

Investigating the effectiveness of machine learning in predicting a double pendulum's motion

Exam Session: May 2025

Word Count: 3803

Research question: To what extent is multiple linear regression effective in predicting the immediate trajectory of a double pendulum given its bobs' initial positions?

CS EE World
<https://cseeworld.wixsite.com/home>
27/34 (A)
May 2025

Submitter info:
Contact me at [trackrays418 \[at\] gmail \[dot\] com](mailto:trackrays418@gmail.com) for any questions. Also, take my EE as proof that you don't need a complicated topic to score well. :]

Table of Contents

1. Introduction	3
2. Background information	5
2.1 Double pendulum.....	5
2.2 Machine learning	6
2.3 Linear regression	7
2.4 Ordinary least squares (OLS)	7
2.5 General least squares (GLS).....	9
3. Methodology	10
3.1 Independent variable.....	11
3.2 Dependent variable	11
3.3 Control variables.....	13
4. Data analysis & evaluation	15
5. Limitations	19
6. Conclusion	21
7. Further Scope	21
Bibliography	24
Appendices.....	26
Appendix 1: Code used to pull values from simulation.....	26
Appendix 2: Code of the program used	27
Appendix 3: Table of data values used	38

1. Introduction

Chaos theory is an interdisciplinary area of study that deals with complex deterministic systems that are sensitive to their initial conditions. Although predicting the long-term behavior of such systems is impossible, and prediction itself is made harder by numerical imprecisions, they are still extensively covered due to their prevalence and utility.

For example, the ‘butterfly effect’ – called so as its analogy suggests a butterfly’s flapping in Brazil could cause a tornado in Texas¹ – makes weather unpredictable in the long run due to its dependent factors. Trying to predict the state of ‘chaotic’ systems thus becomes an arduous challenge, especially when trying to make more accurate predictions for longer periods of time. However, despite this, chaos theory is extensively studied due to its real-life applications, one of the most notable being weather. This proves the relevance of this investigation to society.

Double pendulums are a simpler example of a chaotic system. They are constructed by attaching one pendulum to the end of another. For small initial angles (with respect to its position at rest), it would behave similarly to a simple pendulum. However, at larger angles, its motion grows unpredictable due to the momentum of both pendulums interfering with each other. The angles of both bobs are governed by a pair of second-order, non-linear differential equations. Since no closed form solution for the angles as a function of time exists, the equations must be numerically integrated to find the angles at an instant.

¹ Lorenz, N. Edward. "Math! Science! History!" 29 December 1972. *Predictability: Does the Flap of a Butterfly's Wings In Brazil Set Off a Tornado in Texas?* 13 June 2024.

Machine learning techniques naturally lend themselves to chaos theory. Although chaotic systems appear random over time, they are anything but. For this reason, it is still possible to make short-term predictions with reasonable accuracy (else weather forecasts would not have existed) based on existing data. As such, this investigation aims to answer the question: **to what extent is multiple linear regression effective in predicting the immediate trajectory of a double pendulum given its bobs' initial positions?**

In response to the proposed research question, two hypotheses are proposed.

- Main hypothesis (H_0): The percentage error of the value predicted by multiple linear regression is on average less than or equal to 50%, and reduces with the number of data values fed. In other words, the technique is reasonably effective in predicting the immediate trajectory of a double pendulum given its bobs' initial positions.
- Alternative hypothesis (H_1): The percentage error of the value predicted by multiple linear regression is on average greater than 50% of the latter, and does not reduce with the number of data values fed. Thus, the technique is ineffective in predicting the immediate trajectory of a double pendulum given its bobs' initial positions.

2. Background information

2.1 Double pendulum

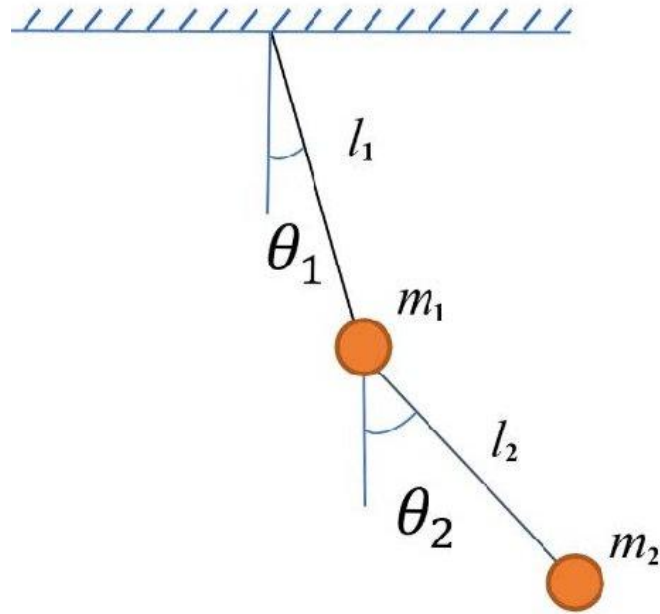


Figure 1: Diagram of a double pendulum. Adapted from Izadgoshasb, Lim and Tang²

As depicted in Figure 1, the double pendulum in question is a double ‘simple’ pendulum, where a simple pendulum consists of a spherical bob attached by a massless rod to a pivot. This specification is required as double ‘compound’ pendulums are often referred to as double pendulums, but they consist of swinging rigid bodies instead of a rod and a mass. θ_1 and θ_2 are the angles for the first and second bobs respectively, l_1 and l_2 are the lengths of the two massless rods, and m_1 and m_2 are the masses of the two bobs.

² Izadgoshasb, Izad, et al. "Improving efficiency of piezoelectric based energy harvesting from human motions using double pendulum system." Energy Conversion and Management (2019): 3. Document. 14 June 2024.

2.2 Machine learning

Machine learning falls under the larger umbrella of artificial intelligence. It is defined by the International Business Machines Corporation (IBM) as focusing on using data and algorithms to allow artificial intelligence to learn as humans do, growing more accurate over time³. This broad definition enables several techniques within machine learning, such as linear regression, logistic regression, neural networks, and decision trees⁴. Each technique has its applications, advantages, and limitations. However, to remain within the scope of this investigation, only linear regression will be used.

Applying machine learning to chaos theory and chaotic phenomena is an extensively covered subject. An immediate example is Ghorbani and colleagues' application of multiple linear regression to predict wave parameters. In their article, they employed a more nuanced approach termed "Chaos-MLR", or multiple linear regression accounting for characteristics of the system's chaotic behavior. However, many other publications on the same utilize different techniques other than linear regression. Fan and colleagues used reservoir computing, a type of recurrent neural network, to predict the behavior of two chaotic systems, Kolmogorov-Sinai entropy and the complex Ginzberg-Landau equation. Similarly, Pathak and his team made use of reservoir computing to calculate important attributes of chaotic systems, such as Lyapunov exponents.

