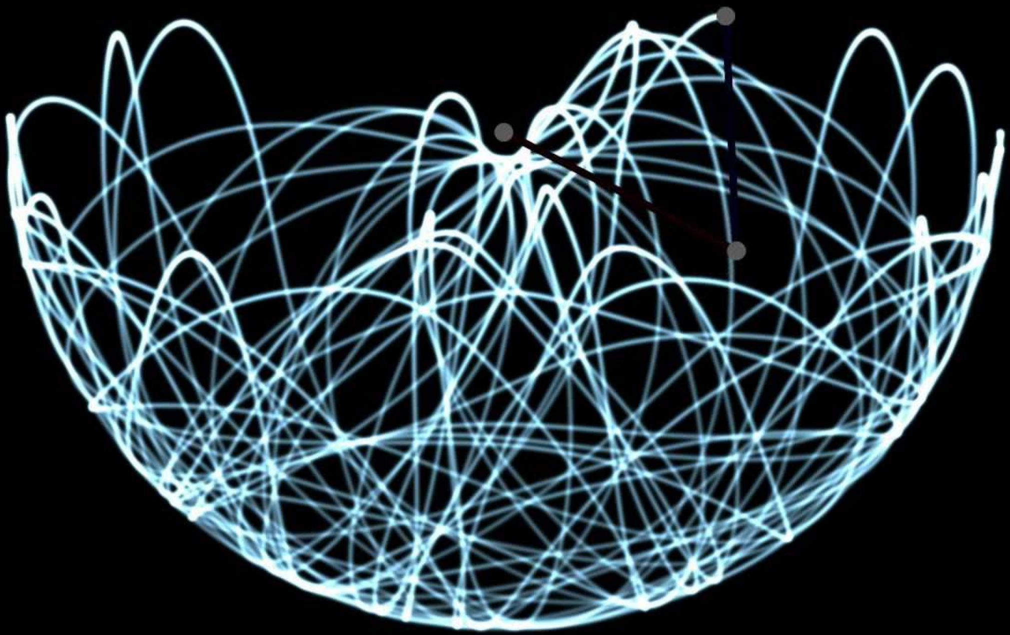
Approximating the Hamiltonian of the double pendulum using an evolutionary algorithm



Caitlin D. Smith

Layout: typeset by the author using LaTeX.
Cover illustration: Unknown artist

Approximating the Hamiltonian of the double pendulum using an evolutionary algorithm

Caitlin D. Smith
11045132

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Prof. dr. P. Sloot

Institute for Logic, Language and Computation
Faculty of Science
University of Amsterdam
Science Park 907
1098 XG Amsterdam

Jan, 2021

Chapter 1

Abstract

Michael Schmidt and Hod Lipson have revolutionised data-usage in the field of physics by developing an evolutionary algorithm which forms the equation corresponding to a given physical system [14]. They claim to do this without any knowledge of physics, however, it shall be shown that they infer the knowledge of conservation of momentum into the algorithm when producing the Hamiltonian of the double pendulum. Additionally, the shortcomings of the original paper will be laid out. This thesis has reproduced their original experiment with the addition of polynomial fitting in a bid to create a purely data driven algorithm, where the form of the system can be extracted. Yet, many steps still need to be taken in order to produce results as strong as Schmidt and Lipson have published.

Contents

Chapter 2

Introduction

AI has long been a supporting system in physics and biology, ranging from prediction of dynamical systems such as climate and ocean circulation [22] to detection of causality in complex ecosystems [19]. As AI is most commonly used to understand the patterns behind large multi-dimensional systems. With a vast amount of research done in dimensionality reduction and forecasting techniques AI is rapidly improving the human understanding of both physics and biology in regards to short term prediction and modelling.
Physics has in its turn held an important position in the advancements of AI, such as shown by Choudhary et al. [4] in their recent research; in which they use the Hamiltonian structure to improve on artificial neural networks by providing it with the knowledge of physics to describe order and chaos. Rather than using the recurrent neural networks as done in reservoir computing and many other applications, notably absent from this is the AI Feynman algorithm, Choudhary et al. use feed-forward neural networks in order to achieve this integration. As opposed to previous methods taking positions and velocities as their input - resulting in an output of their time derivatives - the Hamiltonian neural network (HNN) takes position and momenta as input and outputs the Hamiltonian for the system, which is more widely applicable. The HNN is shown by Choudhary et al. to learn the dynamics of single, double and triple pendulum among other systems.

In their opinion piece Succi and Coveney argue that Big Data and AI will not be the end of the scientific method, which has been the basis of physics up until recent times [18]. Going up against the bold claims of Big Data, Succi and Coveney outline four arguments to stress the importance of modelling. Their first argument on correlation within complex systems aims to undermine the assumption that with enough data, uncertainty will succumb to certainty. This assumption stems from the Law of Large Numbers, which requires independent data and a finite variance

[18]. Using these properties the Law of Large Numbers approximates Gaussian statistics. The Gaussian, in it's turn, pushes the odds of outliers manifesting to a near zero. Since the outliers carry uncertainty, the uncertainty of the system is also pushed towards zero. This process works very well for a large number of instances, however, as Succi and Coveney argue, not all data is uncorrelated. An example of such a system would be the double pendulum, where the derivatives of either angles heavily depend on the other [8]. Succi and Coveney go on to show that the Lorenz distribution, as found in the statistics of correlated events, is a better fit. It changes not much for inliers, but it is more tolerant towards outliers. As the mathematics become correct, the odds of manifestation of outliers becomes larger as a result and thus the claim "uncertainty will succumb to certainty" is undercut.

Succi and Coveney also pose that understanding, or wisdom, is fundamental for the human mind in order to make good decisions. As they show, wisdom is the top of the DIKW-pyramid (data-information-knowledge-wisdom) and to reach that wisdom tier one must model the knowledge which they have acquired. Using data only, it is known to reach a ceiling after which more data can lead to loss of information, a concept known as nonlinear saturation [18]. For all this, these models are still restricted, such that they must be both plausible and interpretable, which typically produces models with limited variables [13].

As such using big data and AI to create a model for complex systems could stimulate understanding, as Succi and Coveney do admit that pattern recognition is a strong feature of big data, if used to find patterns that need further explanation. Confirming this need for models, Shugihara states "Identifying causal networks is important for effective policy and management recommendations on climate, epidemiology, financial regulation and much else." [19] In their research Sugihara et al. use convergent cross mapping, a technique based on cross prediction, to test unilateral causality by measuring how the values of state Y can estimate the values in state X [19]. Showing the dependency of research on models and causality. Succi and Coveney additionally admit that the use of big data for understanding non-linear systems, non-locality and grasping higher dimensions, is of great value. It is thus important to use big data and AI to support understanding, rather than rendering modelling obsolete [18].

It is on the basis of this need for models this thesis will aim to recreate one of the fundamental evolutionary algorithms first performed by Schmidt and Lipson [14]. Their aim was to be able to extrapolate laws of physics from data without any prior knowledge of the matter. An interesting and noteworthy point of the study is that they have combined two different double pendulum systems by which it can be said that knowledge is inferred into the total system. Hence, their evidence does not support the claim that their system can retrieve the Hamiltonian of a double pendulum without prior knowledge. This thesis will strive to recreate the original algorithm, taking into account this falsified claim. Furthermore it will be attempted to distil the Hamiltonian of one single double pendulum system without the combination of two separate systems, using function properties such as polynomial fitting.

Chapter 3

Literature

In the field of physics it has been a deep-rooted method for scientist to extrapolate natural laws from data using maths and scientific thinking, modelling their knowledge into wisdom. Schmidt and Lipson have pioneered the research in automated detection of natural laws for implicit equations [14], of which Lipson has contributed to the detection of natural laws in non linear equations prior to this publication with Bongard [2]. For both methods they use symbolic regression to generate the equation form and parameters simultaneously. When finding non-linear equations this was coupled with a three step process of partitioning, automated probing and snipping [2].
Partitioning the algorithm describes each variable separately, even if it were to be coupled with other variables. For the double pendulum this might lead to insights in each pendulum's separate behaviour. The single variable equations are then integrated by substituting references to other variables form the observed data. The Bongard and Lipson used automated probing to transform this process to an active modelling system. This step uses the data it receives form the system to predict the following time-steps, which can be used to bridge gaps in acquired data. The results are then simplified using snipping; replacing certain sub-expressions by constants, uniformly chosen from a given range, to shorten the expressions.
Schmidt and Lipson forgo these steps in later research, however partitioning and snipping might still lead to a better targeted algorithm. It is in this later research that Schmidt and Lipson use the partial derivatives of candidate functions to compare to the derivatives of the corresponding data in order to calculate the error of the candidate equation in their developed fitness function [15]. In a later chapter they state that the fitness function is imperative for the use of symbolic regression [15].
Although this research is very promising for lower order expressions, their claim to not use any prior knowledge of physics proves false in the case of the double pendu-

