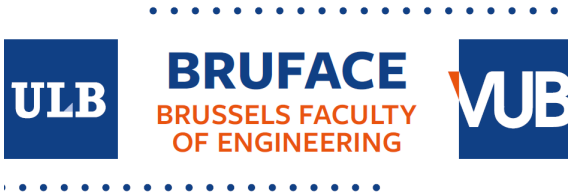


Vrije Universiteit Brussel & Université Libre de Bruxelles

Data-Driven Engineering

MECA-H-419

Project Report

Study and prediction of a pendulum motion using a data-driven approach

Authors:

Hamed Firouzipouyaei
Viktor Laurens De Groote
Robin Charissé
Maxime Monsieur

Professors:

Emanuele Garone
Mehrdad Teratani
Alessandro Parente

BRUFACE Master

Electromechanical Engineering:
Robotics and Mechanical Construction

Academic Year 2022-2023

Contents

List of Figures

List of Tables

1 Introduction

Models nowadays are becoming so complex that analytic or closed solutions are simply not feasible anymore. This is where the computational power of computers comes in very handy. Next to the 'traditional' science methods, new approaches are found to study systems, processes, phenomena, etc. solely based on acquired data. From this data, computer models are built which are becoming increasingly important.

1.1 Objectives

The objective of this project is to explore the vast methods and strategies that data and optimization offer. For this project, a (simple) mechanical system is chosen as the subject of the analysis, namely: a single pendulum with viscous friction. The pendulum serves as a classic example in physics that allows us to explore and understand fundamental concepts such as gravity, oscillatory motion, and conservation of energy. Despite its simplicity, the pendulum exhibits a rich variety of behaviors. This subject was chosen because it is intuitive, possible to build ourselves and, under certain assumptions, analytic solutions to the problem exist. This allows us to link the results from the data-driven methods to the theory and to explore the advantages and limitations of both and how these relate to each other.

The report is structured around the following objectives:

- Acquiring experimental data by tracking the ball from a home-made pendulum using computer vision, see Section 2.1. Next to the experimental data set, the theoretical formulation of the problem is used to build a virtual data set in Section 2.2. The acquired data is processed using outlier detection, see Section 2.3.
- Dimensionality reduction using PCA, see Chapter 3.
- Linear regression for prediction of the pendulum motion, see Chapter 4.
- Analysis of the dynamical system using POD in Chapter 5.
- Building a neural network for parameter classification and trajectory prediction, see Chapter 6.

2 Data-acquisition and Outlier Detection

The first step is to acquire the data. An experimental setup is built and the data is collected by tracking the ball using computer vision, see Section 2.1. However, performing these experiments are very time-consuming. A virtual data set is generated using the theoretical formulation of the problem, see Section 2.2. By using simulation to obtain a diverse data set, we can efficiently explore a wide range of parameter variations and observe the corresponding effects on the pendulum's motion. This approach saves significant time and resources compared to conducting physical experiments for each parameter combination.

2.1 Experimental dataset

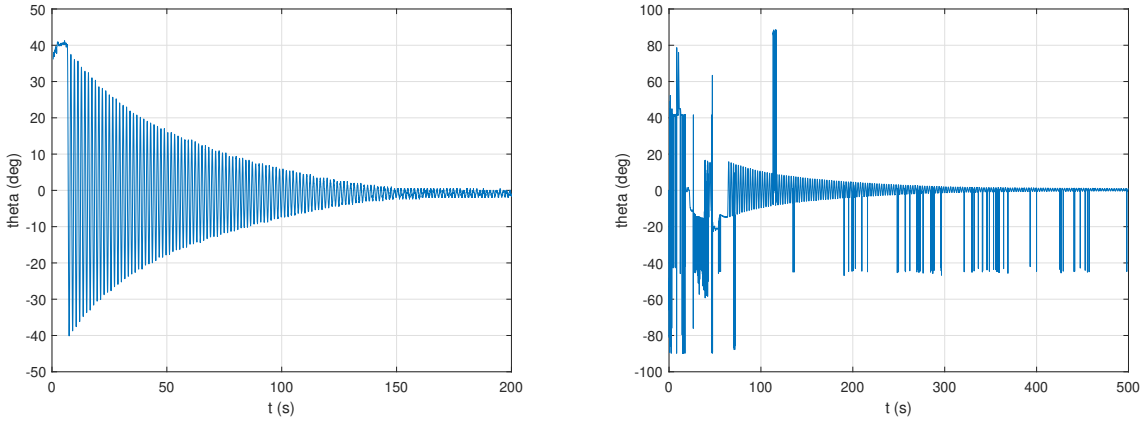
The pendulum is built by taping a small piece of wood to the wall. A screw is drilled into the wood to attach the rope. The tennis ball is attached to the rope by means of a knot. A triangle ruler is also attached to have an estimation of the (initial) angle. The complete setup is shown in Figure 2.1.

(a) Complete pendulum.

(b) Anchor point of the pendulum.

Figure 2.1: Pendulum that is used for gathering the experimental data set.

The *OpenCV* plugin for Python is used. *OpenCV* is a library mainly used for computer vision. The ball is tracked based on its color. The code reads the data coming from the camera of the laptop. Trackbars are added which allows for dynamically changing the range of HSV that the program detects while recording. This is more practical as lighting conditions might vary between experiments. The contour and center of the ball are determined, see Figure 2.3. Thus the x and y position of the ball is obtained in time. This method works quite well, however, sometimes shadows on the wall were detected which resulted in outliers. Some results are given in Figure 2.2.



(a) Example 1.

(b) Example 2, a lot of outliers are present.

Figure 2.2: Two examples of the waveform of theta obtained using tracking of the ball with computer vision.

The parameters and initial conditions of the experiments are given in Table 2.1.

Table 2.1: Parameters and initial conditions of the experimental dataset.

Tennis Ball	
Mass	58g
Length	30cm, 60cm and 100cm
Initial angle	15°, 30° and 45°
Damping coefficient	To be determined
Initial velocity	0 rad/s
Samples	3

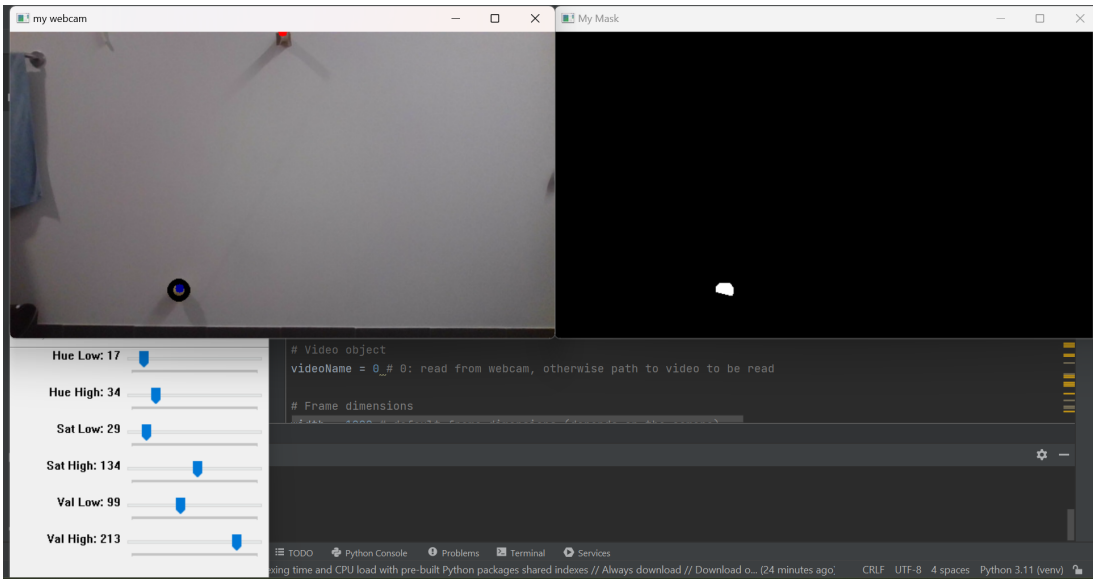


Figure 2.3: Tracking of the pendulum using computer vision, the contour, and center of the ball are estimated based on the shape of the ball in the mask.

2.2 Virtual dataset

To acquire the virtual data set, we require knowledge of the equations that describe the system. A physical understanding of the problem and knowing the influence of the different parameters are also very useful for understanding the results of the data-driven methods. Assume a mass m hanging from a massless string with length L. The forces acting on the mass are gravity $m\vec{g}$, the tension in the string $\vec{T}$, and viscous damping $c\vec{v_t}$.

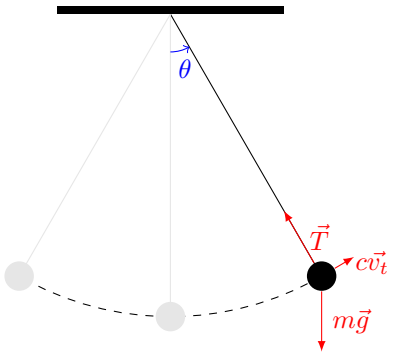


Figure 2.4: Free body diagram of a single pendulum with viscous friction, adapted from [1].

The equation of motion is the following:

$$J\ddot{\theta} = -mgLsin(\theta) - cv_tL \tag{2.1}$$

The inertia is calculated using the parallel axis theorem considering that the inertia of a point is equal to zero.

$$J = J_{point} + mL^2 = 0 + mL^2 \tag{2.2}$$

The tangential velocity v_t can be written as:

$$v_t = L\dot{\theta} \tag{2.3}$$

Thus the differential equation can be rewritten as:

$$\ddot{\theta} + \frac{c}{m}\dot{\theta} + \frac{g}{L}sin(\theta) = 0 \tag{2.4}$$

The system is described by six parameters:

- L the length of the pendulum.
- m the mass of the pendulum.
- c the viscous damping coefficient.
- θ_0 the initial angle (related to the boundary conditions).
- ω_0 the initial angular velocity (related to the boundary conditions).
- g the gravitational acceleration.

By solving this differential equation, which can easily be done numerically using MATLAB or Python, for different conditions, a large and diverse data set is obtained capturing the diverse characteristics of the pendulum's motion.

The problem as described in Equation (2.4) is a nonlinear. If we make a small angle approximation $sin(\theta) \approx \theta$ the equation becomes linear and is equal to a damped harmonic oscillator. If the system is underdamped, the analytic solution to the problem is given by Equation (2.5).

$$\theta(t) = A_0 e^{\frac{-c}{2m}t} cos(\omega t + \phi) \tag{2.5}$$

Where the amplitude A_0 and the phase angle ϕ depend on the initial conditions. The pulsation is equal to $\omega = \frac{2\pi}{T} = \sqrt{\frac{g}{L} - (\frac{c}{2m})^2}$.