³ IBM. *What Is Machine Learning (ML)?* | IBM. n.d. 14 June 2024.

⁴ UC Berkeley School of Information. *What Is Machine Learning (ML)? - I School Online*. 26 June 2020. 14 June 2024.

2.3 Linear regression

Linear regression, as implied by the name, takes input data (the independent variable) to estimate the coefficients of the equation of a line. The output of the equation gives the predicted value of the dependent variable. Given the nature of the double pendulum, there are two possible sets of independent variables: θ_1, θ_2 and x_1, x_2, y_1, y_2 (the coordinates). As using all four coordinates to predict each of the next four would add unneeded complexity, the angles will be the independent variables. Thus, **multiple** linear (more than one independent variable) regression is required on the two independent variables (θ_1 and θ_2), and the former dependent variable (the predicted value).

2.4 Ordinary least squares (OLS)

Ordinary least squares (OLS) models the relationship between two or more dependent variables and one independent variable by fitting a linear equation to the given data. Both simple and multiple linear regression have similar steps, thus it will not be necessary to explain both.

The simple linear equation for this method⁵ takes the form:

$$y = X\beta + \epsilon_i$$

where

- Y is an $n \times 1$ vector of dependent variable values,

⁵ Soch, Joram. *Ordinary least squares for multiple linear regression* | *The Book of Statistical Proofs*. 27 September 2019. 16 June 2024.

- X is an $n \times p$ matrix with n rows of input data and p columns of independent variables,
- β is a $p \times 1$ vector of parameters, and
- ϵ_i is an $n \times 1$ vector of errors, such that ϵ_i is independent and identically distributed in a normal distribution of mean 0 and variance σ^2 ($\epsilon_i \stackrel{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$)

Least squares regression in general optimizes the parameters by minimizing the squared ‘residuals’ (differences between the actual value and that predicted by the equation). In the multiple linear case, this is done by solving the equation:

$$X^T X \beta = X^T y$$

Solving these results in the least squares estimates of the parameters:

$$\hat{\beta} = (X^T X)^{-1} (X^T Y)$$

This equation yields the estimated parameters that minimize the sum of squared residuals, hence the moniker ‘least squares’. However, it is important to note that the ordinary least squares method assumes little to no correlation between the independent variables and thus could be inaccurate. It also assumes homoskedasticity, or that the variance of the errors is constant, amongst other things⁶. This is why the general least squares (GLS) method will also be used in this investigation.

⁶ Frost, Jim. *7 Classical Assumptions of Ordinary Least Squares (OLS) Linear Regression - Statistics By Jim*. n.d. 16 June 2024.

2.5 General least squares (GLS)

Generalized least squares (GLS) is an extension of the ordinary least squares method for multiple linear regression. GLS is used when the assumptions of OLS are violated and do not hold, specifically when the errors are heteroskedastic (the errors do not have a constant variance) or correlated.

The GLS regression linear equation is the same as that of OLS, namely:

$$y = X\beta + \epsilon$$

The chief difference between ordinary and general least squares is that GLS accounts for the structure of the error terms. In GLS, one assumes the error vector ϵ to follow a multivariate normal distribution with a mean of zero and covariance matrix Σ ; in other words, $\epsilon \sim N(0, \Sigma)$.

The GLS estimator of the parameters is given by:

$$\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} y$$

$$\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} y$$

where Σ^{-1} is the inverse of the covariance matrix of errors. Again, this formula yields the parameter estimates that minimize the generalized sum of the squared residuals.

As mentioned, GLS is more efficient than OLS when the errors are heteroskedastic or correlated, but it requires knowledge of the error covariance matrix Σ . In practice, Σ is often unknown and must be estimated from the available data, extending to the method of feasible generalized least squares (FGLS).

3. Methodology

To carry out the investigation, I had to acquire a data set of a double pendulum's motion, containing the angles of both bobs and the timestamp for the given data set. For this, I used an online simulation produced by the website myPhysicsLab⁷ to generate a comma-delimited (CSV) file of the timestamps and angles of each bob. The code used to retrieve the values from the simulation can be found in Appendix 1. However, despite the simplicity of the method, problems arose in the setup.

The simulation I used allows for customization of every aspect, from the masses of the bobs to the acceleration due to gravity, as well as the time increment between frames of the simulation. A lower time increment would result in more data points but could increase the time taken to train the model, depending on the duration for which data was collected. The reason for this is that the simulation both renders the double pendulum and prints out its bobs' angles for each frame, meaning the total number of data points would be given by

$$\text{No. of data points} = \text{time step} \times \text{duration}$$

where the time step and duration are given in seconds.

As mentioned on the simulation's website, the system is chaotic for large motions (i.e. large initial angles), but a simple linear motion for small ones (i.e. small angles). Thus, if linear regression is to predict its motion accurately, numerous data points are required.

⁷ Neumann, Erik. "myPhysicsLab Double Pendulum." *myPhysicsLab*. 19 Dec. 2023. Web. 06 Dec. 2024.

While performing trial and error with the given time controls, I found that increasing the time step allowed for longer durations but made it harder to trace the pendulum's motion between frames – even the simulation broke for large initial angles at a 0.5-second time step. Smaller time steps had the opposite problem. I finally decided to use a time step of 0.1 seconds with a total duration of 30 seconds for each set of initial angles, as that would provide enough data points for a human to trace out its motion while offering a reasonably long duration.

The program used in the investigation is available under Appendix 1. I used the Apache Commons library's implementation of OLS and GLS linear regression; further information about the specifications is listed under the control variables.

3.1 Independent variable

The independent variable will be the **number of data values** fed to the machine learning algorithm – in this case, the linear regression algorithms. Each data value is a floating-point (decimal) number of fixed length. The number of data values fed will vary from 50, 100, 150, 200, 250, to finally 300.

3.2 Dependent variable

The dependent variable will be the **percentage error** of the value predicted by the linear regression algorithm (both ordinary and general least squares, or OLS and GLS). This will be calculated through a sequence of steps.