lum. The double pendulum is seen as a system of chaos; the reaction of the system varies greatly on the initial position. Levien et al. show that a difference of one degree in starting position magnifies over a thousandfold in the starting seconds of the experiment [8]. Their research focuses on three systems of double pendula: small angle motion (described in Schmidt and Lipson's paper as low velocity in-phase oscillations [14]), zero-gravity motion (described by Schmidt and Lipson as high energy[14]) and large angle motion. It is interesting to note that Levien et al. were able to produce separate laws for each of these situations, whereas Schmidt and Lipson were only able to produce the Hamiltonian for a combined system of high energy and low velocity in-phase oscillations. Where the high energy double pendulum system resulted in the approximation of the conservation of angular momentum, the low energy double pendulum system was approached the small angle approximations and uncoupled energy terms [14]. In joining these two separate systems Schmidt and Lipson bootstrapped the algorithm to take conservation of momentum into account for the low energy system, thus taking away from the promise of the algorithm: by giving it prior knowledge of physics.
In the data found by Schmidt and Lipson significant gaps occurs when the lower pendulum crossed the upper [14]. Brunton et al. propose a method of sparse identification to smooth over these gaps [3], however, using automated probing the algorithms can be tested on their predictability in filling in these gaps [2].
Another system specialised in evolutionary algorithms is DEAP, designed by Fortin et al. [5]. It aims to make the use of evolutionary algorithms easier by providing a simpler and more understandable code. The system uses two components; a creator and a toolbox. The creator constructs classes through composition and inheritance from previous versions and allows for the dynamical addition of both data as well as functions. Due to this concept the DEAP framework is able to generate different types of evolutionary algorithms such as genetic algorithms and evolution strategies. It's toolbox allows for the user to manually choose the operators which are to be used within the algorithm. Further steps taken by Fortin et al. are based in usability of the system, rather than optimisation.
It is in further recent research that significant developments have been made in the field of automated detection of natural laws. The symbolic regression method used by Udrescu et al. in their AI Feynman algorithm [20]. In their proposed method they use simplifying properties of functions in order to fast track the search for an equation. This exploitation leads them to a structure in which the previously described methods can act as one smaller wheel, named the *brute force method*, in a larger mechanism, as shown in section 4.3, figure 4.10. Besides this *brute force* method, which is similar to that of Schmidt and Lipson, Udrescu et al; use dimensional analysis, polynomial fitting and an Artificial Neural Network (ANN). In doing so they successfully reduced computation time of the original experiment

to a set maximum. This was tested on a large problem set in which the method proved to extract 100% of the equations from the original set.
In their followup research Udrescu et al; optimise this algorithm by using learned neural-network gradients [21], although it is interesting to evaluate different neural networks in this set up. Udrescu et al; have chosen for a feed-forward, fully connected neural network with six hidden layers [20]. Adversely, it is proven that reservoir computing, a variation on the ANN, can very closely mimic and predict the form of chaotic systems, as done by Pathak et al [11].
Inherently, reservoir computing is an ANN where only the output weights are changed. Unlike typical neural networks, reservoir computing does not use nodes in the regular sense, rather the nodes are variables that hold states, which are mathematically linked to one another. For ease of reference the variables are often called nodes. In the structure of the reservoir each node is able to communicate with another, differing greatly form the rigid form of the multi-layer perceptron, and making it especially suitable for chaotic systems.
Pathak's use of parallel reservoirs in order to allow for large spatiotemporality will be able to translate well to the problems posed by Udrescu's problem set [20]. The parallel reservoirs each target different spatial regions, as to increase the accuracy of each reservoir for their specific region. Additionally it is essential in real-world applications to ensure that the 'natural' time scale of the reservoir equals that of the problem in order of magnitude [16]. Although many applications of reservoir computing have their fundamentals in robotics, their abstract applications can be related well to the field of physics. Reservoir computing, much like computational neural networks (CNN) and recurrent neural networks (RNN) finds its basis in model-free prediction, which is why it cannot stand alone in the purpose of finding models for free-form natural laws. Using it in a set up such as the AI Feynman algorithm possesses will allow the reservoir to distinguish functional properties better than any other ANN due to their capabilities in handling the time constraint. Previous research done by Schrauwen et al; shows that reservoir computing can be used for dynamic pattern classification as well as autonomous sine-generation [16].
Sahoo et al. propose a different type of network to extrapolate concise equations from data: the shallow neural network approach [13]. This research is an addition to earlier work by Martius and Lampert in which they improve on the original equation learner network (EQL) [10], to form EQL$^{\%}$. The improvements lie in the inability of the original system to incorporate division as well as better model selection criteria and reliability. Much like the AI Feynman framework EQL uses a multi-layer feed-forward network. The difference within the EQL network is that the units represent the expression's building blocks, such as operators and trigonometric functions, making it possible for the network to produce models for natural

laws. Further differences from ANN's can be found in the presence of multiplication units, which multiply the values stored in certain nodes within each layer. As multiplication is an important building block within physical expressions, this addition to the framework allows for polynomials to occur. In order to avoid overfitting Martius and Lampert limit the multiplication units to one per layer in the network [10]. Sahoo et al. have introduced the division as part of the final layer of the network to create EQL$^{\%}$, building in division units, which work similar to the multiplication units of EQL. In order to avoid complex infinite equations, which occur after a division by zero, Sahoo et al. assume that there is no real value at the poles, as well as an additional penalty for forbidden inputs. Although suitable for equation extrapolation this method has yet to be proven effective on chaotic system prediction.
As mentioned before, the AI Feynman algorithm offers a general framework upon which symbolic regression can be executed faster by using simplifying properties found in the data. Peterson converges on a 'expectation problem' in his research, which can be found in many machine learning algorithms, as well as in the AI Feynman algorithm [12]. In this problem the policy gradient methods used in most machine learning algorithms target the optimisation of expectations, rather than evaluating the x best samples. To achieve the goal of maximising the best performance Peterson developed a risk-seeking gradient which maximises the best performing equations at the cost of the worst performing equations. As Peterson specifically states this will be of great use for symbolic regression and as such might be the solution to the combination of the systems by Schmidt and Lipson.
An alternative method for creating natural laws from data has shown that the use of Long Short Term Memory (LSTM) can result in better prediction of of high dimensional dynamics, by use of recent history of the reduced state [22]. Using the time-series data Vlachas et al. have managed to approximate a forecasting rule for such a system, a concept which was based on Taken's theorem. Combining this model with a mean stochastic model ensured that the system would avoid the exponential growth of it's error. Tested on four different physical models, Vlachas et al. created a data-driven algorithm for prediction of chaotic dynamical systems, which outperforms methods such as RNNs, multilayer perceptrons (MLP), and echo state networks (ESN), yet it has not been proven to perform better than reservoir computing.

Chapter 4

Method

4.1 The physics of a double pendulum

The chaotic system that will be discussed in this thesis is that of the double pendulum. In this system one pendulum is suspended from another which is hung from a fixed point. A chaotic system is defined by the path taken from $t = 0$, as the effects of a small change in starting positions is noticeable soon after, by a great change in said path of the object.
In order to best describe the physics of the double pendulum system, the single pendulum will be outlined first. The laws which are outlined in this section will be used for data formation, as seen further in section 4.2.

In fig 4.2 a weight of mass m is suspended from a rod of length l. It's position is determined from the point of attachment, moving along the regular axis with origin at this point, with y moving positively upwards. θ_0 is the angle which is formed between the rod and the y-axis, for further use this angle will be referred to as θ.
This system of the single pendulum works on the principle of Newton's second law of energy; $F_g = ma$, as the single point of attachment forces the mass to move in a circular motion the rotational version of the second law of motion will be of better use.

$$\tau = I\alpha$$

Where τ denotes the systems total torque, I is the inertia of the mass and α equals the angular acceleration, which stems from the second time derivative of θ. Since the tension on the rod does not give any torque, the only force doing so is the gravitational force F_g, which equals $-mgl * sin(\theta)$. As such the equation becomes.

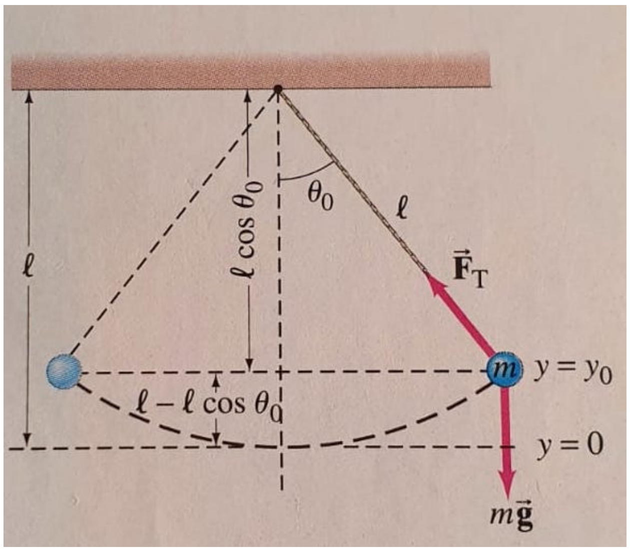


Figure 4.1: The single pendulum system, with a weight of mass m suspended on a rod of length l form a single point of attachment. The angle formed between the rod and the y-axis is stated as θ_0 and the height of mass m is defined as the difference between the deepest possible point and it's current height; calculated by $l - l * cos(\theta_0)$. The image is presented in *Physics for scientists and engineers* [6].

$$\tau = -mgl * sin(\theta)$$
$$-mgl * sin(\theta) = I\alpha$$

The inertia equals ml^2 and α can be calculated as stated below. In the following equations α will be formally denoted as the second time derivative of θ; θ^{**}.

$$-mgl * sin(\theta) = ml^2 * \theta^{**}$$
$$-g * sin(\theta) = l * \theta^{**}$$
$$\theta^{**} = -(g/l) * sin(\theta)$$

Besides being described by Newton's second law of rotation, the system also deals with conservation of energy, in which the kinetic and potential energies of the system remains stationary in a closed system.

$$E_{k1} + E_{p1} = E_{k2} + E_{p2}$$

For two separate moments in time $t1$ and $t2$. The systems' work must equal 0 since it is described as a closed system, as such,

$$\Delta E_k + \Delta E_p = W = 0$$

$$\Delta T + \Delta V = 0$$

As the potential energy is only described by the gravitational potential energy (V), it is possible to substitute this by mgh. In which h is defined by the height difference at time t from the lowest possible point, which has the length of the rod; length l. As such $h = l - l * cos(\theta)$, also seen in fig 4.2.

$$V = mg * (l - l * cos(\theta)) = mgl - mgl * cos(\theta)$$

The kinetic energy (T) is described as $0.5 * mv_\tau^2$. The velocity in this equation can be rewritten as:

$$v_\tau = r\theta^* = l\theta^*$$

With r being the radius of the circle, which in the case of the single pendulum is equal to l, and θ^* describes the first time derivative of θ, which is referred to by Schmidt and Lipson as ω [14].