2.3 Outlier Detection

There are different ways in which noises and disturbances can impact a system, ranging from system identification to the establishment of a control strategy for said system. The utilization of sensory devices constitutes one of the sources that may introduce outliers into the acquired data. In more general terms, outliers can arise due to various factors, such as measurement errors, unexpected events, or anomalies in the system. In our study, computer vision techniques were employed to generate our data set, resulting in the presence of outlier data. In order to assess the algorithm's performance, we also constructed a virtual data set for comparative analysis, aiming to evaluate the accuracy of outlier detection in both the original and virtual data sets. For this task, we used some of the libraries of *sklearn*. Figure 2.5 shows different rope lengths' actual and virtual data sets.

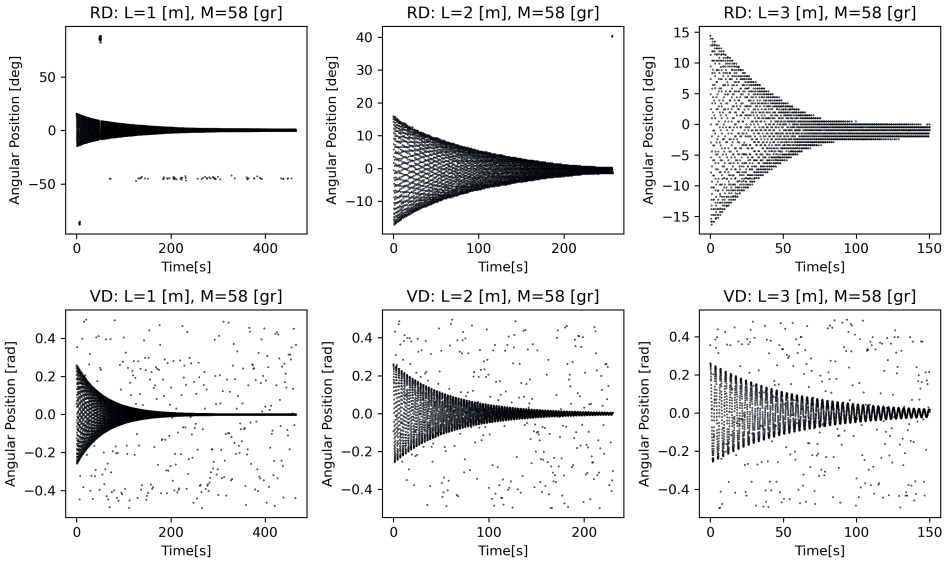


Figure 2.5: Actual (RD) and virtual (VD) datasets for outlier detection.

In this study, we compared both the *Mahalanobis* and *Euclidean* distance finding method for the angular position in order to determine the unwanted data points. The cumulative distances have been shown in Figure 2.6.

The diagram depicted in Figure 2.6 reveals the possibility of identifying specific data points as anomalies. This outcome is expected due to the extensive time span involved. As time progresses, the envelope becomes narrower, causing the central point of the data sets to gradually shift towards the right. Consequently, the distance from the central point might exceed a predetermined threshold and result in the removal of certain data points.

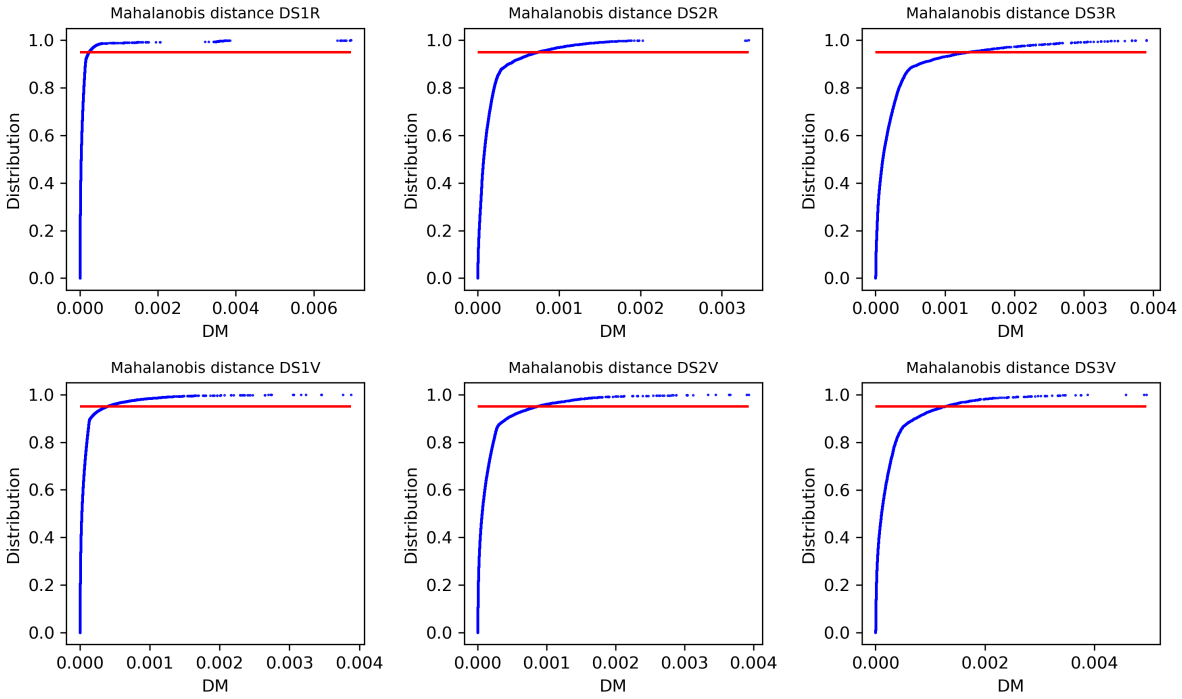


Figure 2.6: DM Distribution

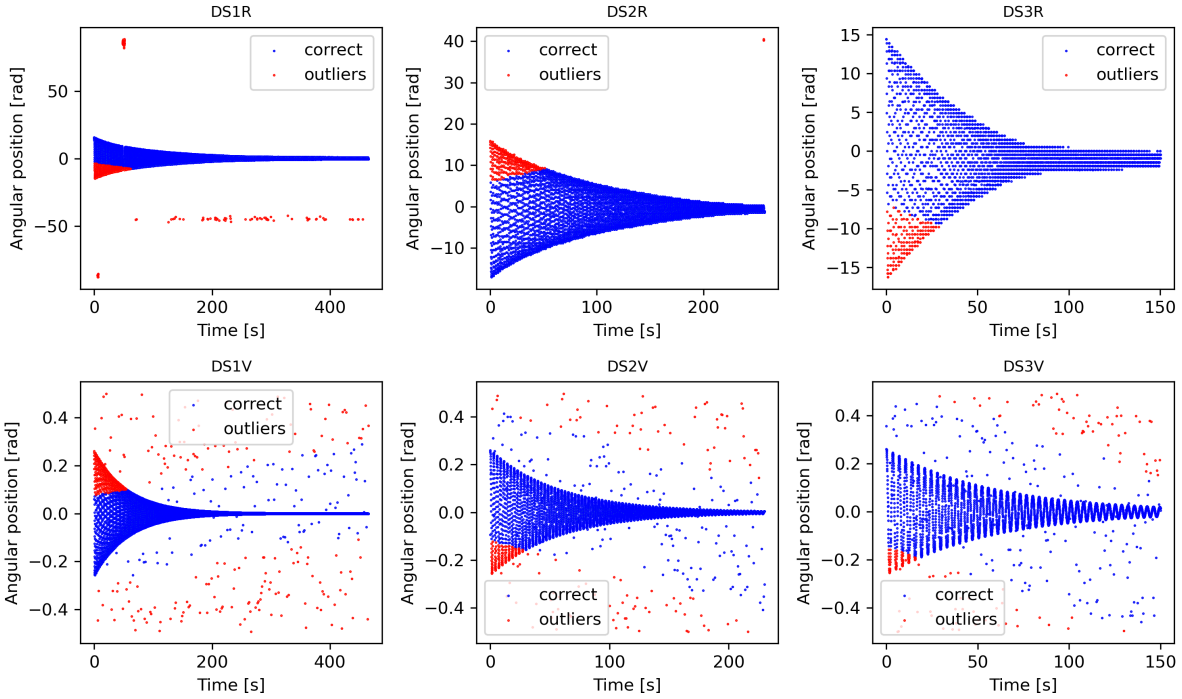


Figure 2.7: Detected Outlier with Mahalanobis Distance

As anticipated, Figure 2.7 illustrates that during the initial phase of the envelope, where the distance from the center point is significant, certain data points have been classified as outliers. Conversely, the simulated data sets demonstrate that the detection of outliers becomes more challenging when they are distributed across a wider range.

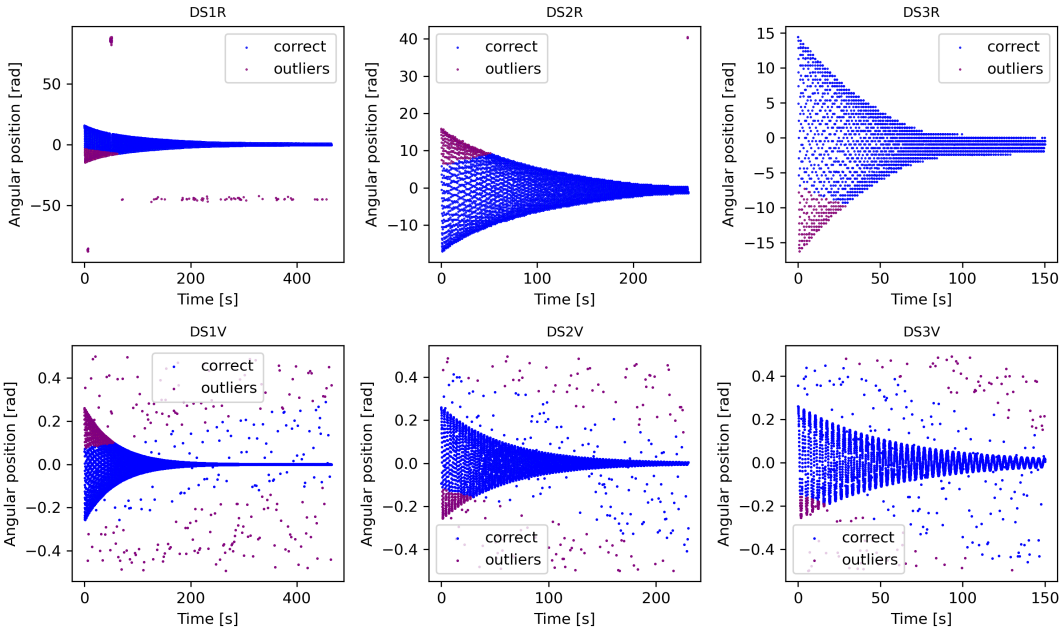


Figure 2.8: Detected Outlier with Euclidean Distance

The outcome of outlier detection utilizing the *Euclidean* method is displayed in Figure 2.8. The results from both methods are nearly identical. Hence, it can be inferred that a larger number of data points does not necessarily indicate a better outcome. For instance, in the Actual data set with a rope length of 1 $[m]$, the mean value approaches zero after approximately 200 seconds. Consequently, by truncating the data set from 200$[s]$ onwards, better detection of undesired data points can be achieved. This has been shown in Figure 2.9