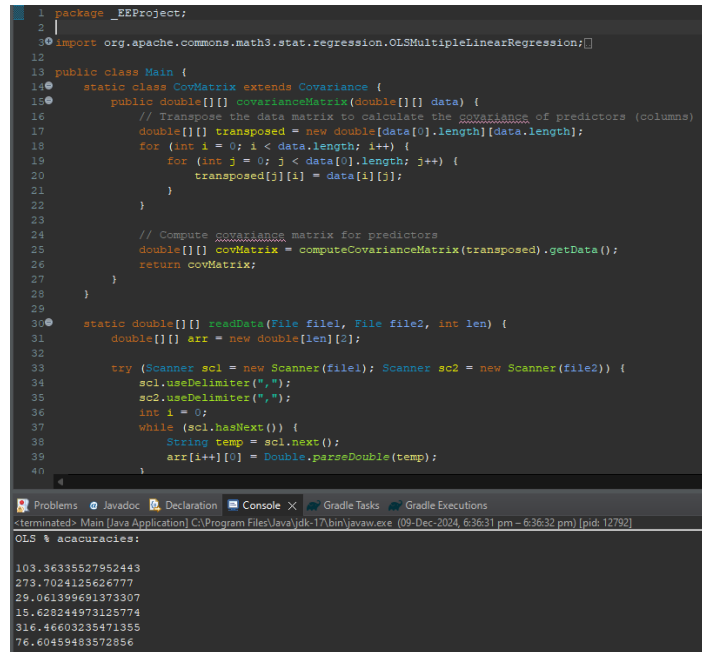
First, the initial data values are passed to the linear regression algorithm, through which the parameters and residuals of the equation are found. The predicted value is then found by summation of the residual and product of the variables and their parameters. The percentage error is then found by dividing the difference between the actual and predicted value over the actual value.

3.3 Control variables

Variable	Reason for controlling	Specified value	Method of controlling
<u>Implementation of linear regression used</u>	Different implementations may calculate the parameters or residuals differently, introducing errors and thus making comparisons meaningless.	Apache Commons Math library, version 3.6.1, from Maven	Using the same version of the library for the entire program.
<u>Simulation used</u>	Different simulations may have varying levels of accuracy in which they simulate the double pendulum's motion, preventing fair comparisons.	myPhysicsLab's double pendulum simulation	Using the same simulation when collecting the initial data.
<u>Differential equation-solving method</u>	Different methods have different accuracies, preventing fair comparisons.	Runge-Kutta method	Using the same differential equation-solving method when collecting the initial data.
<u>Time step</u>	In solving the differential equations required to find the motion of a double	A timestep of 0.1 s	Using the same time step when collecting the initial data.

	<p>pendulum, a time step is set.</p> <p>A smaller time step can result in more accurate values but increases the number of values recorded for the same time duration.</p>		
<u>System specifications</u>	<p>Exact implementations of operations vary based on the architectures of the CPU and GPU. Changing these during the experiment may affect the results, thus preventing fair comparisons.</p>	<p>CPU: i9-13980HX</p> <p>RAM: 32 GB</p> <p>GPU: NVIDIA RTX 4080</p> <p>Operating System: Windows 11 Home</p>	<p>Using the same system when collecting the initial data.</p>
<u>Initial angle</u>	<p>The kinematics of the double pendulum varies with the initial conditions, namely the initial angle of both bobs.</p>	<p>$\frac{\pi}{4}$ radians</p>	<p>Using the same initial angle when running the code to collect the initial data, for both bobs.</p>

4. Data analysis & evaluation



```
1 package _EEProject;
2
3 import org.apache.commons.math3.stat.regression.OLSMultipleLinearRegression;
4
5 public class Main {
6     static class CovMatrix extends Covariance {
7         public double[][] covarianceMatrix(double[][] data) {
8             // Transpose the data matrix to calculate the covariance of predictors (columns)
9             double[][] transposed = new double[data[0].length][data.length];
10             for (int i = 0; i < data.length; i++) {
11                 for (int j = 0; j < data[i].length; j++) {
12                     transposed[j][i] = data[i][j];
13                 }
14             }
15
16             // Compute COVARIANCE matrix for predictors
17             double[][] covMatrix = computeCovarianceMatrix(transposed).getData();
18             return covMatrix;
19         }
20     }
21
22     static double[][] readData(File file1, File file2, int len) {
23         double[][] arr = new double[len][2];
24
25         try (Scanner sc1 = new Scanner(file1); Scanner sc2 = new Scanner(file2)) {
26             sc1.useDelimiter(",");
27             sc2.useDelimiter(",");
28             int i = 0;
29             while (sc1.hasNext()) {
30                 String temp = sc1.next();
31                 arr[i++][0] = Double.parseDouble(temp);
32             }
33         }
34     }
35 }
36
37 OLS % accuracies:
38 103.36335527952443
39 273.7024125626777
40 29.061399691373307
41 15.628244973125774
42 316.46603235471355
43 76.60459483572856
```

Figure 2: Screenshot of the program's output, taken by me

With the program already providing me with the percentage errors for each independent variable value, all that was left to do was plot them on a graph. The results can be seen in Figures 3 and 4. A truncated table of the data values used can be found in Table 1; the entire table is provided in Appendix 3.

Data value no.	Angle 1	Angle 2
1	0.785	0.785
2	0.751	0.785
3	0.654	0.778
4	0.509	0.748
5	0.341	0.667
6	0.177	0.514
7	0.04	0.28
8	-0.063	-0.016
9	-0.152	-0.325
10	-0.255	-0.588
11	-0.377	-0.775
12	-0.503	-0.885
13	-0.615	-0.927
14	-0.694	-0.918
15	-0.728	-0.876

Table 1: Truncated table of data values used to train linear regression algorithm

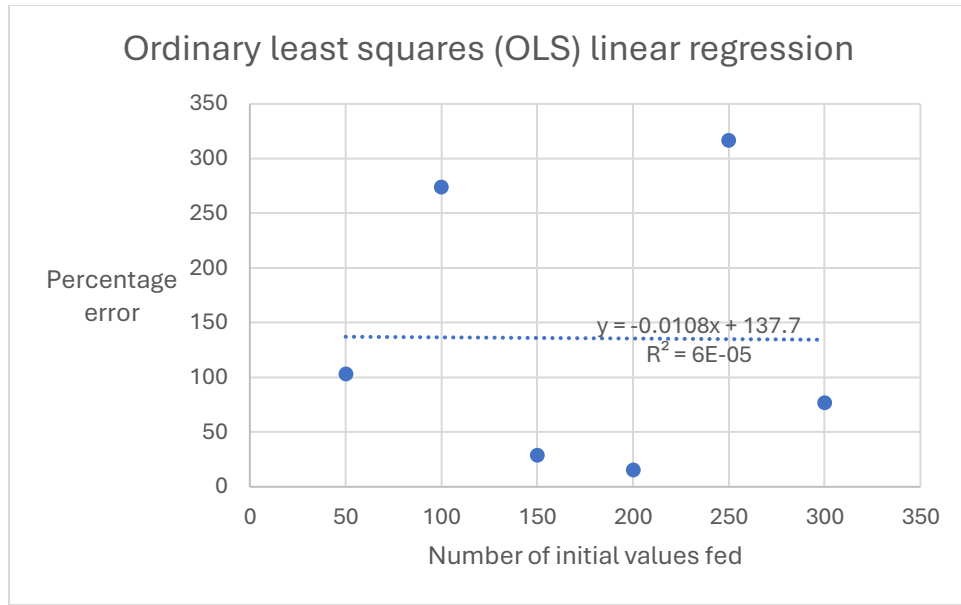


Figure 3: Graph of percentage error against the number of initial values fed for OLS regression