$$T = 0.5 * ml^2(\theta^*)^2$$

The Lagrangian of the system is described as $L = T - V$, as such the Lagrangian for the single pendulum becomes:

$$L = 0.5 * ml^2(\theta^*)^2 - mgl - mgl * cos(\theta)$$
$$L = 0.5 * ml^2\omega^2 - mgl - mgl * cos(\theta)$$

For the single pendulum the positional data is assumed to result in the equation for circular motion, where the angular velocity and the angle will be able to describe the Lagrangian of the system, as such the target expression is of the following form with k_i being the corresponding constants, which coincides with the results found by Schmidt and Lipson [14].

$$k_1\omega^2 - k_2 cos(\theta) - k_3$$

Within the system of the double pendulum seen in figure 4.2, another weight, m_2, is suspended from the original weight, m_1, by another rod of length l_2. In order to calculate the position of the mass m_2 the following calculations are performed:

$$x_1 = l_1 * sin(\theta_1)$$
$$y_1 = -l_1 * cos(\theta_1)$$
$$x_2 = l_2 * sin(\theta_2) + x_1$$
$$y_2 = -l_2 * cos(\theta_2) + y_1$$

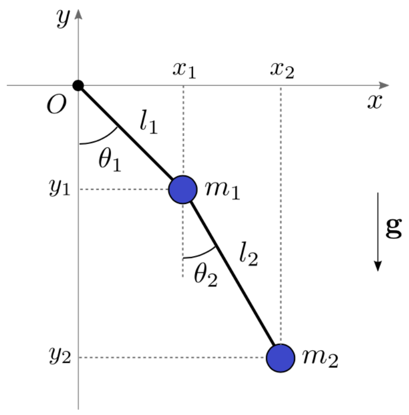


Figure 4.2: The double pendulum system, with a weight of mass m_1 suspended on a rod of length l_1 and mass m_2 attached to m_1 by rod l_2. The angle formed between the rod and the y-axis is stated as θ_1 whereas angle θ_2 is measured from the rod l_2 and the positional axis of mass m_1. This image is found on [1]

.

For a value of θ smaller than 15° the equations can be reduced to $sin(\theta) \approx \boldsymbol{\theta}$ and $cos(\theta) \approx 1$ according to the small angle approximations. This can only be done provided that $\boldsymbol{\theta}$ is denoted in radians and the input value for θ is in degrees. It is assumed that this will only be of influence for the low energy system of the double pendulum, as that would be the only system where θ might remain below the set out threshold [9].

We assume no dampening occurs as long as we keep the run time short. As such we assume the concept of conservation of angular momentum, in which the combined momentum of the system equals zero, meaning the mechanism is moment balanced [17]. The conservation of momentum of the double pendulum is denoted by the following equation:

$$m_1\theta_1^*(t_{before}) + m_2\theta_2^*(t_{before}) = m_1\theta_1^*(t_{after}) + m_2\theta_2^*(t_{after})$$

The Lagrangian of the system is then given by $L = T - V$;

$$T = 0.5(I_1 + m_2 l_3^2)\theta_1^* + 0.5I_2(\theta_1^* + \theta_2^*)^2 + m_2 l_2 l_3 \theta_1^*(\theta_1^* + \theta_2^*)cos(\theta_2)$$

$$V = g(m_1 l_1 + m_2 l_3)(1 - cos(\theta_1)) + m_2 g l_2(1 - cos(\theta_1 + \theta_2)) \text{ [8]}$$

where g is the gravitational constant, I_i is the respective moment of inertia, and l_3 is the distance between the two pivots , as shown by Levien et al [8]. The

Lagrangian is fundamental to describing the Hamiltonian of the system, as, in order to find the Hamiltonian of the system, the partial derivative of the Lagrangian towards the angles must first be calculated;

$$L_1 = \frac{dL}{d\theta_1} = \theta_1^*(I_1 + m_2 l_3^2 + I_2 + 2m_2 l_2 l_3 cos(\theta_2)) + \theta_2^*(I_2 + m_2 l_2 l_3 cos(\theta_2))$$

$$L_2 = \frac{dL}{d\theta_2} = \theta_1^*(I_2 + m_2 l_2 L_3 cos(\theta_2)) + \theta_2^* I_2$$

$$H = \Sigma_{i=1,2} L_i \theta_i^* - \mathcal{L}$$

Without dampening the Hamiltonian system is described by:

$$H = T + V \text{ [8]}$$

with the following equations of motion:

$$\theta_i^* = \frac{\delta H}{\delta L_i}$$

$$L_i^* = -\frac{\delta H}{\delta \theta_i}$$

According to this structure the form of the Hamiltonian of the double pendulum will be as follows:

$$H = T + V$$

$$H = 0.5(I_1 + m_2 l_3^2)\theta_1^* + 0.5I_2(\theta_1^* + \theta_2^*)^2 + m_2 l_2 l_3 \theta_1^*(\theta_1^* + \theta_2^*)cos(\theta_2) +$$
$$g(m_1 l_1 + m_2 l_3)(1 - cos(\theta_1)) + m_2 g l_2(1 - cos(\theta_1 + \theta_2))$$

$$H = k_1\theta_1^* + k_2\theta_1^{*2} + k_3\theta_2^{*2} + k_4\theta_1^*\theta_2^* + k_5(\theta_1^{*2} + \theta_1^*\theta_2^*)cos(\theta_2) - k_6 cos(\theta_1) + -k_7 cos(\theta_1 + \theta_2)$$
$$H = k_1\omega_1 + k_2\omega_1^2 + k_3\omega_2^2 + k_4\omega_1\omega_2 + k_5(\omega_1^2 + \omega_1\omega_2)cos(\theta_2) - k_6 cos(\theta_1) + -k_7 cos(\theta_1 + \theta_2)$$

This last equation is similar to that found by Schmidt and Lipson [14], although a large amount of components are missing;

$$H = k_1\omega_1^2 + k_2\omega_2^2 - k_3 cos(\theta_1) - k_4 cos(\theta_2) - k_5\omega_1\omega_2 cos(\theta_1 + \theta_2)$$

The Hamiltonian is, however, more commonly written in the form shown below, in which the absence of dampening is not assumed;

$$H = \Sigma L_i \theta_i^* - \mathcal{L}$$

$$H = \frac{L_1^2 I_2 + L_2^2(I_1 + m_2 l_3^2 + I_2 + 2m_2 l_2 l_3 cos(\theta_2)) - 2L_1 L_2(I_2 + m_2 l_2 l_3 cos(\theta_2))}{2I_2(I_1 + m_2 l_3^2) - 2m_2^2 l_2^2 l_3^2 cos(\theta_2)^2} ...$$

$$... + g(m_1 l_1 + m_2 l_3)(1 - cos(\theta_1)) + m_2 g l_2(1 - cos(\theta_1 + \theta_2))$$

Thus the general form of the targeted output could be;

$$H = \frac{k_1 + k_2 cos(\theta_2)}{k_3 - k_4 cos(\theta_2)^2} + k_5(1 - cos(\theta_1) + k_6(1 - cos(\theta_1 + \theta_2))$$

4.2 Data collection

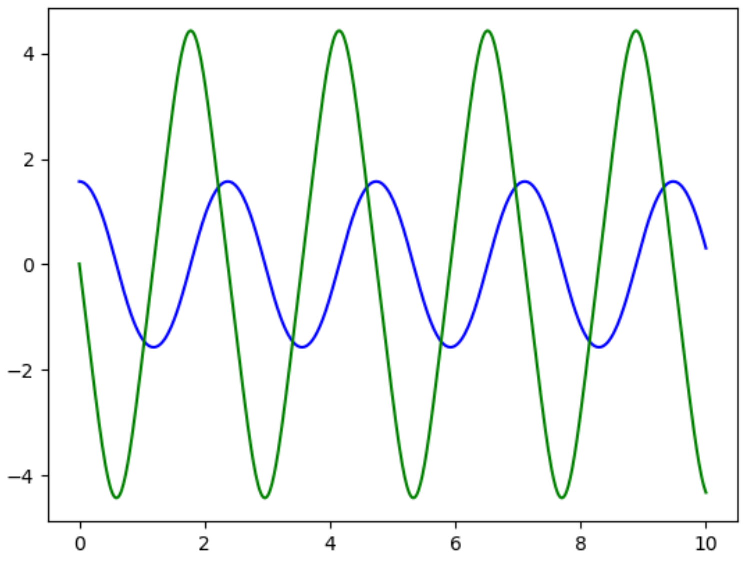
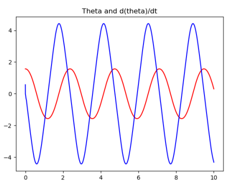


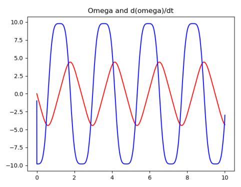
Figure 4.3: Single pendulum data; with θ in blue and ω in green.

The data provided by Schmidt and Lipson [14] has been processed, in order to trim experiment numbering, as well as trimming further calculated derivatives as the documentation of which derivative pairing was presented in which column was done illegible. The data originally gathered for the experiment by Schmidt and Lipson initially shows no anomalies. On further examination of the derivative pairings in both directions of the single pendulum it becomes clear that there are significant outliers at specific time points, which correspond with the local extrema of the original data sets. These spikes occur both in the data provided as well as in the generated data. The derivative pairings responsible for the spikes are calculated within the first step of the algorithm, using their respective time-derivatives and the following equation;

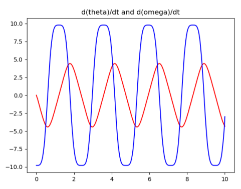
$$\frac{\Delta x}{\Delta y} = \frac{dx}{dt} / \frac{dy}{dt}$$



(a) θ in red and $\frac{d\theta}{dt}$ in blue.



(b) ω in red and $\frac{d\omega}{dt}$ in blue.



(c) $\frac{d\theta}{dt}$ in red and $\frac{d\omega}{dt}$ in blue.