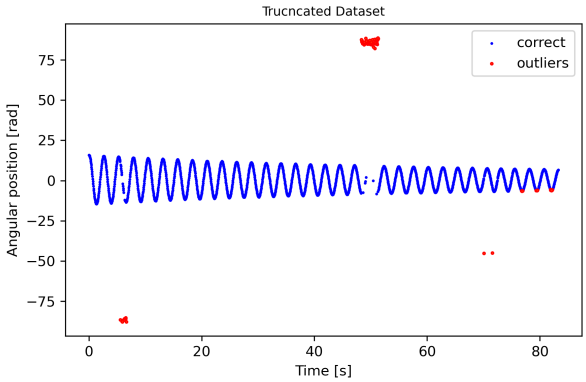


Figure 2.9: Outlier Detection on the Truncated Dataset

3 | PCA for dimensionality reduction

3.1 Observed features

The data set on which PCA will be performed is given in Table 3.1. Only the first three samples are shown in this table, while the data set contains 300 samples in the 12-dimensional space of the original features.

Table 3.1: The first three samples of the data set with 12 features and 300 samples on which PCA is performed.

c/m	g/l	θ_0	ω_0	H_1	H_2	H_3	T_1	T_2	δ	T	N
0.662	6.989	0.905	0.022	0.394	0.177	0.078	2.410	2.390	0.800	7.280	2
2.596	22.04	1.047	-0.002	0.160	0.026	0.004	1.400	1.390	1.810	2.840	1
0.482	9.204	0.970	0.026	0.566	0.339	0.204	2.110	2.080	0.513	10.52	4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

All the quantities listed in the table are expressed in SI units. They correspond to the following units: $\frac{m^2}{s}$, $\frac{1}{s^2}$, rad, $\frac{rad}{s}$, m, m, m, s, s, $dimensionless$, s, $dimensionless$.

This virtual data set is generated by sampling the parameters related to the first four features in the data set from uniform distributions or Gaussian distributions. These distributions are given in Table 3.2. Uniform distributions are denoted [a, b] (where a is the lower limit and b is the upper limit of the distribution) and the Gaussian distributions are denoted as $\mathbb{N}(\mu, \sigma)$.

Parameter	Unit	Value	Parameter	Unit	Value
l	m	[0.5, 1]	θ_0	deg	[45, 67.5]
m	kg	[0.057, 0.285]	ω_0	rad/s	$\mathbb{N}(0, 0.02)$
c	kg/s	[0.05, 0.15]	g	m/s^2	[2.943, 32.373]

Table 3.2: Parameters used for creating the data set for performing PCA.

Now for this dataset, certain features of the data itself will be studied and by identifying the main features and extracting them by using PCA, dimensionality reduction can be achieved. The features that will be studied are: the height of the first, second and third peak (H1, H2 and H3), the time between two consequent peaks of those three peaks (T1 and T2), the logarithmic decrement (δ) of the peak height and the time and number of oscillations it takes for the peaks to reach a certain limit (T and N). This limit is chosen to be $\theta_{\text{limit}} = 5°$.

The logarithmic decrement δ is defined as the natural logarithm of the ratio of any two consecutive amplitudes of an oscillating system:

$$\delta = \ln\left(\frac{A_n}{A_{n+1}}\right) \tag{3.1}$$

Where A_n and A_{n+1} are the amplitudes of the oscillation at two consecutive peaks. The logarithmic decrement provides information about the damping characteristics of the system. It quantifies the rate at which the amplitude of the oscillations decreases over time. A higher logarithmic decrement indicates a stronger damping effect, meaning the oscillations decay more rapidly. Conversely, a lower logarithmic decrement corresponds to weaker damping and slower decay of the oscillations.

3.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely employed unsupervised learning technique in data analysis, particularly for large multidimensional data sets with numerous features. Its primary purpose is dimensionality reduction and feature extraction by transforming the correlated features into a set of uncorrelated features.

The heatmap of the correlation matrix of the features is given in Figure 3.1. Based on our intuition and understanding of the physics underlying the pendulum motion problem, it is evident that the features in the data set exhibit a strong correlation. Specifically, the motion of the pendulum can be fully determined by the first four features of Figure 3.1. As a result, all the remaining features in the data set are also entirely dependent on these initial four features. Furthermore, since the first four features can be chosen independently

from each other, they are uncorrelated with one another.

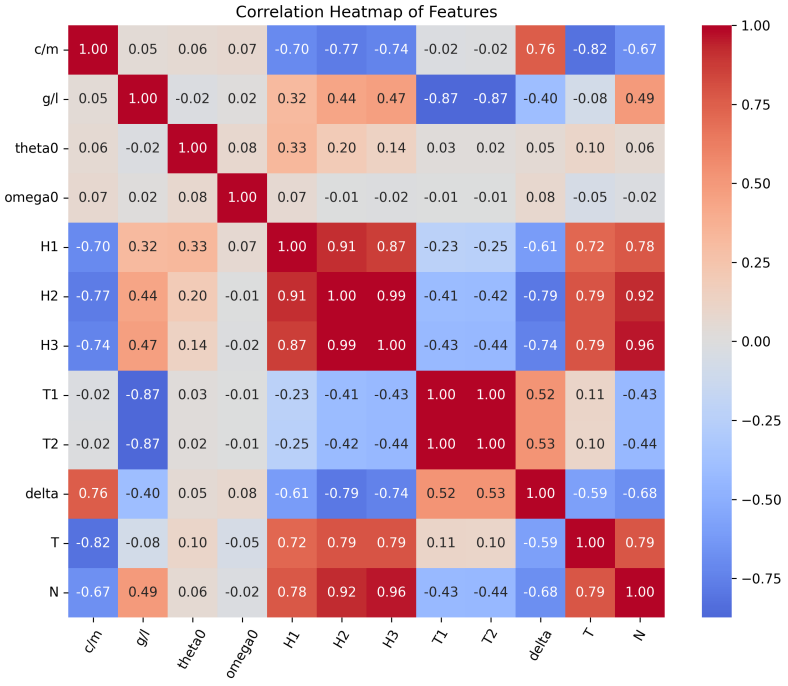


Figure 3.1: A heatmap representation of the correlation matrix of the features.

From this heatmap, we can observe several clear linear dependencies between features. One notable example is the positive correlation between $\frac{c}{m}$ and δ. This correlation aligns with our expectations when studying the linearized problem of eq. (2.4) around $[\theta, \omega] = [0, 0]$. In this case, the system's behavior is described by an underdamped harmonic oscillator and the relationship between $\frac{c}{m}$ and δ is given by:

$$\frac{c}{m} = \frac{2 \cdot \omega_n \cdot \delta}{\sqrt{\delta^2 + (2\pi)^2}} \tag{3.2}$$

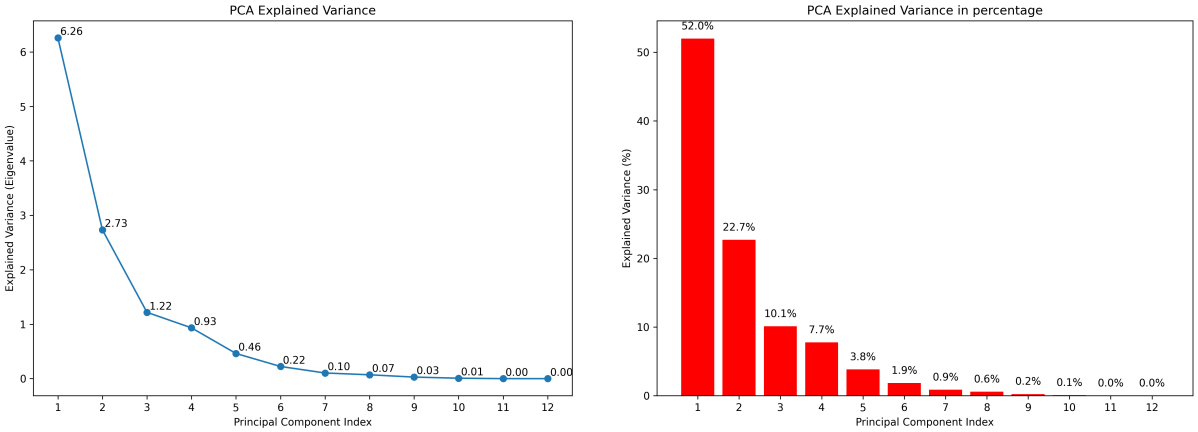
where ω_n is defined as:

$$\omega_n = \frac{2\pi}{T} = \sqrt{\frac{g}{l}} \tag{3.3}$$

The relatively small value of δ compared to 2π explains the positive correlation between $\frac{c}{m}$ and δ (Equation (3.2)). Additionally, Equation (3.3) sheds light on the positive correlation between T_1 and T_2, as they are simply equal to each other, as well as their strong negative correlation with $\sqrt{\frac{g}{l}}$. Similarly, other relationships between the features in the linearized case can help us understand the various correlations depicted in the heatmap.

Because the motion of the pendulum can be fully determined by the first four features, we anticipate that this particular data set will have at least four important principal com-

ponents, referring to the components that account for a significant amount of explained variance. This expectation is supported by the performed PCA analysis, where the explained variance and cumulative sum of variance with respect to the principal components are illustrated respectively in Figure 3.2 and Figure 3.3.



(a) The explained variance is given as the eigenvalue of the covariance matrix in the function of the sorted eigenvectors (principal components) of this matrix.

(b) The explained variance for the sorted principal components is given as the percentage of the total variance of the data set.

Figure 3.2: The explained variance as a function of the index of the principal components.