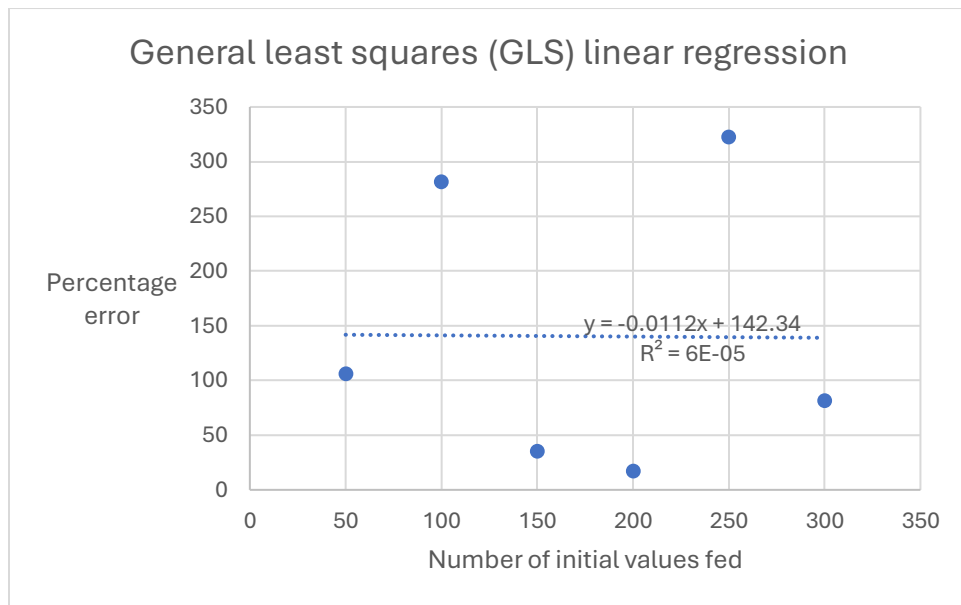


Figure 4: Graph of percentage error against the number of initial values fed for GLS regression

An initial observation of the graphs shows no correlation between the number of initial values fed to the algorithm (both OLS and GLS) and the percentage error. Proof of this is the low R^2 value, of only 6×10^{-5} ($= 0.00006$). This means that the graph above shows that the percentage error from the OLS regression model is not affected by the number of starting points for data. The percentage errors found show great variance, ranging from beyond 300% to close to 50%. Furthermore, the regression line has a very shallow slope (with a gradient of 0.0108). This made me think that factors beyond the dataset size, such as data quality, feature selection, or violations of OLS' assumptions could be influencing the model's performance.

However, the GLS regression model did not perform any better, having a similar distribution of values compared to OLS. It also has an R^2 value of 6×10^{-5} , meaning virtually no correlation between the number of data values fed and the percentage error found. A possible explanation for the unsuccessful outcome could be that the randomness of a double pendulum's motion prevents it from being predicted with simple techniques like linear regression. Even if the double pendulum started at a small enough angle, such that it oscillated like a simple pendulum, the relationship is still unlikely to be linear. Klinkachorn and Parmar, while investigating the effectiveness of different machine learning techniques in achieving the same, used a linear regression model with a polynomial feature map. Considering that this model managed to predict the kinematics for a double pendulum well at small angles, the underlying relationship is not linear.

An interesting feature to note about Figures 3 and 4 is the presence of peaks at 100 and 250 initial values fed, conveying that the percentage error was highest for those. The graphs produced already rule out any correlation between the independent and dependent variables, so there is no connection between both. However, a reason for these peaks can be found by considering the angles at that time. Considering that the pendulum moves chaotically for large angles, the chosen

initial angle $-\frac{\pi}{4}$ radians – may therefore be too high for linear regression to be able to predict. A solution for this would either be to reduce the initial angle to a value like $\frac{\pi}{6}$ radians or use a more sophisticated algorithm.

In general, other machine learning techniques might perform better than standard linear regression. In the same paper by Klinkachorn and Parmar, a long short-term memory (LSTM) neural network performed well for both chaotic and periodic (like a simple pendulum) motion, with minimal error⁸. As for the effectiveness of linear regression, however, it is completely ineffective.

5. Limitations

There were several limitations to this investigation. Firstly, the training dataset's quality: being sourced from a simulation rather than a real-life experiment, inaccuracies can arise for several reasons. First is the fact that numerically solving differential equations is bound to have some errors. Although this can be solved partially by increasing the number of terms, some errors will always remain. Furthermore, solving to a greater number of terms increases the computational cost⁹, making it less feasible depending on the scenario. Even assuming that the differential equation solver is fully accurate, limitations are brought on by the system itself. A simulation does not account for environmental factors such as air resistance, slight changes in the Earth's gravity based on location, or even friction between the joints of the double pendulum. The build of the

⁸ Klinkachorn, Sirapop and Jupinder Parmar. "Evaluating Current Machine Learning Techniques On Predicting Chaotic Systems." *CS 229 projects, Spring 2019 edition*. 2019. 2-5. Document.

⁹ Harder, Douglas Wilhelm. "Error analysis of Euler's, Heun's and the 4th-order Runge-Kutta method." n.d. *ECE 204: Course home* | Department of Electrical and Computer Engineering | University of Waterloo. 3 December 2024.

double pendulum also affects its kinematics. In the simulation, the rods are light compared to the bobs of the pendulum; however, this may not always be the case with physical double pendulums. For example, double compound pendulums are made of two rods, whose masses are evenly distributed over their lengths¹⁰, and thus would swing differently than the standard double pendulum.

Secondly, I faced difficulties in understanding the concept of covariance matrices, used in general least squares regression. Consequently, I could not manually verify the covariance matrices used or perform sample calculations to ensure the code was bug-free. This hampered the accuracy of this investigation, as I was more reliant on my programming knowledge to verify the same. If I had received guidance or otherwise managed to understand matrices and statistics, I could have calculated the covariance matrix for a small set of training data as a trial and then compared it with the calculated matrix to ensure the code worked as intended.