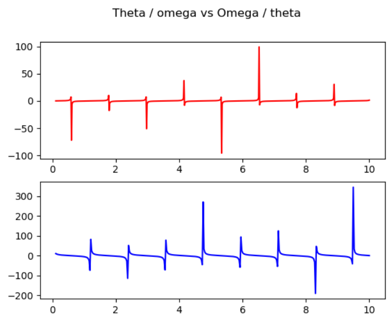
Figure 4.4: Visual representation of the single pendulum data gathered by Schidt and Lipson compared to their respective time derivatives. [14]

To test the validity of the data set given by Schmidt and Lipson another dataset is generated as shown in earlier physics calculations for the single pendulum. The double pendulum data is acquired from the matplotlib.org function [7]. Where the first row of data is stripped off to excluded division by zero possibilities. In figure 4.5a both derivative pairings of the single pendulum data sets are compared.

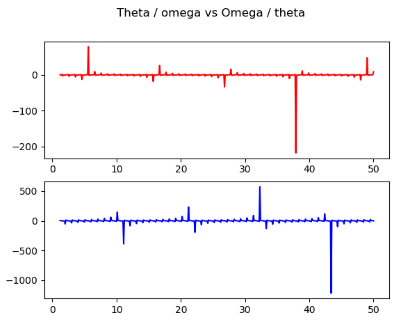
As can be seen in figure 5.1 the change of momentum in the respective systems causes a large shift in the derivative pairing values. This effect might make targeting the right equation harder, however, it will ensure a result that more closely resembles the original system.

To compare the simulated data for the double pendulum to that measured by Schmidt and Lipson both datasets have been plotted in figures 4.6 and 4.7. In both plots θ_1 is coloured blue, ω_1 in green, θ_2 in yellow and ω_2 is coloured in red.

It is not surprising that these plots differ greatly. The starting position of the measured double pendulum was not given, as such the difference in starting



(a) The derivative pairings with $\frac{d\theta}{dt}$ in respect to $\frac{d\omega}{dt}$ above and $\frac{d\omega}{dt}$ in respect to $\frac{d\theta}{dt}$ below.



(b) Colouring is equal to 4.5a, but data is spaced out over a timeset of 50 sec rather than 10.

Figure 4.5: The separate derivative pairings displayed both ways for both the original dataset acquired by Schmidt and Lipson as well as the generated dataset.

positions will lead to great differences in a matter of seconds as shown by Levien et al. [8].
For the generated data set the derivative pairings have been plotted as can be seen in figure 4.8. In these plots it can be seen that there are still large spikes, which might be interpreted as sharp changes of direction.

4.3 The algorithm

This thesis largely uses the structure from Schmidt and Lipson's research for the "brute force" method [14], as seen in figure 4.9, as well as using their collected data.
The first step of the algorithm has been described in section 4.2, by calculating the derivative pairings of the gathered data of the pendulum system. When creating the candidate expressions for the second step an expression tree is created using breadth-first placing. The tree is filled up randomly using four operators, {+, -, *, /}, and a random amount of variables, which will be added to an arbitrary amount of existing numbers. To incorporate the sine and cosine functions into the expression trees, these must be added into the algorithm separately as individual variables.
The input variables as used by Schmidt and Lipson have not been documented, as such the forming of their constants in the equations is untraceable. In their *Supporting Online Material* (SOM) the authors have included a chapter, S.5, which explains how they have created bulk constants, similar to the constants used in chapter 4.2 of this thesis, which are later replaced by separately calculated values.

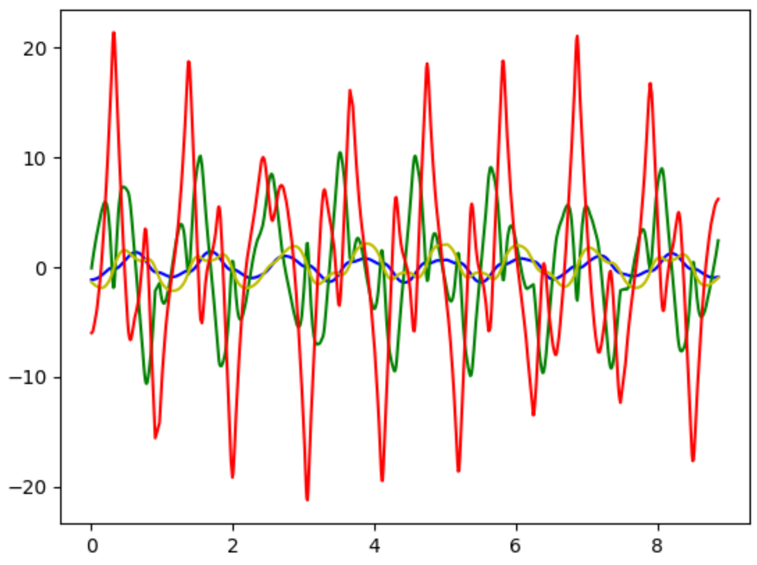


Figure 4.6: Double pendulum data; with θ_1 in blue, ω_1 in green, θ_2 in yellow and ω_2 in red.

These coefficients are determined using explicit symbolic regression, utilising the form of the already generated expression. Interestingly, this method is not used in the creation of the expression trees for the algorithm, as such it seems this is a method used to alter the coefficients, rather than creating them.

```
def create_tree(nodes, basis, random = False):
    # set random for given input in tree
    if random == False:
        startkey =  rnd.choice(nodes)
        expressionTree = Node(startkey)
        i = 0
        full = False

        # add more nodes until the tree is full
        while full == False:
            key = gen_rand_node(basis)
            full = add_node(expressionTree, key, nodes)
            i += 1
```

Listing 4.1: create_tree

In order to create and mutate the expressions efficiently, different variable and operator inputs have been tested, for which data is not shown. It is concluded that

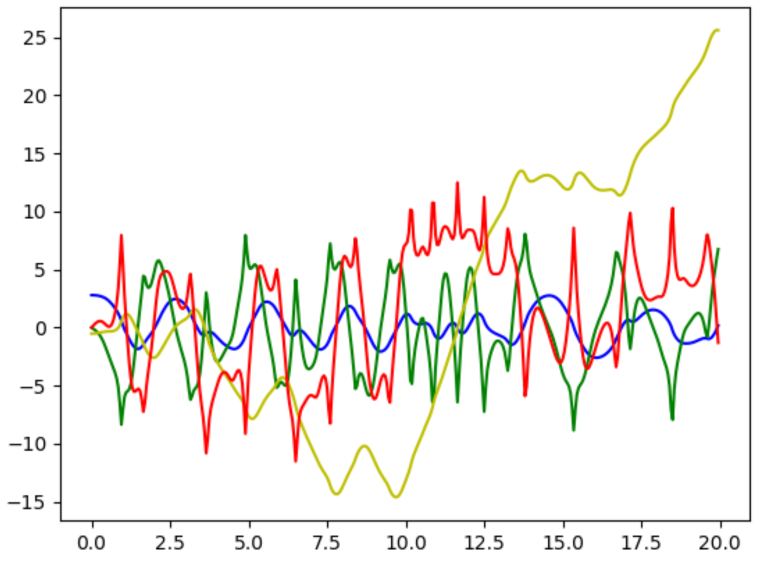
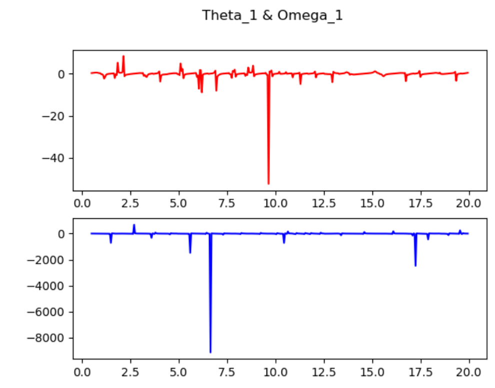


Figure 4.7: Generated double pendulum data, starting θ_1 off at 160 degrees and θ_2 at -30. ; with θ_1 in blue, ω_1 in green, θ_2 in yellow and ω_2 in red.

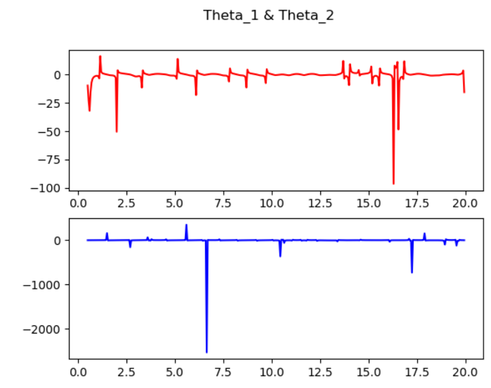
the addition of the operator added to the complexity of the equations, without improving the final errors. For the single pendulum the choice for a smaller variable input set of integers was chosen in order to maximise the odds of selecting an operator, whilst still ensuring the growth of different constants. Contrary to the single pendulum, it is imperative for the double pendulum equations to be kept smaller in size as the growth rate of the candidate expressions was significantly higher due to the larger amount of input variables. As such, for the double pendulum more integers were added in the mutation of the expressions.

The equations are then mutated by change of one node/leaf within the tree, by changing the direction under a node within the tree, the addition of a new subtree instead of a leaf and merging the expression with one of the best expressions. These functions are shown in appendix A, and are randomly selected to mutate the equation over each iteration. In total, ten equations are formed, which are then allowed to merge and grow together.
To evaluate the generated expressions the derivatives between variable pairs, as done in step three, are calculated as follows:

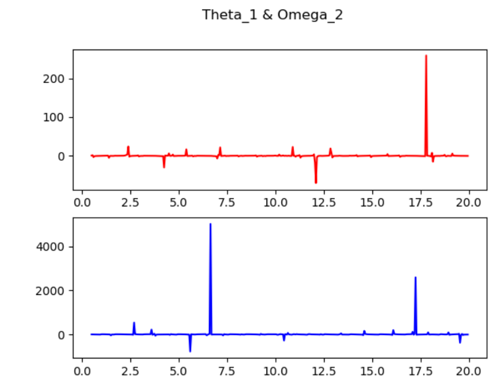
$$\frac{\delta x}{\delta y} = \frac{\delta f}{\delta y} / \frac{\delta f}{\delta x}$$



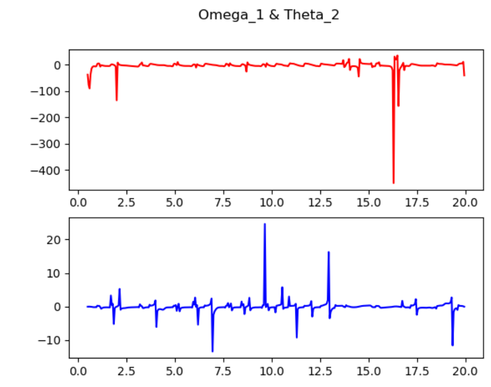
(a) $\frac{d\theta_1}{dt}$ in respect to $\frac{d\omega_1}{dt}$ above and $\frac{d\omega_1}{dt}$ in respect to $\frac{d\theta_1}{dt}$ below.