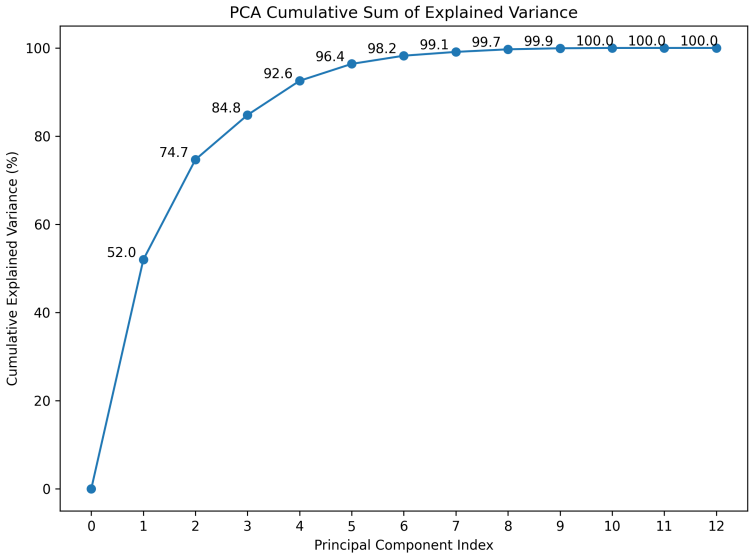


Figure 3.3: The cumulative explained variance for the sorted principal components is given as a percentage of the total variance of the data set.

3.3 Dimensionality reduction

Dimensionality reduction is interesting when the original features exhibit strong correlations and not all the original features are essential for explaining the variance of the

data set. Therefore, it makes more sense to perform PCA on the data set in Table 3.1 discarding three of the first four original features as it is known upfront that these first four features are already uncorrelated.

The first step in the PCA process is to $z-$score the original features stored in X, ensuring that each feature (column of X) has zero mean and unit standard deviation.

$$Z_{\text{scored}} = \frac{X - \mu}{\sigma} \tag{3.4}$$

Next, we construct the covariance matrix S, which is equal to the correlation matrix when the data is z-scored. The correlation matrix is given in Figure 3.1.

$$S = \frac{1}{n-1} \cdot (Z_{\text{scored}}^{\intercal} \cdot Z_{\text{scored}}) \tag{3.5}$$

We then perform an eigendecomposition of the covariance matrix, resulting in eigenvalues and eigenvectors. The eigenvalues are sorted in decreasing order, representing the decreasing variance in the data. Each eigenvalue corresponds to the amount of variance explained by its corresponding eigenvector.

$$S = V \Lambda V^{\intercal} \tag{3.6}$$

Finally, we obtain the projection or transformation of the original normalized data onto the PCA space by multiplying the originally normalized data by the eigenvectors of the covariance matrix, which are the principal components.

$$Z_{\text{PCA}} = Z_{\text{scored}} \cdot V \tag{3.7}$$

Using this transformation, the Figures 3.2 and 3.3 are obtained. From these figures, we see that most of the variance can be explained by the first few principal components and that the remaining ones contribute less significantly. By setting a threshold of preserving at least 95% of the information, we see that in the PCA space 96.4% of the variance can be explained with only 5 dimensions instead of the original 12. Thus effective dimensionality reduction is achieved by transforming the data in the lower-dimensional reduced PCA space spanned by the first few principal components that sufficiently account for the variance in the data. In our case we would take the first 5 principal components.

$$Z_{\text{reduced PCA}} = Z_{\text{scored}} \cdot V_{\text{reduced}} \tag{3.8}$$

4 | Linear regression for pendulum motion prediction

Linear regression provides a clear and intuitive understanding of how changes in independent variables influence a dependent variable. The coefficients of the linear regression model represent the direction and magnitude of these effects. This interpretability makes linear regression a valuable tool for gaining insights and making informed decisions. In addition, linear regression allows us to make predictions based on observed data. Once the model is trained, it can be used to estimate the values of the dependent variable for new data points. This predictive capability is valuable in various applications and will be used here to estimate the future position of the pendulum $\theta(t_{\text{i+1}})$ knowing the current position $\theta(t_{\text{i}})$ and the 4 previous positions $\theta(t_{\text{i-1}})$, $\theta(t_{\text{i-2}})$, $\theta(t_{\text{i-3}})$ and $\theta(t_{\text{i-4}})$. This time the data set was obtained by numerically solving Equation (2.4) using the values in table Table 4.1 below.

Parameter	**Unit**	**Value**	**Parameter**	**Unit**	**Value**
g	m/s^2	9.81	l	m	0.5
m	kg	0.2	c	$kg\ m^2/s$	0.05
θ_0	rad	$\frac{45\pi}{180}$	ω_0	rad/s	0

Table 4.1: Parameters used for numerically solving the differential equation 2.4.

The differential equation was solved with a timestep of 0,01s and for the range 0s to 50s. Furthermore, some Gaussian noise was added to the signal with μ equal to 0 and σ equal to 0,02. Doing so we obtain the following signal for $\theta(t)$:

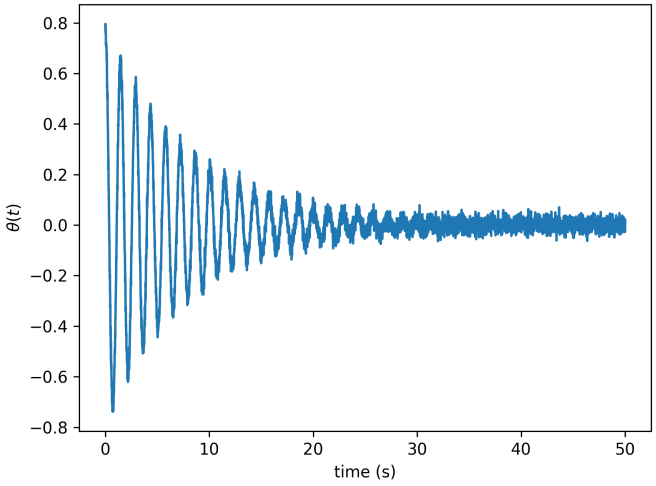


Figure 4.1: The obtained solution for differential equation 2.4 with the parameters from table 4.1 and with added noise with $\mu = 0$ and $\sigma = 0{,}02$.

From this data set 80% of the data will be used to train the regression model and the other 20% will be used to test the model. The linear regression model aims to find a linear relationship between a dependent variable (target variable) and one or more independent variables (predictor variables):

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{p} \mathbf{x}_j \beta_j \tag{4.1}$$

By fitting a line or hyperplane to the data, linear regression seeks to capture the underlying trend and quantify the relationship between the variables.

4.1 Ordinary Least Squares

A first linear regression model is obtained by ordinary least squares (OLS) estimation. The model is built in such a way as to minimize the sum of squared differences between the observed and predicted values:

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{4.2}$$

This optimization approach provides reliable and efficient estimates of the model coefficients with low bias but in general, the estimates have large variance. The linear regression using the least squares cost function 4.2 to minimize is shown in fig. 4.2.

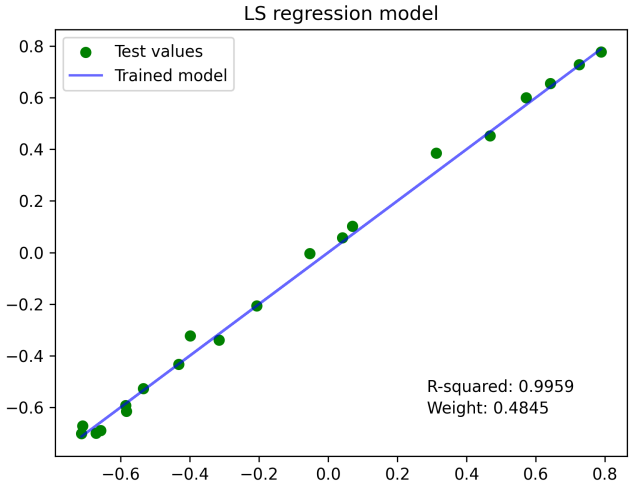


Figure 4.2: Ordinary least squares linear regression model which is tested with the green data points.

Least squares linear regression aims to minimize the sum of the squared differences between the predicted values and the actual values in the training data. This approach generally leads to estimates with low bias because it tries to fit the data as closely as possible.

However, when the model is complex or overfits the training data, it can result in estimates with large variances. Overfitting occurs when the model captures noise or random fluctuations in the training data instead of the underlying patterns. This can lead to high sensitivity to changes in the training set, causing the model to have large variations in its predictions.

To mitigate the issue of high variance, various techniques can be employed, such as regularization (e.g., Lasso and Ridge regression) or model selection methods (e.g., cross-validation). These methods aim to find a balance between low bias and low variance, resulting in more robust and generalizable models.

4.2 Lasso and Ridge regression with cross-validation

In machine learning, overfitting occurs when a model learns the training data too well, capturing noise or irrelevant patterns in the data, which leads to poor performance on unseen data. Regularization helps to address this issue by adding a regularization term to the objective function, which modifies the model's behavior during training. So to reduce the variance of the estimates, regularization terms are introduced to the objective function 4.4. For Lasso regression the objective function becomes:

$$\hat{\beta} = \arg\min \left\{ \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda_1 \sum_{j=1}^{p} \|\beta_j\| \right\} \tag{4.3}$$

and for Ridge regression the objective function becomes:

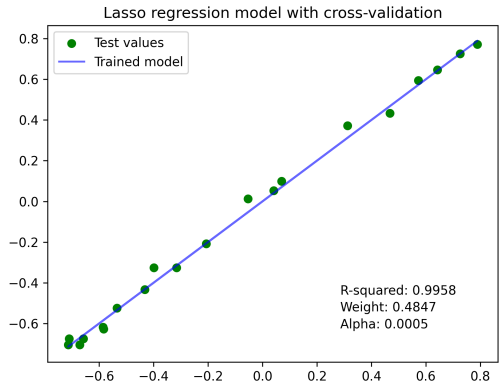
$$\hat{\beta} = \arg\min \left\{ \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\} \tag{4.4}$$

The regularization terms impose a penalty for the amount of regression parameters and their size. The regularization terms control the trade-off between fitting the training data well and keeping the model's parameters small. By adjusting the regularization strength, the model can be tuned to find the right balance between bias and variance. In this way, overfitting is discouraged and the generalization ability of the predictive model is improved.

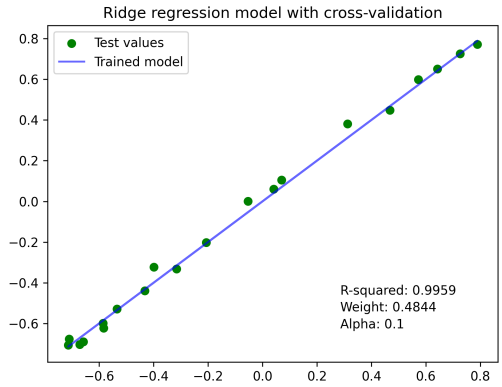
In addition, also cross-validation is introduced to the model. Cross-validation involves partitioning the available data into multiple subsets or folds, where each fold is used as both a training set and a validation set, in order to assess the performance and generalization ability of a predictive model.