The relationship between θ_1 and θ_2 (the angles of the two bobs; refer to Figure 1) is also complicated to describe due to the chaotic nature of the system. Hence, finding the correlation and covariance is tricky, preventing any meaningful relationship from being established. This hampers one when investigating the behavior of the double pendulum, as this information would be useful.

Fourthly, another factor to consider would be the initial angle. For large angles, the double pendulum behaves chaotically at first, due to having two degrees of freedom; as a result, it becomes harder to predict. However, for smaller initial angles, it oscillates like a simple pendulum. As this is far simpler to describe and predict, it is possible linear regression would have been more successful for such cases.

¹⁰ christian. *The double compound pendulum*. Blog, 15 Apr. 2020. Web. 03 Dec. 2024.

6. Conclusion

This paper investigated the efficacy of using multiple linear regression, a machine learning algorithm, to predict the kinematics of a double pendulum using its current position, with the past motion fed to it as training data. The experiment used Apache Commons' implementation of ordinary and general least squares (OLS and GLS) multiple linear regression, trained on data from a simulation from the website myPhysicsLab.

From the results, it can be determined that the alternative hypothesis H_1 was correct: the percentage error is on average greater than 50% and does not reduce with the number of data values fed, for both OLS and GLS regression alike. The gradient and R^2 values for the lines of best fit for OLS and GLS regression are both approximately zero, showing no clear trend between percentage error and the number of data values used. Furthermore, most points fall above 50%, with the maximum error in both cases being close to 325%.

7. Further Scope

One way to extend this investigation would be to use data collected from a real-life experiment, rather than a simulation – however, this raises practical issues. Collecting data itself would be a challenge, as each method of doing so has its limitations. For instance, using motion tracking software to find the angles from a recording of the double pendulum may yield inaccurate results if the frames in between are blurry. If the camera's refresh rate is not high enough, this may end up happening. I experienced this while trying to collect data for my investigation; my first plan was to use angles measured from a recording using said software, but the frames during periods of

chaotic motion were too blurred. In addition, external factors may introduce more chaos to the system, making it less predictable.

Second, as mentioned earlier, machine learning algorithms – or even deep learning – may be better suited for this purpose. For instance, Li utilized random forest regression in predicting the values produced by the Lorenz system, a set of ordinary differential equations with chaotic solutions for certain parameters¹¹, just as the double pendulum is a chaotic system for high enough initial angles. Similarly, Freibergs and colleagues modeled the motion of a double pendulum using a long short-term memory (LSTM) neural network¹², finding that the LSTM model performed better than an ODE-based approach. However, it is important to note that not all machine or deep learning types may be suitable for this purpose. Steger and colleagues evaluated that physics-informed neural networks (PINNs) – a form of neural network that combines data-driven learning with information about the underlying physics – “cheated” in predicting the double pendulum’s kinematics, either adjusting the initial conditions or slightly violating the laws of physics¹³. Implementing these algorithms also requires a solid understanding of the theory behind them to avoid mistakes in their implementation. Neural networks and other deep learning techniques are particularly complex, hampering their use. Care must also be taken in the hyperparameters involved, such as the size of training data, number of epochs, or number of batches.

Third, increasing the number of inputs to the linear regression model may benefit it. For example, offering it the past two sets of angles – rather than just one – could allow it to understand the

¹¹ Li, Yuxuan. "Predicting Time Series of the Lorenz Chaotic System Using Random Forest Regression." *2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP)* (2023). Document.

¹² Freibergs, Reinis, et al. "LSTM Rollout Curriculum Using Double Pendulum." *International Journal of Machine Learning* (2024): 3-5. Document.

¹³ Steger, Sophie, Franz M. Rohrhofer and Bernhard C Geiger. "How PINNs cheat: Predicting chaotic motion of a double pendulum." *The Symbiosis of Deep Learning and Differential Equations II* (2023): 3-4. Document.

pendulum's motion better. This would result in possibly more accurate predictions. Offering the initial angles along with the most recent angles may also benefit, as knowing the initial angle helps understand whether to expect chaotic or periodic motion. However, overfitting could become an issue, where the model fits closely or exactly to its training data and listens too closely to the noise present¹⁴. Of course, this can be extended to increase the number of inputs passed to a more sophisticated model (such as an LSTM neural network), and other information like the time elapsed (from when the pendulum was released) can be passed. However, there is no guarantee that these will lead to better – or even sensible – predictions.

¹⁴ IBM. *What is overfitting?* | IBM. n.d. 04 December 2024.

Bibliography

Fan, Huawei, et al. "Long-term predictions of chaotic systems with machine learning." *Physical Review Research* 30 March 2020. Document.

Freibergs, Reinis, et al. "LSTM Rollout Curriculum Using Double Pendulum." *International Journal of Machine Learning* (2024): 3-5. Document.

Frost, Jim. *7 Classical Assumptions of Ordinary Least Squares (OLS) Linear Regression - Statistics By Jim*. n.d. 16 June 2024.

Ghorbani, M. A., et al. "Augmented chaos-multiple linear regression approach for prediction of wave parameters." *Engineering Science and Technology, an International Journal* 24 December 2016. Document.

Harder, Douglas Wilhelm. "Error analysis of Euler's, Heun's and the 4th-order Runge-Kutta method." n.d. *ECE 204: Course home | Department of Electrical and Computer Engineering | University of Waterloo*. 3 December 2024.

IBM. *What Is Machine Learning (ML)?* | IBM. n.d. 14 June 2024.

—. *What is overfitting?* | IBM. n.d. 04 December 2024.

Izadgoshasb, Izad, et al. "Improving efficiency of piezoelectric based energy harvesting from human motions using double pendulum system." *Energy Conversion and Management* (2019): 3. Document. 14 June 2024.

Klinkachorn, Sirapop and Jupinder Parmar. "Evaluating Current Machine Learning Techniques On Predicting Chaotic Systems." *CS 229 projects, Spring 2019 edition*. 2019. 2-5. Document.

Li, Yuxuan. "Predicting Time Series of the Lorenz Chaotic System Using Random Forest Regression." 2023
8th International Conference on Intelligent Computing and Signal Processing (ICSP) (2023).
Document.

Lorenz, N. Edward. "Math! Science! History!" 29 December 1972. *Predictability: Does the Flap of a
Butterfly's Wings In Brazil Set Off a Tornado in Texas?* 13 June 2024.

Neumann, Erik. *myPhysicsLab Double Pendulum*. 19 December 2023. 16 June 2024.