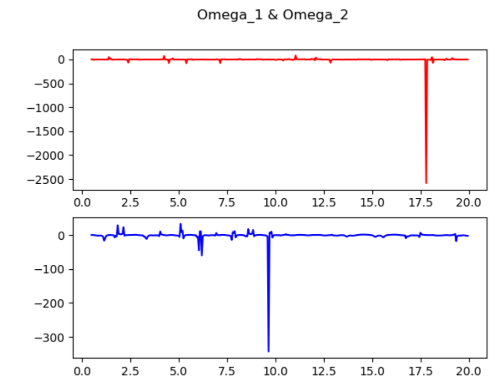
(b) $\frac{d\theta_1}{dt}$ in respect to $\frac{d\theta_2}{dt}$ above and $\frac{d\theta_2}{dt}$ in respect to $\frac{d\theta_1}{dt}$ below.



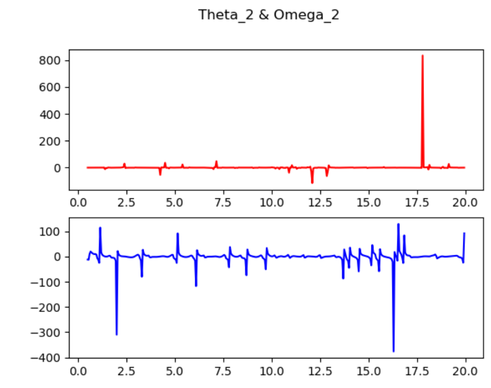
(c) $\frac{d\theta_1}{dt}$ in respect to $\frac{d\omega_2}{dt}$ above and $\frac{d\omega_2}{dt}$ in respect to $\frac{d\theta_1}{dt}$ below.



(d) $\frac{d\omega_1}{dt}$ in respect to $\frac{d\theta_2}{dt}$ above and $\frac{d\theta_2}{dt}$ in respect to $\frac{d\omega_1}{dt}$ below.



(e) $\frac{d\omega_1}{dt}$ in respect to $\frac{d\omega_2}{dt}$ above and $\frac{d\omega_2}{dt}$ in respect to $\frac{d\omega_1}{dt}$ below.



(f) $\frac{d\theta_2}{dt}$ in respect to $\frac{d\omega_2}{dt}$ above and $\frac{d\omega_2}{dt}$ in respect to $\frac{d\theta_2}{dt}$ below.

Figure 4.8: Derivative pairings of all variables of the double pendulum system, created from generated data.

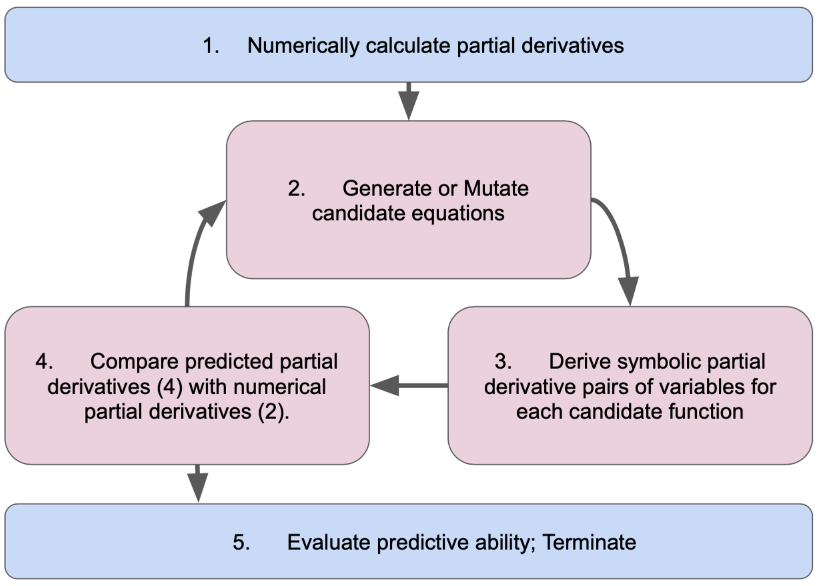

1. Numerically calculate partial derivatives
2. Generate or Mutate candidate equations
3. Derive symbolic partial derivative pairs of variables for each candidate function
4. Compare predicted partial derivatives (4) with numerical partial derivatives (2).
5. Evaluate predictive ability; Terminate

Figure 4.9: General framework of the brute force method as performed in Schmidt and Lipson's research [14]. Their first step was the collection of data, however this dataset was provided by the team, as such this is omitted for this paper. Further more on step 4, the error is checked for each equation, as per step 5, and terminates when the final criteria are met. If that is not the case, step 4 moves forward to step 2.

For variable pairings and their final error this would translate to:

$$min_{pairing}(-\frac{1}{N}\Sigma_{i=1}^{N}log(1+abs(\frac{\Delta x_i}{\Delta y_i}-\frac{\delta x_i}{\delta y_i}))\ [14]$$

Which is the comparing factor of step four. Schmidt and Lipson have found that the worst value of the derivative pairings suffices to generate correct models for their systems [14]. In step five this value will be evaluated whether it meets the minimum requirement of being higher than -0.8 for the single pendulum and 2.0 for the double pendulum. These thresholds have been assessed on the propertied of returning a result which was both legible as well as credible. With an important property being that the threshold was not so low as to benefit overfitting. Data for this testing is not shown.

In order to ensure smaller equations, necessary for better understanding as shown by Succi and Coveney [18], the algorithm both uses the penalty suggested by Sahoo et al. [13] as well as ensuring a greater chance of splitting the equation.

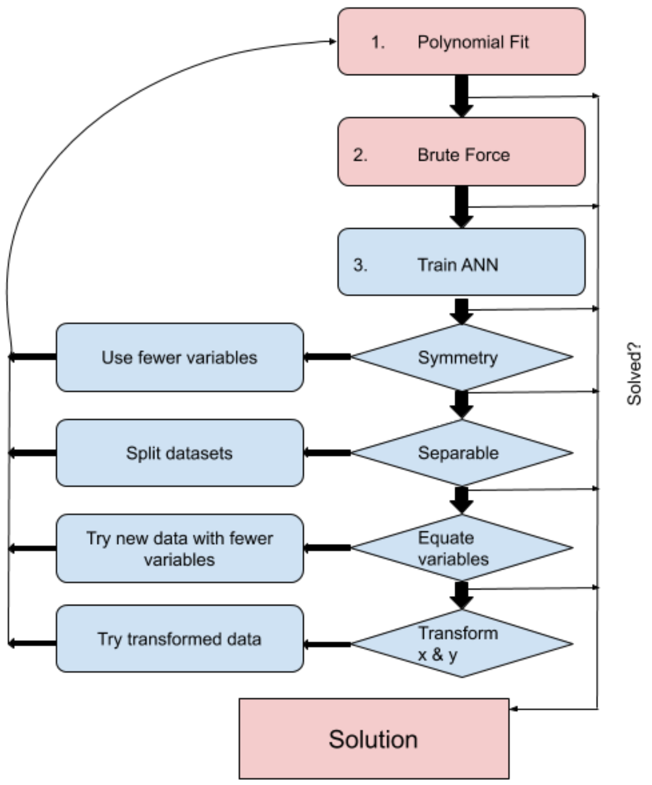


Figure 4.10: The AI Feynman framework with addition of the Schmidt and Lipson algorithm as step 2, Brute Force. Dimensional analysis has been omitted as this does not pertain to the double pendulum and the gathered data. Within the diamonds the system should check for these properties before giving this feedback to the system. Steps included in the framework for this thesis are coloured red and future expansions are coloured blue. [20]

This method is embedded in a structure similar to the AI Feynman structure [20], in which it is given x amount of time to run, after which the best candidate equations will be used to run through the neural network. x has not yet been set, as the use of the neural network is reserved for future research.
Preceding this step within the system shown in figure 4.10, is the Polynomial fitting, in which the algorithm tries to fit an arbitrary polynomial function, with a maximum degree of freedom, to the data. The expressions resulting from this function will be used to bootstrap the brute force algorithm, in order to keep the simplicity within the expressions as well as ensuring a small speed-up.

In the future it will be possible to use these equations, which are returned after running the brute force mechanism for an x amount of time, within the trained neural network, in order to evaluate symmetry and simplicity. After the neural net has evaluated the data-set as well as the functions this information will be fed back into the machine with the corresponding changes. Each iteration will allow the brute force method to run slightly longer, in order to ensure completion.

Chapter 5

Results

	Equation	**Score**
1	3.141592653589793*b**2*cos(a)	-0.5778
2	b*(3.141592653589793*cos(a) + 2)	-0.9582
3	-b + 3.141592653589793*cos(a) + 2	-1.3779
4	3.141592653589793*sin(b)*cos(a) + 1 - 1/b	-1.4199
5	-b - 2*cos(a) + 3.141592653589793	-1.4290
6	-2*b + 3.141592653589793*sin(b)*cos(a) + 5.141	-1.4673
7	b + cos(a)	-1.4740
8	-2*b + cos(a) + 2	-1.5015
9	-cos(a) - cos(b)	-1.5063
10	-cos(a) - cos(b)	-1.5063

Table 5.1: Ordered results from the single pendulum system using the input θ and ω with corresponding equations. For ease of use the programme uses a for θ and b for ω.