The basic idea behind cross-validation is to simulate the performance of a model on unseen data by evaluating it on different subsets of the available data. This helps to estimate how well the model will generalize to new, unseen data. The most common form of cross-validation is called k-fold cross-validation, where the data is divided into k equal-sized folds or subsets. We will choose 5-fold cross-validation.

The linear regression models using Lasso and Ridge regression with cross-validation are shown in Figure 4.3.



(a) Linear regression using least squares objective function with additional L1 constraint (Lasso).



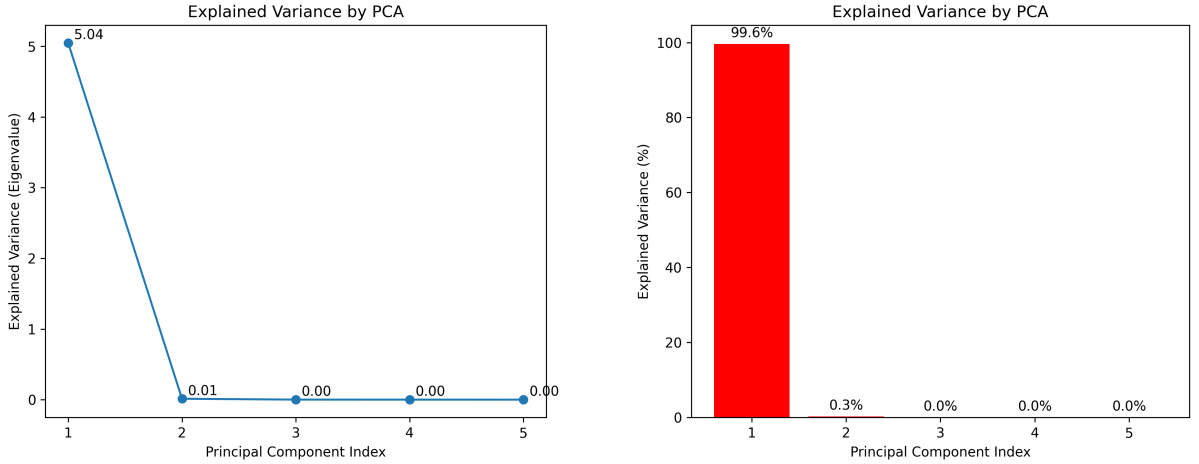
(b) Linear regression using least squares objective function with additional L2 constraint (Ridge).

Figure 4.3: Lasso and Ridge regression with cross-validation.

Using cross-validation, alpha is determined to have an optimal balance between bias and variance of the estimator. To achieve this a lower regularization coefficient is necessary for Lasso than for Ridge indicating the more aggressive penalization nature of Lasso compared to Ridge.

4.3 Linear regression using PCA

Linear regression can be combined with PCA to form the so-called Principal Component Regression (PCR). The idea behind PCR is to reduce the dimensionality of the predictor variables using PCA and then perform linear regression on the principal components instead of the original predictors. This approach allows for capturing the essential information from the predictors while addressing issues such as multicollinearity and high dimensionality.



(a) Explained variance by principal components given as an eigenvalue.

(b) Explained variance by principal components given as a percentage.

Figure 4.4: Principal component regression.

The PCR model was trained and tested with the same 80% and 20% of the dataset. The result is depicted in fig. 4.5.

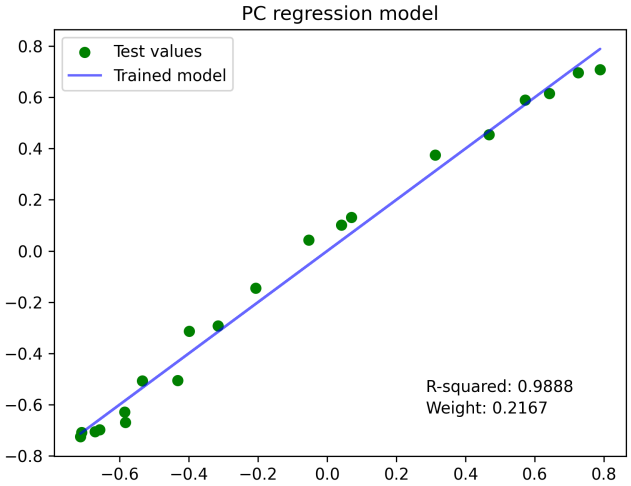


Figure 4.5: PCR model which is tested with the green data points.

PCR demonstrates an interesting characteristic compared to other regression models such as LS, Lasso with cross-validation, and Ridge with cross-validation. It effectively reduces the collective weight or magnitude of the regression parameters to a significantly lower value. Despite this reduction, the R-squared value, which represents the goodness of fit, remains very close to 1. This implies that PCR is able to capture the essential information and patterns in the data while using a reduced set of predictors. In other words, it achieves a parsimonious representation of the relationship between the predictors and the response variable, resulting in a more concise model without sacrificing much predictive power. Figure 4.6 illustrates the regression parameter values related to the predictor variables or principal components, arranged in descending order, for the different regression models.

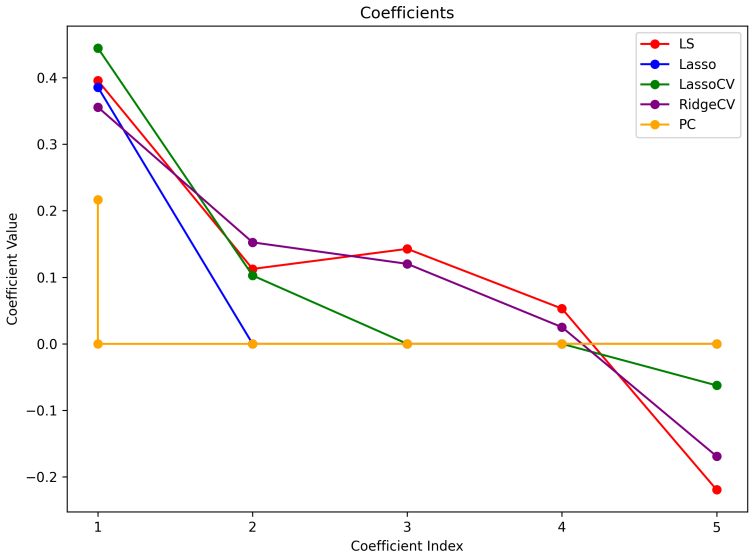


Figure 4.6: Values of the regression parameters of all the linear regression models used on the data set.

5 | Modal analysis and proper orthogonal decomposition

Modal analysis is another possible way to analyze data. The main idea of Modal analysis is to decompose a data set as a linear combination of its modes. Those modes are a linear combination of elementary contributions [2]. Principally, it is also linked with PCA, which has been performed in Chapter 3. The main difference between PCA and modal analysis or rather proper orthogonal decomposition (POD), lies in the approach. POD a via singular value decomposition (SVD) is considered in the literature as a method of POD as well as PCA and are also seen as equivalent to each other [3]. The relevant difference for this report lies in the implementation. While PCA focuses on the principal components, POD via SVD focuses on the spatial and temporal distribution of the data.

The approach of the POD, conforming to [4] is the following: The data is described as a linear combination of the elementary information. The underlying data of the POD can be split into amplitude, spatial structure, and temporal structure.

$$\vec{u_r}(x_i, t_k) = \Sigma_{r=1}^{R} \vec{u_r}(x_i, t_k) = \Sigma_{r=1}^{R} \sigma_r \phi_r(x_i) \psi_r(t_k) \tag{5.1}$$

This has been done with the vector field $\vec{u}$ in Equation (5.1), whereby σ_r represents the amplitudes, ϕ_r represents the spatial bases and ψ_r represents the temporal bases.

To perform the modal analysis on our data, we first have to create snapshot matrices from our data. Generally, snapshot matrices represent the dynamical system. They are created with the following scheme shown in Equation (5.2) [4].

$$D(x,t) = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ d_1(x) & d_2(x) & \cdots & d_n(x) \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \tag{5.2}$$

Where the rows represent time and the columns represent space.
Based on the creation of the snapshot matrix, it is possible to process further with the

data in order to obtain the modes. It should be noted in advance that, for our data set, Equation (5.3) applies.

$$n_s \ll n_t \tag{5.3}$$

Consequently, the "Classical POD", as it is called in the lecture notes of professor *M.Mendez* [4] can be applied to compute the SVD. The algorithm is indicated in Algorithm 1 and split into four general steps.

Algorithm 1 Classical POD

1: $C = DD^T$
2: $C = \Phi_P \Lambda_P \Phi_P^T$
3: $\Sigma = \sqrt{\Lambda_P}$
4: $\Phi_P = D\Psi\Sigma^{-1}$

Those steps can be realized using three different approaches [4]. For the modal analysis, which is performed in this report, the approach via SVD was selected. Generally, there is no significant difference in the result of those approaches. After applying Algorithm 1, the spatial and temporal bases ϕ_r and ψ_r are obtained and can be interpreted.

5.1 Application on the pendulum data set

In order to be able to perform the POD, it is necessary to transform the data set into x and y coordinates, as expressed in angle θ, the snapshot matrix would be one dimensional. The equations to calculate x and y are trivial and given in Equation (5.4) and Equation (5.5) using the length of the pendulum L.

$$x = Lsin(\theta) \tag{5.4}$$

$$y = Lcos(\theta) \tag{5.5}$$

Analogous to that, the velocities, $\dot{x}$ and $\dot{y}$ can be calculated, using the angular velocity $\dot{\theta}$ as indicated in Equation (5.6) respectively Equation (5.7).

$$\dot{x} = Lcos(\theta) \tag{5.6}$$

$$\dot{y} = -Lsin(\theta) \tag{5.7}$$

After gaining these parameters, the snapshot matrix $D(x,t)$ can be developed. Depending on whether the positions x and y or the velocities $\dot{x}$ and $\dot{y}$ are observed, the snapshot matrix is obtained as $D_{pos}(x,t)$.

$$D_{pos}(x,t) = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{pmatrix} \tag{5.8}$$

The Derivation of the snapshot matrix of the velocity $D_{vel}(x,t)$ is obtained in a similar manner, which is described in Equation (5.9).

$$D_{vel}(x,t) = \begin{pmatrix} \dot{x}_1 & \dot{x}_2 & \cdots & \dot{x}_n \\ \dot{y}_1 & \dot{y}_2 & \cdots & \dot{y}_n \end{pmatrix} \tag{5.9}$$

The length of the columns is in both cases $n_\mathrm{s} = n_\mathrm{x} \cdot n_\mathrm{y} = 2$ because $n_\mathrm{x} = n_\mathrm{y} = 1$. The length of the rows of these matrices depends on the number of time samples $n_\mathrm{t} = 800$ which is the temporal length of the experiment, as 20 seconds of the pendulum motion will be taken into account. This leads back to Equation (5.3), where n_x and n_y were anticipated.