Pathak, Jaideep, et al. "Using machine learning to replicate chaotic attractors and calculate Lyapunov
exponents from data." *Chaos* 6 December 2017. Document.

Soch, Joram. *Ordinary least squares for multiple linear regression | The Book of Statistical Proofs*. 27
September 2019. 16 June 2024.

Steger, Sophie, Franz M. Rohrhofer and Bernhard C Geiger. "How PINNs cheat: Predicting chaotic motion
of a double pendulum." *The Symbiosis of Deep Learning and Differential Equations II* (2023): 3-
4. Document.

UC Berkeley School of Information. *What Is Machine Learning (ML)? - I School Online*. 26 June 2020. 14
June 2024.

christian. *The double compound pendulum*. Blog, 15 Apr. 2020. Web. 03 Dec. 2024.

Neumann, Erik. "myPhysicsLab Double Pendulum." *myPhysicsLab*. 19 Dec. 2023. Web. 06 Dec. 2024.

Appendices

Appendix 1: Code used to pull values from simulation

Angle 1

```
SIM_VARS.ANGLE_1 = 0.78539816339744830961566084581988;
```

```
SIM_VARS.ANGLE_2 = 0.78539816339744830961566084581988;
```

```
var angle1 = sim.getVarsList().getVariable('ANGLE_1');
```

```
var printVar = (v) => v.getValue().toFixed(3);
```

```
var memo = new GenericMemo(function(){
```

```
    println(printVar(angle1)+'');
```

```
});
```

```
simRun.addMemo(memo);
```

```
memo.memorize();
```

Angle 2

```
SIM_VARS.ANGLE_1 = 0.78539816339744830961566084581988;
```

```
SIM_VARS.ANGLE_2 = 0.78539816339744830961566084581988;
```

```
var angle2 = sim.getVarsList().getVariable('ANGLE_2');
```

```
var printVar = (v) => v.getValue().toFixed(3);
```

```
var memo = new GenericMemo(function(){
```

```
println(printVar(angle2)+'')  
  
});  
  
simRun.addMemo(memo);  
  
memo.memorize();
```

Appendix 2: Code of the program used

```
package _EEProject;  
  
import org.apache.commons.math3.stat.regression.OLSMultipleLinearRegression;  
  
import org.apache.commons.math3.linear.RealMatrix;  
  
import org.apache.commons.math3.stat.correlation.Covariance;  
  
import org.apache.commons.math3.stat.regression.AbstractMultipleLinearRegression;  
  
import org.apache.commons.math3.stat.regression.GLSMultipleLinearRegression;  
  
import java.io.File;  
  
import java.io.FileNotFoundException;  
  
import java.util.Arrays;  
  
import java.util.Scanner;  
  
public class Main {  
  
    static class CovMatrix extends Covariance {  
  
        public double[][] covarianceMatrix(double[][] data) {
```

```

// Transpose the data matrix to calculate the covariance of predictors (columns)

double[][] transposed = new double[data[0].length][data.length];

for (int i = 0; i < data.length; i++) {

    for (int j = 0; j < data[0].length; j++) {

        transposed[j][i] = data[i][j];

    }

}

// Compute covariance matrix for predictors

double[][] covMatrix = computeCovarianceMatrix(transposed).getData();

return covMatrix;

}

}

static double[][] readData(File file1, File file2, int len) {

    double[][] arr = new double[len][2];

    try (Scanner sc1 = new Scanner(file1); Scanner sc2 = new Scanner(file2)) {

        sc1.useDelimiter(",");

        sc2.useDelimiter(",");

        int i = 0;

        while (sc1.hasNext()) {

            String temp = sc1.next();

```

```

        arr[i++][0] = Double.parseDouble(temp);

    }

    i = 0;

    while (sc2.hasNext()) {

        String temp = sc2.next();

        arr[i++][1] = Double.parseDouble(temp);

    }

}

catch (FileNotFoundException e) {

    e.printStackTrace();

}

return arr;

}

static double[] getAngles(double[][] x, int n, int len) {

    double[] vals = new double[len];

    for (int i = 0; i < len; i++) {

        vals[i] = x[i][n];

    }

    return vals;

}

```

```

static double[][] copyArrayRange(double[][] x, int rows) {

    double[][] arr = new double[rows][2];

    for (int i = 0; i < rows; i++) {

        arr[i][0] = x[i][0];

        arr[i][1] = x[i][1];

    }

    return arr;

}

static double estimateValue(AbstractMultipleLinearRegression r, double[] x, int n) {

    double estimatedValue = 0;

    double[] beta = r.estimateRegressionParameters();

    double[] residuals = r.estimateResiduals();

    for (int i = 0; i < x.length; i++) {

        estimatedValue += (beta[i] * x[i]);

    }

    estimatedValue += residuals[n-1];

```

```

        return estimatedValue;

    }

    public static void main(String[] args) {

        double[][] rawData = readData(new File("pi by 4 angle 1 values.csv"), new File("pi by 4 angle 2
values.csv"), 301);

        double[][] initAngles = Arrays.copyOf(rawData, 300);

        double[][] finalAngles = Arrays.copyOfRange(rawData, 1, 301);

        System.out.println("OLS % acuracies: \n");

        // OLS Regression models for 50 values

        OLSMultipleLinearRegression ols50Angle1 = new OLSMultipleLinearRegression();

        ols50Angle1.newSampleData(getAngles(finalAngles, 0, 50), copyArrayRange(initAngles, 50));

        // OLS Regression models for 100 values

        OLSMultipleLinearRegression ols100Angle1 = new OLSMultipleLinearRegression();

        ols100Angle1.newSampleData(getAngles(finalAngles, 0, 100), copyArrayRange(initAngles, 100));

        // OLS Regression models for 150 values

        OLSMultipleLinearRegression ols150Angle1 = new OLSMultipleLinearRegression();