In table 5.1 the results of a single experiment are shown. The exit criteria of -0.8 are both met and exceeded. The algorithm substitutes a for θ and b for ω, and using these input variables it was aimed to converge onto the equation for the Lagrangian of the single pendulum system.

$$k_1\omega^2 - k_2cos(\theta) - k_3$$

As can be seen in the best result, the algorithm approximates this equation, but for a single operator. It is important to note that the value of k_i plays a great part in the evaluation, as shown in table 5.2, where significant differences have occurred in the final evaluation. The last two rows of table 5.1 are identical due to multiple possibilities of expressions intertwining and reducing back to the better of the two, creating two equals.

For the calculations within the algorithm the amount of decimals is chosen to match that of π, as defined in the math library. During comparison these values are condensed to four decimal floats, which greatly improves readability of the results as well as improving the speed of the algorithm.

	Equation	**Score**
1	b**2 + 19.8*cos(a)	-0.6775
2	1.37*b**2 + 3.29*cos(a)	-1.5147

Table 5.2: Showing the differences between scores for functions based on the same structure, but with different values for k_i

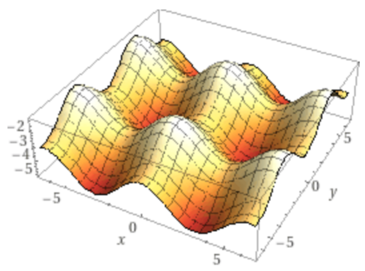
For input types x and y the circular manifold was expected, however, when looking at the best results a circular form can be extracted, although it is not the form that was searched for.

	Equation	**Score**
1	2*sin(y)/((-(-y + cos(x))/cos(y) + cos(y))*cos(y))	-0.9943
2	y*cos(x)/(2*y - sin(y) - cos(x)*cos(y))	-1.0265
3	cos(x) - cos(y) - 3.141592653589793	-1.0383
4	5*y*cos(x)	-1.066
5	5*y*cos(x)	-1.066
6	y*cos(x) + 3	-1.066
7	(1 + 1/cos(y))*cos(x)	-1.1108
8	sin(y)*cos(x)	-1.13
9	2 + (2 - y)*cos(x)/(y*cos(y))	-1.2531
10	2*x + 2/cos(y)	-1.6397

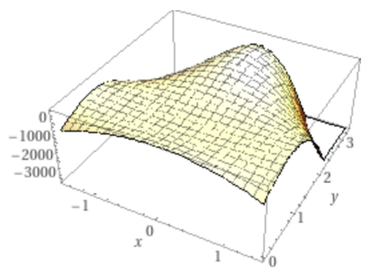
Table 5.3: Results of the single pendulum data with respect to the positional variables x and y, using the geometric functions sine and cosine.

With the best possible plot for the third best scoring equation showing clear circular folds;

When running the same algorithm for the positional data of the single pendulum without the use of trigonometric functions, the algorithm tends to move towards parabolas. Without the trigonometric functions the algorithm also moved towards one single output faster, leading away from the best function and instead



(a) Plot from Wolfram alpha of cos(x) - cos(y) - 3.141592653589793 [23]



(b) Plot from Wolfram alpha of (5.2719-10.6655*y)*(14-12*y)*(-6.2831*x**2-2*y+6.7331), where the constants have been rounded to four decimals. [23]

Figure 5.1: Two plots for the single pendulum positional dataset, where 5.1a includes trigonometry and 5.1b excludes it.

	Equation	**Score**
1	(5.271988252500436 - 10.66552067904007*y)*(14 - 12*y)*(-6.283185307179586*x**2 - 2*y + 6.733185307179586)	-1.036
2	-6.283185307179586*x**2 + y*(3*y - 2) + 0.45	-1.2306
3	270*x*(y + 4.1415)*(4*y + 12.1519)*(7*y + 10.4085)*((y + 1)*(y + 9.4247) + 2.7182)*(x + y + 8)	-1.3571
4	270*x*(y + 4.1415)*(4*y + 12.1519)*(7*y + 10.4085)*((y + 1)*(y + 9.4247) + 2.7182)*(x + y + 8)	-1.3571

Table 5.4: The output for the single pendulum positional data without the use of trigonometry. The last six outputs that were cut off equal that of the prior two.

focusing on optimising the found form of the equation. Given enough time, it will eventually move away from this again.

For the best candidate of 5.4 the output can be written as;

$$(5.271988252500436 - 10.66552067904007 * y) * (14 - 12 * y) *$$

$$(-6.283185307179586 * x * *2 - 2 * y + 6.733185307179586)$$

$$(k_1 - k_2 * y) * (k_3 - k_4 * y) * (k_4 * x * *2 - k_5 * y + k_6)$$

For optimised values of k_i this could move to a form more similar to the circular manifold.

$$(k_1 - k_2 * y)^2 * (k_4 * x * *2 - k_5 * y + k_6)$$

This equation is shown in figure 5.1b. Choosing k_5 to be set to $k_i x$ and k_6 to $k_i y$ the total system equation could become:

$$(k_1 - k_2 * y)^2 * (k_4 * x * *2 - k_5 * x * y + k_6 * y)$$

$$(k_1 - k_2 * y)^2 * (k_3 * x - k_4 * y)^2$$

Although this does take away from the algorithm, as the choice to set certain constants to variables is mathematically improper.

When running the double pendulum algorithm the following results are collected using both sine and cosine as input. As known beforehand the sine function is unnecessary, thus another trial is performed without the use of the sine function.

In the first equation of table 5.5 we see the structure of conservation of angular momentum, much like the conclusion Schmidt and Lipson have published for the high energy system. As time-steps have not been taken into account for formation of the formula it follows that $b * sin(a) + f - sin(e)$ could be more formally written as;

$$\omega_1(t^-) * sin(\theta_1)(t^-) + \omega_2(t^-) - sin(\theta_2)(t^-) = \omega_1(t^+) * sin(\theta_1)(t^+) + \omega_2(t^+) - sin(\theta_2)(t^+)$$

With (t^-) indicating the past time-step and (t^+) indicating the future time-step. Had $\omega_2 - sin(\theta_2)$, become $\omega_2 * sin(\theta_2)$ it would resemble the law of conservation better, with $sin(\theta_i)$ as approximation of the weight m_i.

Important choices for the creation of the results shown in **??** are the set length of a maximum of eight operators, after which a penalty is added. Ensuring smaller equations, reducing computing time and complexity. Furthermore, due to this choice the exit scores have been set to 1.5 and each equation will be brought back as close to eight operators as possible over each iteration.

To try to estimate the Hamiltonian, rather than the conservation of angular momentum the penalty threshold was increased to twelve and the exit criterion was raised to 2.0, to incorporate the larger expressions and the odds of overfitting. Choosing to exclude the sine function to shorten run-time, this resulted in table 5.6.

Observing the first result of table 5.6 some segments of the Hamiltonian of the high energy double pendulum system can be extracted.

$$0.14 * e * f * (2 - 3.141592653589793/b) * (a * *2 - a + f + cos(a) - cos(e) + 4)/a$$

$$k_1 * e * f * (k_2 - k_3/b) * (a^2 - a + f + cos(a) - cos(e) + k_4)/a$$

$$(k_5 ef - \frac{k_6 ef}{b})(a^2 - a + f + cos(a) - cos(e) + k_4)/a$$

$$(k_7a^2ef - k_8aef + k_9ef^2 + k_{10}ef * cos(a) - k_{11}ef * cos(e) + k_{12}ef -$$
$$\frac{k_{13}efa^2}{b} + \frac{k_{14}efa}{b} - \frac{k_{15}ef^2}{b} - \frac{k_{16}ef * cos(a)}{b} - \frac{k_{17}ef * cos(e)}{b} - \frac{k_{18}ef}{b})/a$$

From there parts of the equation can be grouped together to form segments of the Hamiltonian.

$$f^2 : \omega_2^2 : k_9ef^2 - \frac{k_{15}ef^2}{b} = k_{19}e(1 - \frac{1}{b})f^2$$

$$cos(a) : cos(\theta_1) : k_{10}ef * cos(a) - \frac{k_{16}ef * cos(a)}{b} = k_{21}ef(1 - \frac{1}{b})cos(a)$$

$$cos(e) : cos(\theta_2) : k_{11}ef * cos(e) - \frac{k_{17}ef * cos(e)}{b} = k_{22}ef(1 - \frac{1}{b})cos(e)$$

To incorporate more terms the segments can be rewritten as:

$$f^2 + f : \omega_2^2 + \omega : k_{19}e(1 - \frac{1}{b})f^2 + k_{12}ef - \frac{k_{18}ef}{b}$$
$$= k_{20}e(1 - \frac{1}{b})(f^2 + f)$$

Missing segments for the Hamiltonian as found by Schmidt and Lipson are ω_1^2 and $cos(\theta_1 - \theta_2)$. Although this does show that the Hamiltonian could be found from data, it has not yet been shown using the one system.

More output interpretations have been included in Appendix B.

