ENGR 091: Nonlinear Dynamics and Chaos
Lab 1: Linearity, Nonlinearity, and the Compound Pendulum

Neghemi Micah Kimaru Boruett

24 February 2025

1 Introduction

Physical systems can be modeled using mathematical methods. The modeling equations account for various components of a non-linear, oscillating system, and can be used to predict the behavior of such a system. In this experiment, the movement of a pendulum was analyzed. We assumed an equation that explained its damping and oscillatory behavior. Physical properties such as position and momentum are affected by the mass, center of gravity, and angle at which the pendulum was released.

1.1 Theory

The motion for a frictionless pendulum can be modeled as:

Eq.1

$$\ddot{\theta} + \frac{mgL}{I_o} sin(\theta) = 0$$

where m is the mass of the pendulum. L is the distance from the pivot to the pendulum's center of gravity. I_o is the moment inertia about the pivot.

Assuming a small-angle approximation, the equation becomes:

Eq.2.1

$$\ddot{\theta} + \frac{mgL}{I_o}\theta = 0$$

We can find the particular solution of this differential equation by considering the initial condition:

Eq.2.2

$$\theta(0) = \theta_o$$

$$\ddot{\theta}(0) = 0$$

By solving the differential equation, we get a final solution of:
Eq.3.1

$$\theta(t) = \theta_o cos(\sqrt{\frac{mgL}{I_o}}t)$$

Where$\sqrt{\frac{mgL}{I_o}}$ is the angular frequency and the frequency can be found by:
Eq.3.2

$$f = \sqrt{\frac{mgL}{I_o}}\frac{1}{2\pi}$$

In using the actual measurements of the pendulum, we can rewrite Eq.3.2 as:
Eq.3.3

$$\theta(t) = \theta_o cos(7.6t)$$

Interpreting the equation, the angular frequency is 7.6 $\frac{rad}{s}$ and the frequency is 1.206 Hz.
This shows a linear model of the systems assuming no damping effects. If we consider damping effects, such as air resistance, that causes the system to come to an eventual stop, we can model the differential equation as:
Eq.4

$$\ddot{\theta} + M\theta + \omega^2 sin(\theta) = 0$$

where:

$$\omega^2 = \frac{mgL}{I_o}$$

and M is the damping term with units of Hertz (Hz) or T^-1
Solving for the particular solution for Eq.4 using the initial conditions in Eq.2.2 we get:

$$\theta(t) = e^{\frac{-Mt}{2}} cos(\frac{\sqrt{-M^2 + 4\omega^2}}{2}t)$$

If we disregard the small-angle approximation, the movement of a pendulum with a damping effect can be modeled using the nonlinear differential equation:
Eq.5

$$\ddot{\theta} + M\theta + sin(\theta) = 0$$

Overall, the experimental data can be fitted to a general differential equation with a damping and frequency constant. This equation is shown below:
Eq.6

$$\theta(t) = Ae^{Bt} cos(Ct - D)$$

The parameters: A, B, C, and D are found by fitting the equation as closely as possible to the data. An analysis of how accurate these estimations are will be discussed later in the results section. We will also compare the experimental frequency ($\frac{C}{2\pi}$) with the theoretical value from the linear assumption: 1.206 Hz.

2 Procedure

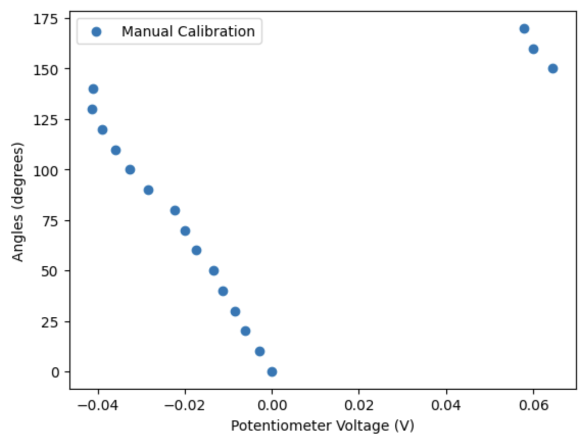
A pendulum was connected to the Vishay Spectrol 157-11103 potentiometer and an Analog Discovery 3 signal-generating device. Using the program WaveForms, data was collected to measure the voltage for a stationary pendulum for angles 0^o to 180^o on a 10^o increment. This data will be used to calibrate the readings for a moving pendulum.

To gather data for the motion, the pendulum's starting angle was adjusted for 6 different cases. The movement and data of the pendulum for each of these cases was recorded for 10 seconds.

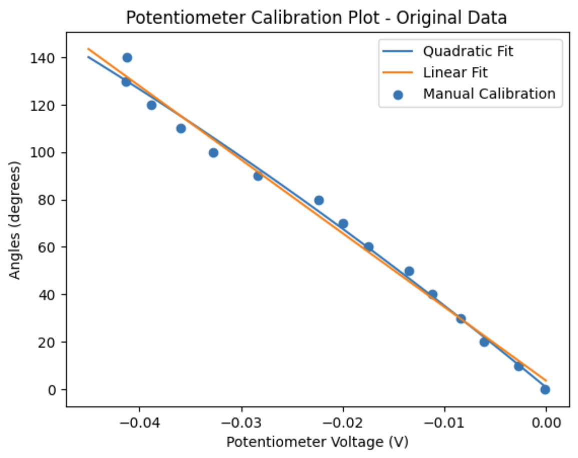
Using Python, the data for the moving pendulum at each case was fit to a linear differential equation. The accuracy of the equation in estimating the experiment is analyzed in the results section.