Normally, the POD is performed on huge data sets with a high number of dimensions and modes. The goal of this is to find the dominant modes and reduce the model order for example. However, this does not apply to the investigated data set. Because $n_s = 2$, the maximum number of modes that can be distinguished is two. Another point that has to be considered is, that the value of x and y respective $\dot{x}$ and $\dot{y}$ are not independent from each other. They are linked via θ, which can be observed via Equation (5.4), Equation (5.5), Equation (5.6) and Equation (5.7).

5.2 Preparation of the POD

Evidently, it is not constructive to perform a POD on every possible permutation of the in Chapter 3 mentioned parameters, which influence the motion of the pendulum, such as c, g, L, and m. Instead, a certain selection of those parameters has been done in advance in order to reduce the range of the analysis.

As the POD is performed on $x, y, \dot{x}$ and $\dot{y}$, it is of interest to first consider the distribution of those values over time. The x- and y-positions of the pendulum over time are shown in Figure 5.1 and Figure 5.2. Analogous to that, the velocities are shown over time in Figure 5.3 and Figure 5.4.

It should be noted that in the studied case of the pendulum movement, the spatial bases of the data have less relevance for the interpretation than in a case of a studied vector field, which describes for example a turbulent fluid, as there are more spatial bases than two. Therefore the spatial bases Φ will be omitted in the following analysis.

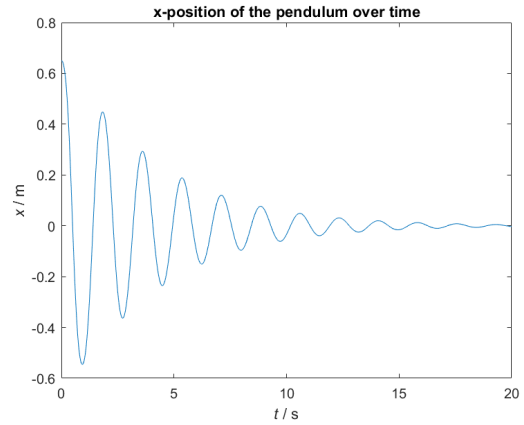


Figure 5.1: Variation of the x-position of the pendulum over time.

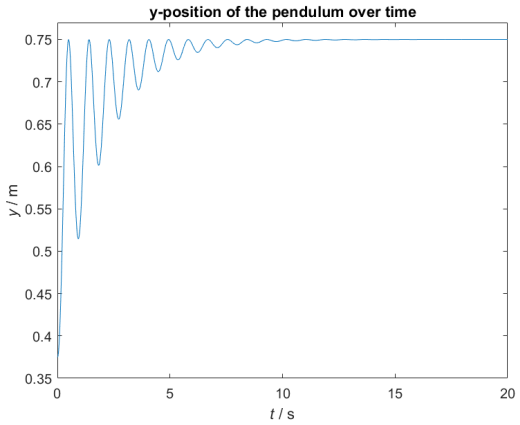


Figure 5.2: Variation of the y-position of the pendulum over time.

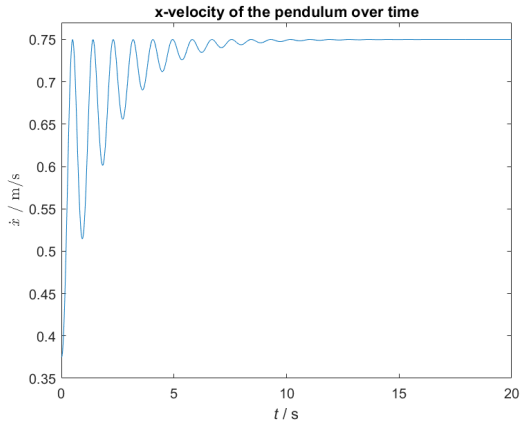


Figure 5.3: Variation of the x-velocity of the pendulum over time.

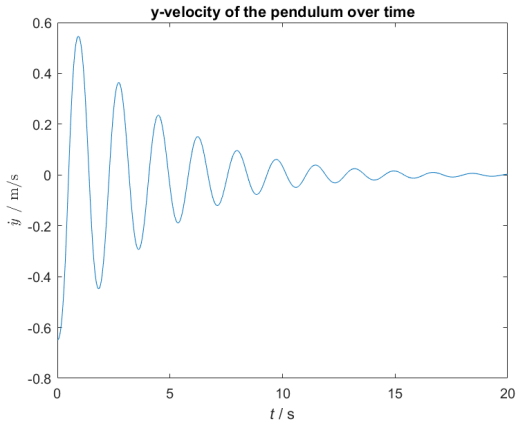


Figure 5.4: Variation of the x-velocity of the pendulum over time.

5.3 POD applied on the pendulum position

First, the distribution of σ_{P} of the positional data is investigated. For the analysis of the positions, σ_{P} is shown in Figure 5.5.

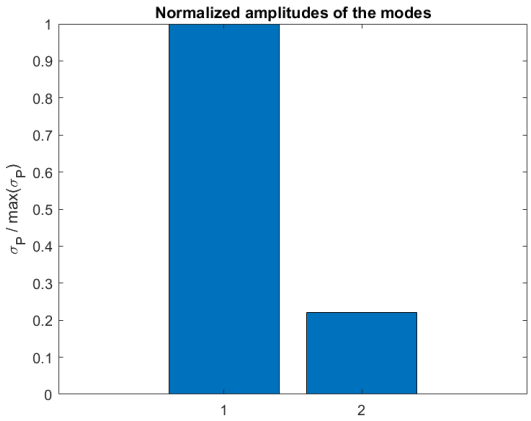


Figure 5.5: Amplitudes σ_P of the modes.

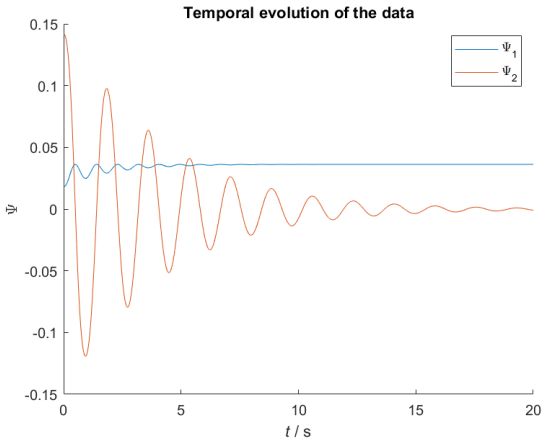


Figure 5.6: Temporal distribution $\Psi_\mathrm{P}(t)$ of the position data.

It is evident, that the first mode plays an important role when it comes to modeling the system behaviour, as the amplitude is significantly higher than the amplitude of the second mode. The amplitude of the second mode is only 20 % of the amplitude of the first mode. In this case it means, that the first mode contributes more to the energy of the system. This means that reconstruction only consisting of the first mode would cover the energy of the system better than a reconstruction with only the second mode.

Next, the distribution of Ψ_P over the time t is indicted in Figure 5.6. A comparison of Ψ_1 from Figure 5.6 with y from Figure 5.2 as well as Ψ_2 from Figure 5.6 with x form Figure 5.1 already allows to recognize which property is mainly represented by which mode. This is due to the fact that graphs appear to be qualitatively similar. However, the progressions of Ψ_P do not match quantitatively. This can be explained by the decomposition, because Ψ_P is modeled Ψ_P without amplitudes σ_P and spatial bases Φ_P. Consequently, it is to be expected that the first mode $P = 1$ contains more information about the y-position of the pendulum and the second mode $P = 2$ contains more information about the x-position of the pendulum.

After performing the POD and obtaining the decomposed structures σ_P, $\Phi(x_i)_\mathrm{P}$ and $\Psi(t)_\mathrm{P}$, it is also possible to use Equation (5.1) to reconstruct the courses of the graphs. This is done in for the x-position in Figure 5.8 with the first mode $P = 1$ and in **??** with the second mode $P = 2$. To compare the reconstruction $x_\mathrm{rec,P}$ with the real course of x, x is also plotted in the same graph. It is observable that the influence of the first mode on x is insignificant, which could be expected based on the conclusions gained from the comparison of Ψ_P in Figure 5.6. Nevertheless, it should be noted that the contribution of the first mode to x is not completely zero, although, compared to the real course of x it is small. The same applies for the comparison of the reconstruction of y in Figure 5.9

and Figure 5.10.

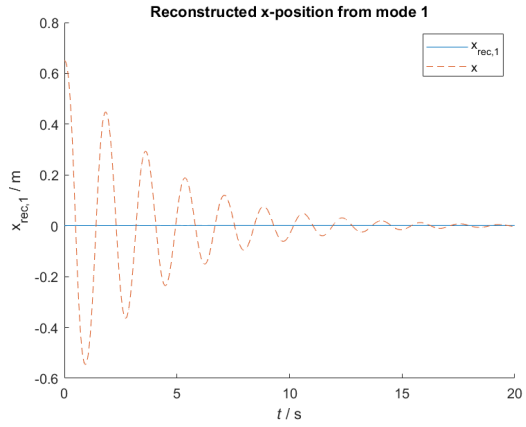


Figure 5.7: Reconstruction of x from the first mode and x.

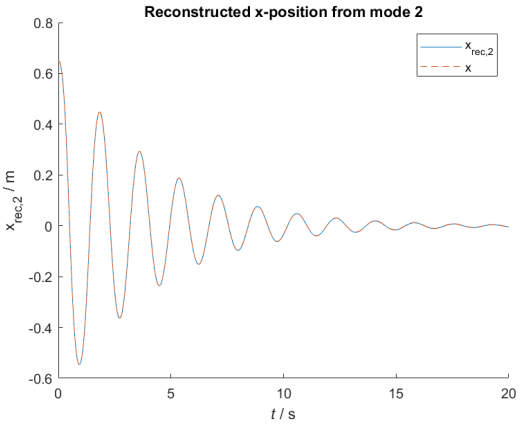


Figure 5.8: Reconstruction of x from the second mode and x.

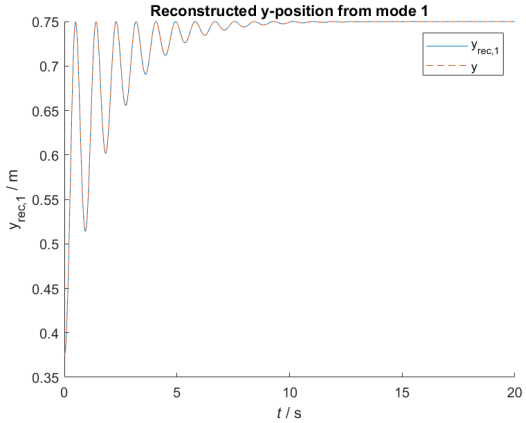


Figure 5.9: Reconstruction of y from the first mode and y.