	Equation	Score
1	b*sin(a) + f - sin(e)	-1.4681
2	e*(b - sin(a)) + cos(f)	-1.4730
3	a/(b*f*(sin(b) - cos(e)))	-2.3175
4	a - e + sin(f) + cos(a) - cos(b) + 3.141592653589793	-2.7404
5	3.141592653589793*sin(a) + 3.141592653589793*sin(b) - 3.141592653589793*sin(f) - 3.141592653589793*cos(e)	-2.7593
6	b*sin(e) + sin(b) + cos(f)/cos(a)	-2.8947
7	3.141592653589793*a*(b + f)*(e/2 + sin(a))*(3.141592653589793*e*sin(f) + 3.141592653589793*sin(e))*cos(b)/(f + 3.141592653589793)	-2.9467
8	2*a*(b + f)*(e/2 + sin(a))*(3.141592653589793*e*sin(f) + 3.141592653589793*sin(e))*sin(e) *cos(b)/(f + 3.141592653589793)	-3.0420
9	2*a*(b + f)*(e/2 + sin(a))*(3.141592653589793*e*sin(f) + 3.141592653589793*sin(e))*sin(e) *cos(b)/(f + 3.141592653589793)	-3.0420
10	0.02*a**2*e*(-a*b + 0.11*a) + 0.14*a**2*f*(a**2 - 0.03) + 0.11*a**2 + 0.02*a*b*e*(0.11*a - b**2) + 0.14*a*b*f*(a*b - 0.03) + 6.0e-6*a*e*f*(a - f)**2 + 0.11*a + 0.0022*b**3*(b - e) + 0.14*b**2*f*(b**2 - 0.03) + 6.0e-6*b*e*f*(b - f)**2 + 0.11*b + 0.01*e*f**2	-3.6603

Table 5.5: The Double pendulum results with respect to the high energy system using the angle and the angular velocity, with geometric functions sine and cosine. Penalty was added for any expression containing more than eight operators; the minimum necessary to create the Hamiltonian of the double pendulum system.

	Equation	Score
1	0.14*e*f*(2 - 3.141592653589793/b)*(a**2 - a + f + cos(a) - cos(e) + 4)/a	-1.8243
2	a - b/(-(a**2 + f)*cos(f) + cos(e))	-2.3355
3	0.2545522822739467*a*(cos(e) - 4)*(-0.0144*a**2*(a + cos(a) - 2) - 2.294971003328297*b + e*(1 - a)*cos(f) + 0.1228318530717959*e + (5 - e)*(0.12*a - 0.28) *(a + e + f) - (15*e + 2*cos(b))*(cos(f) - (cos(b) + 1)/b)/cos(f) - 2.294971003328297*cos(f) + 3.294971003328297)/f	-2.6386
4	b/(-e/(cos(b) + 3) + f + 4) - (-0.0144*a**2*(a - cos(f)) + 0.02*a*e + 1) /(-0.0144*a**2*(a - cos(f)) + 0.02*a*e + 0.1228318530717959*e - 2*e/f + (5 - e)*(0.12*a - 0.28) *(a + e + f) + (b - 5.43656365691809)*(b + cos(a)/5 + cos(f)) + 1)	-2.7451
5	-2*e/f + (b - 5.43656365691809)*(b + cos(a)/5 + cos(f))	-2.7989
6	-e - (-b - 1)*cos(a) + 3.141592653589793*(4*cos(a) + cos(b) + 4)*cos(f) + 4	-2.8115
7	-0.0144*a**2*(a + cos(a) - 2) - 2.294971003328297*b + e*(1 - a)* cos(f) + 0.1228318530717959*e + (5 - e)*(0.12*a - 0.28)*(a + e + f) - 0.4589942006656594*cos(a) - 2.294971003328297*cos(f) + 1 - 0.06366197723675814*e* (15*e - (f + 2)*cos(b) + 5*cos(b))/b	-2.9623
8	-0.0144*a**2*(a + cos(a) - 2) - 2.294971003328297*b + e*(1 - a)* cos(f) + 0.1228318530717959*e + (5 - e)*(0.12*a - 0.28)*(a + e + f) - 1.047197551196598*((f + cos(f)) *cos(b) + cos(f))/cos(a) - 0.4589942006656594*cos(a) - 2.294971003328297*cos(f) + 1	-2.9783
9	0.02*a**2*e*(-a*b + 0.11*a) + 0.14*a**2*f*(a**2 - 0.03) + 0.11*a**2 + 0.02*a*b*e*(0.11*a - b**2)	-3.5197
10	0.02*a**2*e*(-a*b + 0.11*a) + 0.11*a**2 + 0.02*a*b*e*(0.11*a - b**2) + 0.14*a*f*(a**2 - 0.03)	-3.5857

Table 5.6: The Double pendulum results with respect to the high energy system using the angle and the angular velocity, with only the geometric function cosine. Penalty was added for any expression containing more than twelve operators.

Chapter 6

Conclusion & Discussion

In this thesis an algorithm has been created which can produce the general form of the natural laws for both the single pendulum as well as the double pendulum. This has been done by using the framework set out by Schmidt and Lipson in their paper "Distilling Free-Form Natural Laws from Experimental Data" in 2009 [14]. As mentioned in chapter 3 the authors of the original paper have made choices in which they undermine their own conclusion. When producing the Hamiltonian of the double pendulum, Schmidt and Lipson have chosen to combine two separate double pendulum systems in order to push their outcome towards the desired form. Initially the high energy double pendulum system resulted in the approximation of the law of conservation of momentum, whereas the low energy double pendulum system was only able to produce small angle approximations. Only upon combining the two systems, were they able to procure a valid Hamiltonian. In doing so they inferred the knowledge of conservation of momentum into the low energy system. Levien et al; have proven mathematically that for each energy system a separate Hamiltonian can be calculated [8], as such the combination of the two systems should never have returned a valid Hamiltonian.
Besides this inaccuracy, Schmidt and Lipson have made the algorithm hard to reproduce. The data delivered alongside the paper in their *Supporting Online Material* (SOM) included the time derivatives of all variables, where it was not stated which variable was derived in which column.
Upon recreating the candidate expressions, multiple issues arise. In their documentation the authors have stated how they created their expressions in the form of expression trees, however the available integers for the creation of the trees were not stated. During this thesis it has been found that this is an important decision, much like choosing the input operators and variables. Although the authors have stated that choosing input operators and geometric functions is of great significance, they have failed to include their choices for each separate system tested.

Due to this faux pas, it is not possible to recreate the same results the authors. Although, as stated in chapter 5, their reported best functions do not result in the best possible error for the system. In their SOM Schmidt and Lipson have included another function which translates the bulk parameters into specific coefficients after running the algorithm. This choice does confirm the expressions found by this thesis are not the final included equations of the individual systems. Taking this into account, this thesis has been able to produce formulas of the single and the double pendulum. Of which the single pendulum produced the Lagrangian structure and the double pendulum generated the structure for the conservation of angular momentum as well as some fragments for the Hamiltonian of the double pendulum.
During this thesis a large issue has been running out of memory after twelve hours of run time on a separate server. For which a work around has been created, by saving the best equations to a text file which can be reloaded upon unsuccessful termination, until if finally does succeed.
For future research there are many different angles by which to approach this question. Firstly, expanding on the framework set up by Udrescu et al; can lead to better results as well as a faster computation time. As shown, the inclusion of the feed-forward neural network will add onto the created structure to use more equation properties for better prediction. Another interesting path can be the addition of reservoir computing to replace the ANN, as it was shown to resemble the chaos of the double pendulum more closely than a regular feed forward neural network [20]. Further methods of improving on the initial research could be the use of Sahoo et al.'s shallow neural network, which replaces the use of evolutionary algorithms by a neural network with operators and variables for nodes and with the weights as parameters [13]. Using this system it should be able to perform faster on linear equations, and by addition of the division in the system it should be possible to work on chaotic systems as well, however, this has not yet been researched.

Chapter 7

Documentation

The code used for this thesis can be found at:
https://github.com/CaitSmithUvA/Thesis_11045132

The data files included contain the following data;

$$generated\_data.txt : [time, x, y, \alpha, \theta, \omega]$$

$$dp\_data.txt : [time, x_1, y_1, x_2, y_2, \theta_1, \omega_1, \theta_2, \omega_2]$$

Where generated_data.txt contains the generated data for the single pendulum and dp_data.txt stores the produced data for the double pendulum.

To use the algorithm run main.py in the terminal. This will prompt the question if the user would like to see the data visualised. After which another prompt will be shown, asking which system to run. Here a choice of three systems can be made:

- The single pendulum; positional data
- The single pendulum; angular data
- The double pendulum

This will run until the desired output has been met using the prior established values and input. To change these it is possible to add and remove items from the input_var dictionary in main.py, as well as that it is possible to change the exit statement.

7.1 Acknowledgements

Many thanks to Prof. Dr. P. Sloot for supervising this thesis.

Bibliography

[1] Diego Assencio. *The double pendulum: Lagrangian formulation.* Feb. 2014. URL: `https://diego.assencio.com/?index=1500c66ae7ab27bb0106467c68feebc6`.

[2] Josh Bongard and Hod Lipson. "Automated reverse engineering of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 104.24 (2007), pp. 9943–9948.

[3] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937.

[4] Anshul Choudhary et al. "Physics-enhanced neural networks learn order and chaos". In: *Physical Review E* 101.6 (2020), p. 062207.

[5] Félix-Antoine Fortin et al. "DEAP: Evolutionary algorithms made easy". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 2171–2175.

[6] Doug Giancoli. *Physics for Scientists & Engineers with Modern Physics: Pearson New International Edition.* Pearson Higher Ed, 2013.

[7] John Hunter et al. *The double pendulum problem¶.* Jan. 2020. URL: `https://matplotlib.org/3.1.1/gallery/animation/double_pendulum_sgskip.html`.

[8] RB Levien and SM Tan. "Double pendulum: An experiment in chaos". In: *American Journal of Physics* 61.11 (1993), pp. 1038–1044.

[9] FMS Lima and P Arun. "An accurate formula for the period of a simple pendulum oscillating beyond the small angle regime". In: *American Journal of Physics* 74.10 (2006), pp. 892–895.

[10] Georg Martius and Christoph H Lampert. "Extrapolation and learning equations". In: *arXiv preprint arXiv:1610.02995* (2016).

[11] Jaideep Pathak et al. "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach". In: *Physical review letters* 120.2 (2018), p. 024102.

[12] Brenden K Petersen. "Deep symbolic regression: Recovering mathematical expressions from data via policy gradients". In: *arXiv preprint arXiv:1912.04871* (2019).

[13] Subham S Sahoo, Christoph H Lampert, and Georg Martius. "Learning equations for extrapolation and control". In: *arXiv preprint arXiv:1806.07259* (2018).

[14] Michael Schmidt and Hod Lipson. "Distilling free-form natural laws from experimental data". In: *science* 324.5923 (2009), pp. 81–85.