```

```
ols150Angle1.newSampleData(getAngles(finalAngles, 0, 150), copyArrayRange(initAngles, 150));
```

```
// OLS Regression models for 200 values
```

```
OLSMultipleLinearRegression ols200Angle1 = new OLSMultipleLinearRegression();
```

```
ols200Angle1.newSampleData(getAngles(finalAngles, 0, 200), copyArrayRange(initAngles, 200));
```

```
// OLS Regression models for 250 values
```

```
OLSMultipleLinearRegression ols250Angle1 = new OLSMultipleLinearRegression();
```

```
ols250Angle1.newSampleData(getAngles(finalAngles, 0, 250), copyArrayRange(initAngles, 250));
```

```
// OLS Regression models for 300 values
```

```
OLSMultipleLinearRegression ols300Angle1 = new OLSMultipleLinearRegression();
```

```
ols300Angle1.newSampleData(getAngles(finalAngles, 0, 300), copyArrayRange(initAngles, 300));
```

```
// % error for OLS for 50 values
```

```
System.out.println(
```

```
    (estimateValue(ols50Angle1, initAngles[49], 50) - finalAngles[49][0])
```

```
    / finalAngles[49][0] * 100
```

```
);
```

```
// % error for OLS for 100 values
```

```
System.out.println(
```

```
    (estimateValue(ols100Angle1, initAngles[99], 100) - finalAngles[99][0])
```



```

        / finalAngles[99][0] * 100

    );

// % error for OLS for 150 values

System.out.println(

    (estimateValue(ols150Angle1, initAngles[149], 150) - finalAngles[149][0])

    / finalAngles[149][0] * 100

);

// % error for OLS for 200 values

System.out.println(

    (estimateValue(ols200Angle1, initAngles[199], 200) - finalAngles[199][0])

    / finalAngles[199][0] * 100

);

// % error for OLS for 250 values

System.out.println(

    (estimateValue(ols250Angle1, initAngles[249], 250) - finalAngles[249][0])

    / finalAngles[249][0] * 100

);

// % error for OLS for 300 values

System.out.println(

```

```

        (estimateValue(ols300Angle1, initAngles[299], 300) - finalAngles[299][0])

        / finalAngles[299][0] * 100

    );

    CovMatrix cm = new CovMatrix();

    // GLS Regression models for 50 values

    GLSMultipleLinearRegression gls50Angle1 = new GLSMultipleLinearRegression();

    gls50Angle1.newSampleData(getAngles(finalAngles, 0, 50), copyArrayRange(initAngles, 50),
cm.covarianceMatrix(copyArrayRange(initAngles, 50)));

    // GLS Regression models for 100 values

    GLSMultipleLinearRegression gls100Angle1 = new GLSMultipleLinearRegression();

    gls100Angle1.newSampleData(getAngles(finalAngles, 0, 100), copyArrayRange(initAngles, 100),
cm.covarianceMatrix(copyArrayRange(initAngles, 100)));

    // GLS Regression models for 150 values

    GLSMultipleLinearRegression gls150Angle1 = new GLSMultipleLinearRegression();

    gls150Angle1.newSampleData(getAngles(finalAngles, 0, 150), copyArrayRange(initAngles, 150),
cm.covarianceMatrix(copyArrayRange(initAngles, 150)));

    // GLS Regression models for 200 values

    GLSMultipleLinearRegression gls200Angle1 = new GLSMultipleLinearRegression();

```

```
gls200Angle1.newSampleData(getAngles(finalAngles, 0, 200), copyArrayRange(initAngles, 200),  
cm.covarianceMatrix(copyArrayRange(initAngles, 200)));
```

```
// GLS Regression models for 250 values
```

```
GLSMultipleLinearRegression gls250Angle1 = new GLSMultipleLinearRegression();
```

```
gls250Angle1.newSampleData(getAngles(finalAngles, 0, 250), copyArrayRange(initAngles, 250),  
cm.covarianceMatrix(copyArrayRange(initAngles, 250)));
```

```
// GLS Regression models for 300 values
```

```
GLSMultipleLinearRegression gls300Angle1 = new GLSMultipleLinearRegression();
```

```
gls300Angle1.newSampleData(getAngles(finalAngles, 0, 300), copyArrayRange(initAngles, 300),  
cm.covarianceMatrix(copyArrayRange(initAngles, 300)));
```

```
System.out.println("\nGLS % acuracies: \n");
```

```
System.out.println(gls300Angle1.estimateRegressionParameters().length);
```

```
// % error for GLS for 50 values
```

```
System.out.println(  
  

```

```
(estimateValue(gls50Angle1, initAngles[49], 50) - finalAngles[49][0])
```

```
/ finalAngles[49][0] * 100
```

```
);
```

```
// % error for GLS for 100 values
```

```

System.out.println(

    (estimateValue(gls100Angle1, initAngles[99], 100) - finalAngles[99][0])

    / finalAngles[99][0] * 100

);

// % error for GLS for 150 values

System.out.println(

    (estimateValue(gls150Angle1, initAngles[149], 150) - finalAngles[149][0])

    / finalAngles[149][0] * 100

);

// % error for GLS for 200 values

System.out.println(

    (estimateValue(gls200Angle1, initAngles[199], 200) - finalAngles[199][0])

    / finalAngles[199][0] * 100

);

// % error for GLS for 250 values

System.out.println(

    (estimateValue(gls250Angle1, initAngles[249], 250) - finalAngles[249][0])

    / finalAngles[249][0] * 100

);

```

```
// % error for GLS for 300 values

System.out.println(

    (estimateValue(gls300Angle1, initAngles[299], 300) - finalAngles[299][0])

    / finalAngles[299][0] * 100

);

}

}
```

Appendix 3: Table of data values used

Data value no.	Angle 1	Angle 2
1	0.785	0.785
2	0.751	0.785
3	0.654	0.778
4	0.509	0.748
5	0.341	0.667
6	0.177	0.514
7	0.04	0.28
8	-0.063	-0.016
9	-0.152	-0.325
10	-0.255	-0.588
11	-0.377	-0.775
12	-0.503	-0.885
13	-0.615	-0.927
14	-0.694	-0.918
15	-0.728	-0.876
16	-0.708	-0.812
17	-0.633	-0.73
18	-0.51	-0.631
19	-0.354	-0.504
20	-0.184	-0.34
21	-0.02	-0.134
22	0.129	0.104
23	0.261	0.348
24	0.381	0.569
25	0.49	0.747
26	0.581	0.874

27	0.643	0.951
28	0.669	0.983
29	0.656	0.971
30	0.604	0.914
31	0.522	0.808
32	0.419	0.648
33	0.304	0.439
34	0.18	0.196
35	0.042	-0.052
36	-0.116	-0.273
37	-0.287	-0.452
38	-0.453	-0.589
39	-0.593	-0.693
40	-0.689	-0.777
41	-0.731	-0.846
42	-0.717	-0.898
43	-0.653	-0.922
44	-0.55	-0.903
45	-0.424	-0.824
46	-0.296	-0.67
47	-0.183	-0.436
48	-0.091	-0.139
49	0.001	0.172
50	0.122	0.436
51	0.276	0.623
52	0.443	0.731
53	0.599	0.778
54	0.718	0.79
55	0.778	0.789
56	0.772	0.785