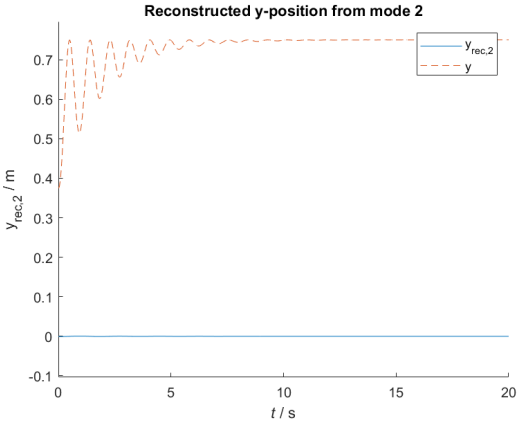


Figure 5.10: Reconstruction of y from the second mode and y.

5.4 POD applied on the pendulum velocity

As a consistent result of the analysis is required, the same case, which was studied in **??** and specified in **??**, are investigated, regarding the velocities $\dot{x}$ and $\dot{y}$.

5.5 Conclusion of the application of the POD on the pendulum

Generally, it can be stated that POD is also applicable on a data set, which describes the two dimensional movement of a pendulum. However, only two modes can be extracted in

position or velocity domain. Those modes describe different features and have an insignificant influence on the features, which they do not mainly describe. The consideration of the amplitudes of the modes σ_{P} provides an insight above the energy distributed on the features. When POD is applied on higher dimensional data sets, as it is usually the case, it is often aimed to reduce the model order by neglecting modes that contribute less to the energy of the system for example. In the studied case, it is not recommended, as a huge amount of information about one position x or y would be lost. The gained kowldege about the energy can be used for ...

6 | Classification and regression using a neural network

Today, neural networks (NNs) are a hot topic. Their popularity has increased tremendously over recent years thanks to an increase in computational power and the availability of large data sets [5]. They are a subset of supervised learning where the algorithm is trained on labeled input data. Once trained, the network can be used for recognizing patterns, classifying data, and making predictions [6]. A NN with sufficient hidden neurons and appropriate activation functions is considered to be a universal function approximation [7]. It can approximate any arbitrary function. However, tuning the NN, for example, choosing the number of hidden layers and neurons per layer, is still challenging. Generally, the more complex the problem, the more layers are required.

In this context, a NN is trained using the virtual data set. It is used to classify the parameters of the problem, see Section 6.1. Additionally, it is used to make a prediction of the trajectory when given the parameters of the problem as input using regression, see Section 6.2. The parameters of the data set are changed according to the task and are specified in each section. The 'Keras' function of the Python package 'TensorFlow' is used to build the model. The data set is first split into 80% for training and 20% for testing and then scaled. The number of layers, neurons per layer and activation functions are chosen arbitrarily and they are tuned by looking at the performance of the NN. The number of epochs is chosen based on the training history, specifically by looking at the validation loss. It is possible that the loss decreases while the validation loss remains the same, this means that the model is overfitting on the training data. The Adam optimizer is used. The loss function is chosen depending on the task.

6.1 Classification of the parameters

The first task of the NN is to classify the parameters based on the evolution of the angle in time. Thus, the input for each sample is $\theta(t)$ where t ranges from 0 to 100s in steps of 0,01s.

6.1.1 Length

First, it was tried to classify the length of each sample. A test is performed on a data set where only the length was varied and all other parameters are kept constant, without noise, see Table 6.1.

Parameter	Unit	Value	Parameter	Unit	Value
L	m	0.1, 0.25, 0.5, 0.75, 1	g	m/s^2	9.81
M	kg	0.1	μ		0
C	kg/s	0.05	σ		0
θ_0	deg	25	Samples		100
ω_0	rad/s	0	Tot. Samples		500

Table 6.1: Parameters used for creating the data set for classifying the length using a NN.

The input is immediately connected to the output using a fully connected neural network (FCNN). The model is given in Figure 6.1. A softmax layer is added later to add probabilities to the model. The 'Sparse Categorical Crossentropy' loss function is used. The labels are mapped as follows:

$$[0.1, 0.25, 0.5, 0.75, 1] \rightarrow [0, 1, 2, 3, 4]$$

```
# Fully-connected neural network with input directly connected to output
output_shape = 5 # 5 possible values for the length
input_shape = (X_train.shape[1],)
model = tf.keras.Sequential([
    tf.keras.layers.Dense(output_shape, input_shape=input_shape, activation='linear')
])

model.summary()
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Figure 6.1: FCNN model for classifying the length. The input is immediately connected to the output using a linear activation function.

The model is trained for 10 epochs, the training history is given in Figure 6.2. After only one epoch, the model is fully trained on the training data and the test accuracy equals 100%. The NN can accurately classify the length for each testing sample using only a linear activation between input and output.

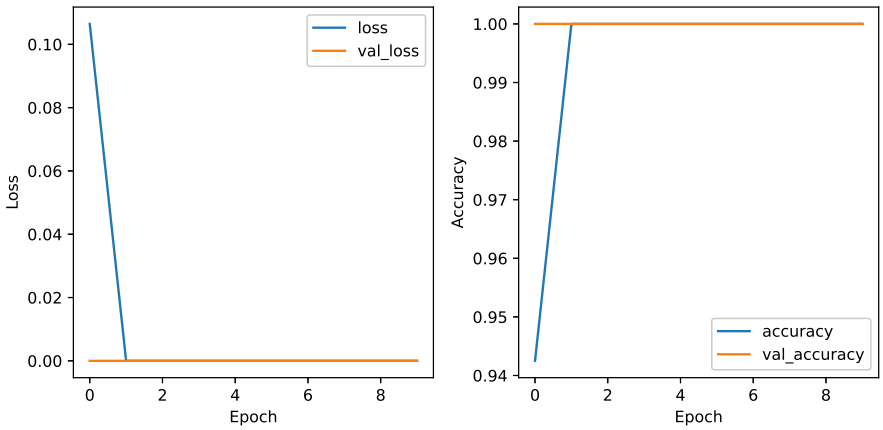


Figure 6.2: Training history of classifying the length for 10 epochs. 100% test accuracy is reached after 1 epoch.

Some results of the classification are given in Figure 6.3.

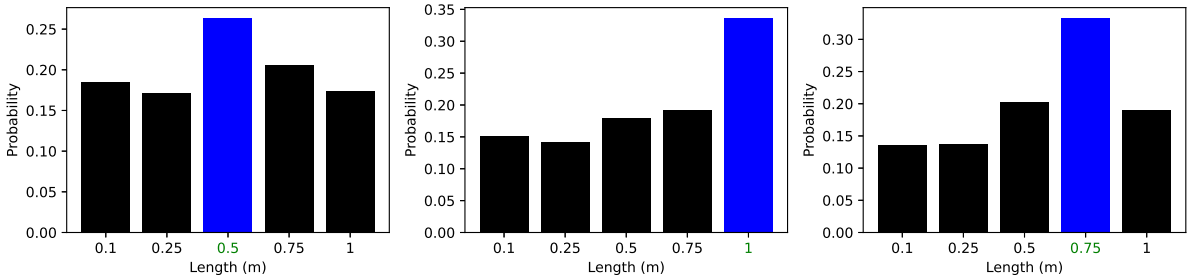


Figure 6.3: Results of classification of the length. The true length is colored in green and the predicted length in blue

6.1.2 All parameters

The classification of the length is a success and was easily realized. The next task for the NN is to classify all of the parameters at once. The parameters used for the creation of the data set are given in Table 6.2. No noise is added as the classification will already be challenging enough.

Parameter	**Unit**	**Value**	**Parameter**	**Unit**	**Value**
L	m	0.1, 0.5, 1	g	m/s^2	9.81
M	kg	0.05, 0.5, 1	μ		0
C	kg/s	0.001, 0.01, 0.05	σ		0
θ_0	deg	25, 50, 75	Samples		25
ω_0	rad/s	0, 0.5, 1	Tot. Samples		6075

Table 6.2: Parameters used for creating the data set for classifying all parameters using a NN.

The amount of neurons in the last hidden layer is chosen to be five. The idea is that one neuron in the last hidden layer corresponds to one of the five parameters that describe the system. This last hidden layer is linearly connected to the output layer. Each output is a certain combination of the input parameters, for five parameters with each 3 possible values, this yields 243 possible combinations. The output is chosen to be the combination with the highest probability. Different models are tried by adding more hidden layers. Eventually, a model with four hidden layers is chosen, see Figure 6.4.

```
# Fully-connected neural network with four hidden layers
output_shape = 3*3*3*3*3 # possible combinations
input_shape = (X_train.shape[1],)
model = tf.keras.Sequential([
    tf.keras.layers.Dense(200,input_shape=input_shape, name='Hidden1',activation = 'relu'), # First hidden Layer
    tf.keras.layers.Dense(128, activation="relu", name="Hidden2"), # Second hidden Layer
    tf.keras.layers.Dense(16, activation="relu", name="Hidden3"), # Third hidden Layer
    tf.keras.layers.Dense(5, activation="relu", name="Hidden4"), # Fourth hidden Layer
    tf.keras.layers.Dense(output_shape, name ='Output',activation='linear')
])

model.summary()
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Figure 6.4: FCNN model for classifying all parameters. There are six layers; the input layer, four hidden layers, and the output layer.

The training history for 50 epochs is shown in Figure 6.5.

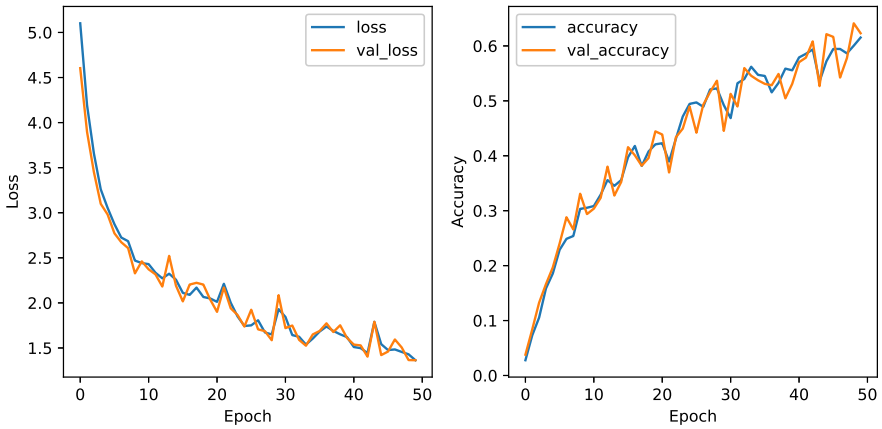


Figure 6.5: Training history of classifying all the parameters for 50 epochs.