[15] Michael Schmidt and Hod Lipson. "Symbolic regression of implicit equations". In: *Genetic Programming Theory and Practice VII.* Springer, 2010, pp. 73–85.

[16] Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. "An overview of reservoir computing: theory, applications and implementations". In: *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007.* 2007, pp. 471–482.

[17] Milena Sipovac, Stefanie Winkler, and Andreas Körner. "Simulation Study on Various Double Pendulum Configurations". In: ().

[18] Sauro Succi and Peter V Coveney. "Big data: the end of the scientific method?" In: *Philosophical Transactions of the Royal Society A* 377.2142 (2019), p. 20180145.

[19] George Sugihara et al. "Detecting causality in complex ecosystems". In: *science* 338.6106 (2012), pp. 496–500.

[20] Silviu-Marian Udrescu and Max Tegmark. "AI Feynman: A physics-inspired method for symbolic regression". In: *Science Advances* 6.16 (2020), eaay2631.

[21] Silviu-Marian Udrescu et al. "AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity". In: *arXiv preprint arXiv:2006.10782* (2020).

[22] Pantelis R Vlachas et al. "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2213 (2018), p. 20170844.

[23] Stephen Wolfram. *cos(x)+-+cos(y)+-+3.141592653589793 - Wolfram: Alpha.* 2021. URL: `https://www.wolframalpha.com/input/`.

Chapter 8

Appendix A

```python
def add_node(root, data, nodes):
    # place given node at next free location
    next_node = find_next_node(root, nodes)

    if next_node.data == None:
        # tree is full
        return True

    next_node.data = data
    return False

# finds the next free node in the tree recursively
def find_next_node(tree, nodes):
    next_node = Node(None)

    if tree.left == None:
        tree.left = Node(0)
        return tree.left

    if tree.right == None:
        tree.right = Node(0)
        return tree.right

    # traverse to left side
    if tree.left != None and is_operator(tree.left.data, nodes):
        next_node = find_next_node(tree.left, nodes)

    # if no free node has been found yet, traverse right
    if next_node.data == None and tree.right != None and
   is_operator(tree.right.data, nodes):
        next_node = find_next_node(tree.right, nodes)

    return next_node
```

Listing 8.1: additional functions to create_tree

The functions shown below in Listing 7.2 up until Listing 7.5 are the mutational functions, which are used to change the expression up until the optimal expression is reached.

```
# mutate the tree accordig to operant/operator and position
def mutate(tree, target, nodes, leafs, index, changed = False):

    # if it should be changed, change for an operant
    leftOrRight = rnd.randrange(0, 2)
    if leftOrRight == 0:
        if is_operator(tree.left.data, nodes) == False:
            node = change(tree.left, leafs)
            tree.left = node
            changed = True
    else:
        if is_operator(tree.right.data, nodes) == False:
            node = change(tree.right, leafs)
            tree.right = node
            changed = True

    # if it should be changed, change for an operator or traverse left
    if leftOrRight == 0:
        if tree.left != None and is_operator(tree.left.data, nodes):
            index += 1
            if index == target:
                node = change(tree.left, nodes)
                tree.left = node
                changed = True
            mutate(tree.left, target, nodes, leafs, index, changed)
    else:
    # change or traverse right, unless change is already executed
        if tree.right != None and is_operator(tree.right.data, nodes):
            index += 1
            if index == target:
                node = change(tree.right, nodes)
                tree.right = node
                changed = True
            mutate(tree.right, target, nodes, leafs, index, changed)
```

Listing 8.2: mutation by means of mutation at location

```python
def subtree(tree, target, nodes, basis, changed = False):
        # if it should be changed, change for an operant
        leftOrRight = rnd.randrange(0, 2)
        if leftOrRight == 0:
            if is_operator(tree.left.data, nodes) == False:
                changed = True
                new = create_tree(nodes, basis)
                tree.left = new
        else:
            if is_operator(tree.right.data, nodes) == False:
                changed = True
                new = create_tree(nodes, basis)
                tree.right = new

        # if it should be changed, change for an operator or traverse left
        if leftOrRight == 0:
            if tree.left != None and is_operator(tree.left.data, nodes) and changed==False:
                next_node = subtree(tree.left, target, nodes, basis, changed)
        else:
        # change or traverse right, unless change is already executed
            if tree.right != None and is_operator(tree.right.data, nodes) and changed==False:
                next_node = subtree(tree.right, target, nodes, basis, changed)
```

Listing 8.3: mutation by means of addition of subtree

```python
def merge(tree1, tree2, nodes, operator = False):
    # if two random trees are randomly combined
    if operator == False:
        if tree1 != None and tree2 != None:
            startkey = gen_rand_node(nodes)
            expressionTree = Node(startkey)

            expressionTree.left = tree1
            expressionTree.right = tree2
        else:
            expressionTree = tree1
```

Listing 8.4: mutation by means of merging

```python
def switch(tree, nodes):
    if tree.left != None and tree.right != None:
        placeholder = tree.left
        tree.left = tree.right
        tree.right = placeholder
    return tree

def switch_in_tree(tree, target, nodes, index):
        next_node = Node(None)

        # if it should be changed, change for an operant
        if tree.left != None and is_operator(tree.left.data, nodes) == False:
            index += 1
            if index == target:
                switch(tree, nodes)
                return tree

        if tree.right != None and is_operator(tree.right.data, nodes) == False:
            index += 1
            if index == target:
                switch(tree, nodes)
                return tree

        # if it should be changed, change for an operator or traverse left
        if tree.left != None and is_operator(tree.left.data, nodes):
            index += 1
            if index == target:
                switch(tree, nodes)
            else:
                next_node = switch_in_tree(tree.left, target, nodes, index)

        # change or traverse right, unless change is already executed
        if next_node == None and tree.right.data != None and is_operator(tree.right.data, nodes):
            index += 1
            if index == target:
                switch(tree, nodes)
            else:
                next_node = switch_in_tree(tree.right, target, nodes, index)

        return tree
```

Listing 8.5: mutation by means of switching subtrees

Chapter 9

Appendix B

9.1 Single pendulum positional dataset

Shown in table 9.1 is another output for the single pendulum system with positional data and exit criterion of 1.5.

$$(-2*x+5)*(y-5)*(x+2*y+6)$$

$$(k_1x+k_2)(y-k_3)(x+k_4y+k_5)$$

$$(k_1xy+k_2y-k_6x-k_7)(x+k_4y+k_5)$$

$$\begin{gathered}k_1x^2y+k_8xy^2+k_9xy+k_{10}xy+k_{11}y^2+k_{12}y\\-k_{13}x^2-k_{14}xy-k_{15}x-k_{16}x-k_{17}y-k_{18}\end{gathered}$$

$$-\mathbf{(x+k_{19})^2}-\mathbf{(y+k_{20})^2}-k_1x^2y-k_8xy^2-k_{21}xy$$

9.2 Single pendulum angular dataset

As can be seen in table 9.2 some shown expressions might not score as well as others, but might have more promise, such as equation seven and eight, which

	Equation	**Score**
1	(66.08059122125625*x + 328.5386659644076)*(x*y + 5.821452394214521)	-1.4931
2	3.141592653589793*x + y*(x + y - 3.141592653589793) + 15.14159265358979	-1.6077
3	3.141592653589793*x - 2*y*(-2.718281828459045*x + 1) + 9.859874482048838*y - 1.858407346410207*(y - 5)*(x + 2*y + 6.141592653589793) + 23	-1.6258
4	-(-x + 5)*(5*y + 13.59140914229523) + 3	-1.6329
5	-x + 3.141592653589793*y + 4	-1.6664
6	(-2*x + 5)*(y - 5)*(x + 2*y + 6)	-1.6688
7	x + 3*y + 7.718281828459045	-1.6713
8	3.141592653589793*x + 9.859874482048838*y + 23	-1.6852
9	-y - (x + 4*y)*(y + 2) - 33.44981648870543	-1.6948
10	(-y*(7.141592653589793*y*(y - 31.41592653589793) - 8.539734222673566) + 3.141592653589793*y)*(12*x - 5.424777960769379*y - 24.27433388230814)	-1.7294

Table 9.1: Single pendulum positional data results

	Equation	**Score**
1	(cos(b) - 3)*cos(a) - 4*cos(a)**2	-0.7276
2	sin(a)*cos(a)*cos(b)	-1.4125
3	(0.09*a**2 + 0.15)*(5*a - 5*cos(a) + 12.70796326794897)*cos(b)	-1.5174
4	0.09*a**2 + a - cos(a) + cos(b) + 3.75	-1.5276
5	6*b*(a + cos(b)) - (cos(a) - cos(b) + 8.141592653589793)*sin(b)	-1.5594
6	(a + 4)*(b + 2)	-1.5936
7	-a - 3*b + 0.15*sin(b) - 2*cos(a) + 10	-1.5945
8	-a - b + cos(a) + 6.281718171540955	-1.5967
9	-b - sin(a) + 3	-1.5973
10	2*(6.283185307179586*b - cos(a) - 3)*(2*a + b + sin(a) - 6)	-1.6104

Table 9.2: Single pendulum angular data results

bare closer resemblance to the final form of the Lagrangian of the single pendulum. Another such a case it equation nine, the worst performing equation, which can be rewritten to contain the Langrangian of the system.

$$2 * (6.283185307179586 * b - cos(a) - 3) * (2 * a + b + sin(a) - 6)$$

$$k_1 * (k_2 * b - cos(a) - k_3) * (k_4 * a + b + sin(a) - k_5)$$

$$k_1 * (k_6 * ab - k_7 * a * cos(a) - k_8 * a - k_2 * b^2 - b * cos(a) - k_3 * b +$$
$$k_2 b * sin(a) - sin(a)cos(a) - k_3 sin(a) - k_9 * b + k_5 cos(a) + k_{10})$$

$$k_1 * (\mathbf{-k_2 * b^2 (-k_7 a - b - sin(a) + k_5) cos(a)} -$$
$$(k_{11} + k_6 a + k_2 sin(a))b - k_3 sin(a) + k_{10} - k_8 * a)$$