57	0.699	0.779
58	0.571	0.757
59	0.409	0.698
60	0.24	0.575
61	0.089	0.374
62	-0.029	0.101
63	-0.122	-0.208
64	-0.218	-0.493
65	-0.333	-0.711
66	-0.458	-0.851
67	-0.575	-0.919
68	-0.667	-0.931
69	-0.717	-0.901
70	-0.718	-0.845
71	-0.664	-0.768
72	-0.56	-0.673
73	-0.416	-0.555
74	-0.251	-0.403
75	-0.084	-0.211
76	0.072	0.016
77	0.212	0.258
78	0.338	0.488
79	0.454	0.681
80	0.554	0.826
81	0.627	0.826
82	0.666	0.922
83	0.666	0.972
84	0.626	0.98
85	0.553	0.943
86	0.455	0.858

87	0.344	0.719
88	0.226	0.527
89	0.097	0.291
90	-0.052	0.038
91	-0.219	-0.199
92	-0.389	-0.396
93	-0.543	-0.546
94	-0.66	-0.658
95	-0.725	-0.746
96	-0.733	-0.817
97	-0.686	-0.874
98	-0.595	-0.91
99	-0.474	-0.911
100	-0.342	-0.859
101	-0.22	-0.737
102	-0.12	-0.535
103	-0.032	-0.259
104	0.075	0.055
105	0.215	0.344
106	0.378	0.564
107	0.54	0.703
108	0.675	0.772
109	0.76	0.794
110	0.781	0.794
111	0.735	0.788
112	0.627	0.78
113	0.475	0.764
114	0.305	0.72
115	0.144	0.624
116	0.011	0.455

117	-0.091	0.21
118	-0.184	-0.089
119	-0.291	-0.388
120	-0.413	-0.635
121	-0.534	-0.805
122	-0.635	-0.901
123	-0.701	-0.935
124	-0.72	-0.921
125	-0.686	-0.874
126	-0.601	-0.805
127	-0.473	-0.715
128	-0.316	-0.603
129	-0.149	-0.462
130	0.012	-0.283
131	0.159	-0.068
132	0.293	0.17
133	0.415	0.404
134	0.522	0.61
135	0.607	0.772
136	0.659	0.886
137	0.672	0.954
138	0.644	0.979
139	0.581	0.962
140	0.49	0.898
141	0.383	0.782
142	0.269	0.61
143	0.146	0.387
144	0.008	0.133
145	-0.152	-0.117
146	-0.323	-0.332

147	-0.487	-0.499
148	-0.621	-0.623
149	-0.709	-0.715
150	-0.74	-0.789
151	-0.713	-0.849
152	-0.636	-0.893
153	-0.523	-0.909
154	-0.392	-0.881
155	-0.262	-0.789
156	-0.151	-0.62
157	-0.061	-0.372
158	0.034	-0.067
159	0.16	0.24
160	0.314	0.492
161	0.478	0.664
162	0.626	0.759
163	0.732	0.796
164	0.779	0.8
165	0.759	0.793
166	0.675	0.783
167	0.538	0.769
168	0.372	0.737
169	0.204	0.663
170	0.057	0.523
171	-0.057	0.308
172	-0.151	0.027
173	-0.252	-0.277
174	-0.369	-0.547
175	-0.491	-0.748
176	-0.6	-0.873

177	-0.679	-0.93
178	-0.716	-0.934
179	-0.702	-0.9
180	-0.636	-0.838
181	-0.523	-0.755
182	-0.377	-0.651
183	-0.214	-0.518
184	-0.05	-0.351
185	0.103	-0.147
186	0.243	0.084
187	0.372	0.32
188	0.487	0.536
189	0.582	0.712
190	0.646	0.843
191	0.673	0.929
192	0.659	0.972
193	0.607	0.972
194	0.524	0.928
195	0.421	0.834
196	0.309	0.684
197	0.192	0.478
198	0.062	0.231
199	-0.088	-0.027
200	-0.256	-0.261
201	-0.426	-0.447
202	-0.575	-0.585
203	-0.683	-0.684
204	-0.737	-0.761
205	-0.732	-0.824
206	-0.673	-0.873

207	-0.571	-0.901
208	-0.443	-0.893
209	-0.308	-0.829
210	-0.187	-0.692
211	-0.09	-0.475
212	-0.001	-0.187
213	0.11	0.127
214	0.254	0.406
215	0.416	0.611
216	0.571	0.735
217	0.695	0.792
218	0.767	0.806
219	0.774	0.8
220	0.714	0.789
221	0.596	0.774
222	0.439	0.748
223	0.268	0.692
224	0.11	0.579
225	-0.019	0.394
226	-0.119	0.136
227	-0.216	-0.163
228	-0.327	-0.45
229	-0.448	-0.679
230	-0.563	-0.833
231	-0.653	-0.915
232	-0.705	-0.939
233	-0.71	-0.92
234	-0.663	-0.869
235	-0.568	-0.793
236	-0.434	-0.696

237	-0.277	-0.572
238	-0.112	-0.416
239	0.045	-0.223
240	0.191	-0.001
241	0.326	0.235
242	0.448	0.458
243	0.552	0.649
244	0.629	0.795
245	0.67	0.897
246	0.67	0.957
247	0.63	0.975
248	0.556	0.95
249	0.458	0.877
250	0.348	0.75
251	0.234	0.564
252	0.112	0.328
253	-0.028	0.067
254	-0.189	-0.181
255	-0.361	-0.388
256	-0.521	-0.543
257	-0.648	-0.653
258	-0.725	-0.734
259	-0.743	-0.8
260	-0.704	-0.852
261	-0.615	-0.889
262	-0.494	-0.896
263	-0.358	-0.856
264	-0.228	-0.75
265	-0.121	-0.565
266	-0.032	-0.302

267	0.067	0.008
268	0.198	0.308
269	0.354	0.544
270	0.514	0.7
271	0.652	0.782
272	0.745	0.809
273	0.777	0.808
274	0.743	0.796
275	0.646	0.78
276	0.502	0.758
277	0.334	0.714
278	0.167	0.625
279	0.026	0.468
280	-0.085	0.238
281	-0.181	-0.049
282	-0.287	-0.346
283	-0.405	-0.599
284	-0.523	-0.782
285	-0.623	-0.891
286	-0.689	-0.937
287	-0.711	-0.934
288	-0.683	-0.894
289	-0.606	-0.828
290	-0.486	-0.738
291	-0.337	-0.623
292	-0.175	-0.477
293	-0.015	-0.296
294	0.136	-0.082
295	0.276	0.15
296	0.405	0.379

297	0.519	0.581
298	0.607	0.743
299	0.662	0.86
300	0.676	0.86
301	0.65	0.935

Table 2: Data values used in training the linear regression algorithm