([Y_test])
print('Train Mean Absolute Error:',
mean_absolute_error(Y_train[0],train_predict[:,0]))
print('Train Root Mean Squared
Error:',np.sqrt(mean_squared_error(Y_train[0],train_predict[:,0])))
print('Test Mean Absolute Error:',
mean_absolute_error(Y_test[0], test_predict[:,0]))
print('Test Root Mean Squared
Error:',np.sqrt(mean_squared_error(Y_test[0],test_predict[:,0])))

plt.figure(figsize = (15, 15))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Test Loss')
plt.ylabel('loss', labelpad=20, fontsize=30)
plt.xlabel('epochs', labelpad=20, fontsize=30)
plt.legend(loc='upper right')
plt.show();


train_predict_plot = np.empty_like(diff2)
train_predict_plot[:, :] = np.nan
train_predict_plot[look_back:len(train_predict) + look_back, :] = train_predict
print(len(train_predict_plot))

g=np.arange(0,12,1000)

test_predict_plot = np.empty_like(diff2)
test_predict_plot[:, :] = np.nan
test_predict_plot[len(train_predict) + (look_back * 2) + 1:len(diff2) - 1, :] =
test_predict

plt.figure(figsize = (15, 15))
plt.plot(tn, scaler.inverse_transform(diff2),"o", markersize=2,
label = "Observation value",linewidth=2.0)
plt.plot(tn, train_predict_plot, label = "Training set prediction")
plt.plot(tn, test_predict_plot, label = "Test set prediction")
plt.xlabel("Time" , labelpad=20, fontsize=30)
plt.ylabel("$\\nu$ (Hz)", labelpad=20, fontsize=30)
plt.title("Comparison true vs. predicted training / test")
plt.legend()
plt.show()
```

CURRICULUM VITAE

Name SURNAME: Esma HASANÇEBİ ESER

EDUCATION:

- **B.Sc.:** 2016, Istanbul Technical University, Faculty of Science and Literature, Physics Engineering
- **M.Sc.:** 2018, Istanbul Technical University, Informatics Institute, Information Technologies

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2016-2021 ING Bank, Senior IT consultant.
- 2018 Istanbul Technical University, Big Data and Business Analytics Certificate Program.