It is interesting to make a comparison between the predicted and actual values of the parameters. The results for three random test samples are shown in Figure 6.6. The test accuracy is 28%. The prediction for one in four samples is completely correct. For the other samples, usually, some parameters are guessed correctly, but not all. The test accuracy for each individual parameter is also calculated; the length 81%, the mass 61%, the damping coefficient 57%, the initial angle 69%, and the initial angular velocity 49%.

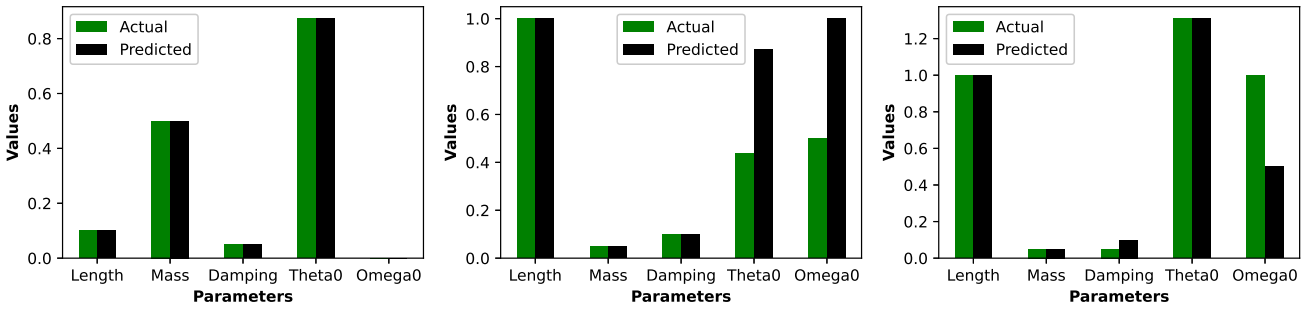


Figure 6.6: Results of classification of all parameters.

6.2 Prediction of the trajectory

Lastly, a NN is trained to predict the trajectory of the pendulum given the parameters of the problem (length, mass, damping coefficient, θ_0 and ω_0) as input. The output is the evolution of $\theta(t)$ in time.

Multiple models are tried by changing the activation functions (relu, tanh, elu), the number of layers, the number of neurons per layer, different optimizers (adam, sgd), etc. However, the model never seemed to be able to train sufficiently, the test accuracy is usually around 5%.

An example of one of the models is given in Figure 6.7.

```
# Fully connected neural network with three hidden layers
output_shape = y_train.shape[1]
input_shape = (X_train.shape[1],)

# Fully-connected neural network
model = tf.keras.Sequential([
    tf.keras.layers.Dense(16,input_shape=input_shape, name='Hidden1',activation = 'elu'), # First hidden Layer
    tf.keras.layers.Dense(64, activation="tanh", name="Hidden2"), # Second hidden Layer
    tf.keras.layers.Dense(128, activation="relu", name="Hidden3"), # Third hidden Layer
    tf.keras.layers.Dense(output_shape, name ='Output',activation='linear')
])

model.summary()
model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
```

Figure 6.7: FCNN model for regression of the trajectory. There are five layers; the input layer, three hidden layers, and the output layer.

The training history is shown in Figure 6.8 The model has a test accuracy of 9,7%.

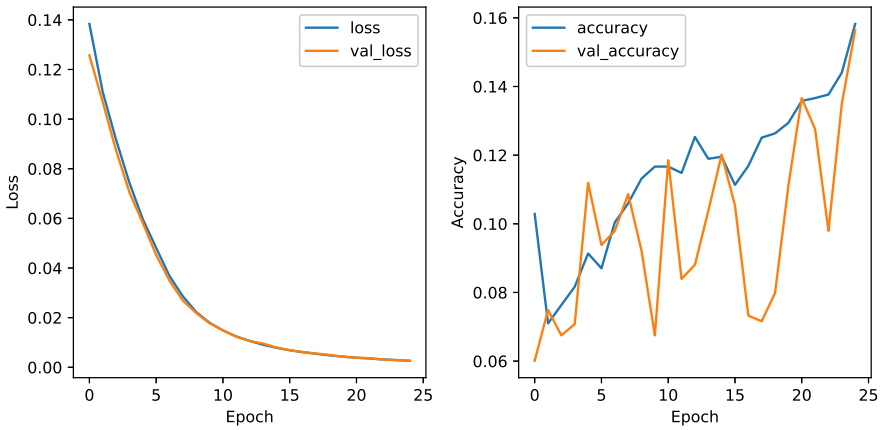


Figure 6.8: Training history of regression of the trajectory for 25 epochs.

The prediction of the trajectory is compared to the actual trajectory for three random samples in Figure 6.9.

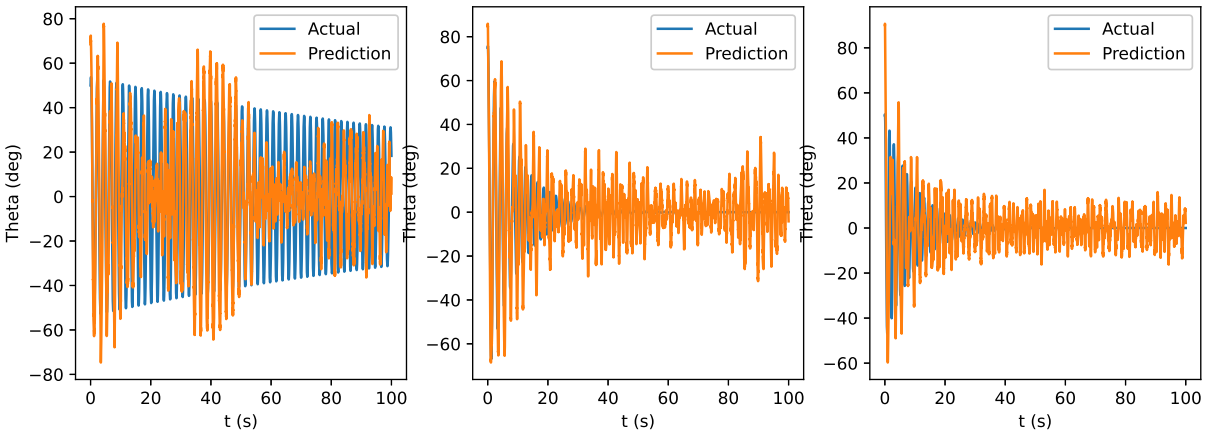


Figure 6.9: Results of regression of the trajectory for three random test samples.

The current model is not performing well. However, the regression could be improved by adding information on the physics of the pendulum in a so-called Physics Informed Neural Network (PINN). This is a specific type of NN that lies in between traditional physical models and data-driven solutions [8]. By combining knowledge of the system and a data-driven supervised NN, a robust model is achieved that is able to reproduce a consistent result. Moreover, it is able to extrapolate accurately beyond the available data set. In this case, the differential equation of the pendulum could be encoded in the loss function [9]. By making the loss function the residual of the differential equation, minimizing the loss function will automatically solve the equation. Making a PINN requires extensive knowledge of the 'PyTorch' library from Python. Due to timing constraints, this is not performed in this report. However, it could be interesting for next year's students to explore this topic.

7 | Conclusion

In conclusion, various data set analysis methods were applied to the classic problem of a single pendulum with viscous friction. Spatiotemporal data of the evolution of the pendulum position in time was obtained using computer vision to track a homemade pendulum. Tracking the ball based on its color was successful and by varying the length and the initial angle multiple instances of the pendulum phenomenon were obtained which could be analyzed. However, the tracking algorithm was not perfect, this resulted in outliers in the data set. These were removed using the Euclidian and Mahalanobis distances.

To be able to evaluate more parameters, simulation was used to create a virtual data set using the differential equation. PCA is performed on this data which allows to reduce the dimensionality of the data set by eliminating the correlation between variables. The original twelve variables can be reduced to five new variables while still retaining 96,4% of the variance. PCA also allows for feature extraction, relating the original variables to the principal components to be able to discard some and only keep the most relevant original variables. Next, linear regression of the trajectory is performed. The least squares, Lasso, Ridge, and principal components methods are compared. POD is applied to the data set to extract the most important modes which gives insight into the energy of the system. Lastly, neural networks are used for the classification of the parameters and trajectory prediction. The parameter classification was largely a success where the network reached a test accuracy of 28% for classifying all parameters at once. The accuracies on the individual parameters are even higher. The neural network failed to correctly predict the trajectory. This could potentially be improved by building a physics-informed neural network.

Bibliography

[1] admin. "How to draw an ugly pendulum in tikz?" (2021), [Online]. Available: https://latexdraw.com/how-to-draw-a-pendulum-in-tikz/ (visited on 16/05/2022).

[2] M. Mendez, "Generalized and multiscale modal analysis", in *Data-Driven Fluid Mechanics: Combining First Principles and Machine Learning*, M. A. Mendez, A. Ianiro, B. R. Noack and S. L. Brunton, Eds. Cambridge University Press, 2023, pp. 153–181. DOI: 10.1017/9781108896214.013.

[3] Y. LIANG, H. Lee, S. LIM, W. Lin, K. LEE and C. WU, "Proper orthogonal decomposition and its applications - part i: Theory", *Journal of Sound and Vibration*, vol. 252, pp. 527–544, May 2002. DOI: 10.1006/jsvi.2001.4041.

[4] M. A. Mendez, *What is data driven modal analysis?*, Von Karman Institut For Fluid Dynamics, 2023.

[5] A. Parente and R. De Freitas, *Neural networks*, Brussels Schoof Of Engineering, 2023.

[6] "Neural networks", IBM. (), [Online]. Available: https://www.ibm.com/topics/neural-networks (visited on 07/05/2023).

[7] Wikipedia contributors. "Universal approximation theorem", Wikipedia. (2023), [Online]. Available: https://en.wikipedia.org/wiki/Universal_approximation_theorem (visited on 07/05/2023).

[8] M. Raissi. "Physics-informed neural networks (PINNs): An intuitive guide". (2019), [Online]. Available: https://towardsdatascience.com/physics-informed-neural-networks-pinns-an-intuitive-guide-fff138069563 (visited on 10/05/2022).

[9] E. Chu, "Solving differential equations with neural networks", *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/solving-differential-equations-with-neural-networks-afdcf7b8bcc4.