3 Results & Discussion

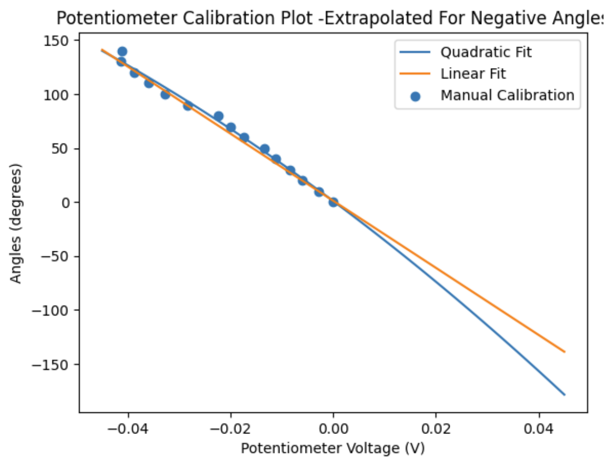
3.1 Calibration



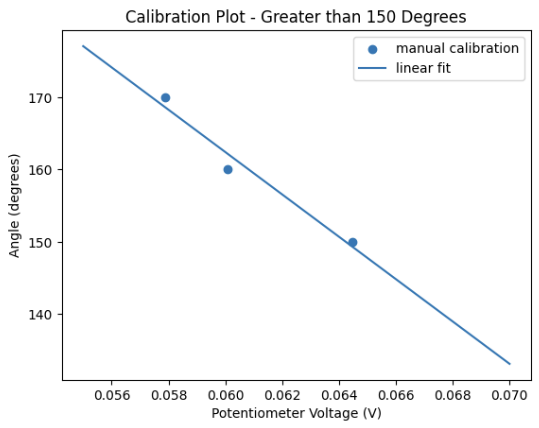
(a) Original Data



(b) Values up to 150 Degrees



(c) Extrapolated for Negative Values

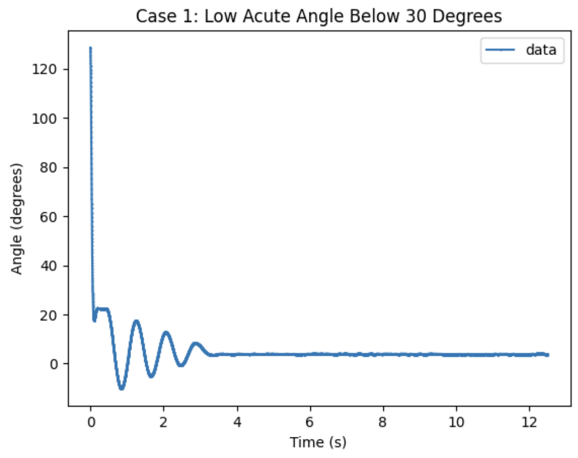


(d) Values Greater than 150 Degrees

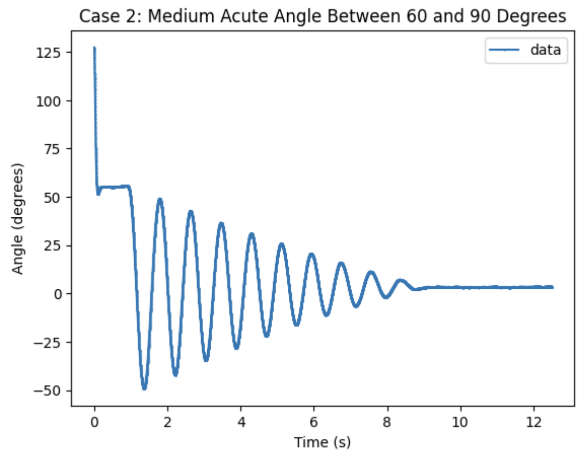
Figure 1: Calibration Data Plots

Figure 1 above displays the calibration plots. In our manual calibration, the data largely followed a linear trend, with angles exceeding 150° classified as outliers. Consequently, we employed Python's curve_fit to derive the linear function coefficients best suited to our measurements. This process was carried out separately for the primary data set, illustrated in Figure 1(c), and for the outliers, shown in Figure 1(d).

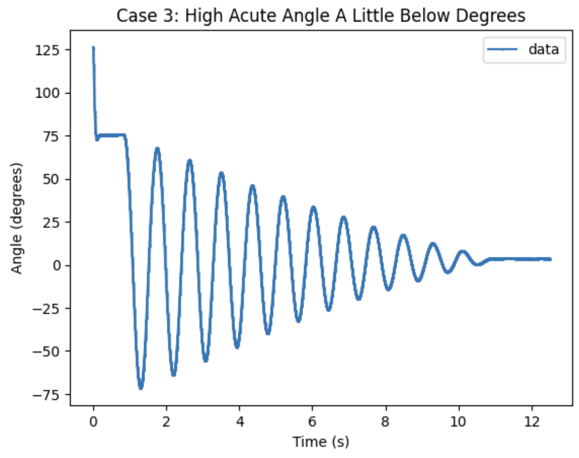
3.2 Dynamic Experiments



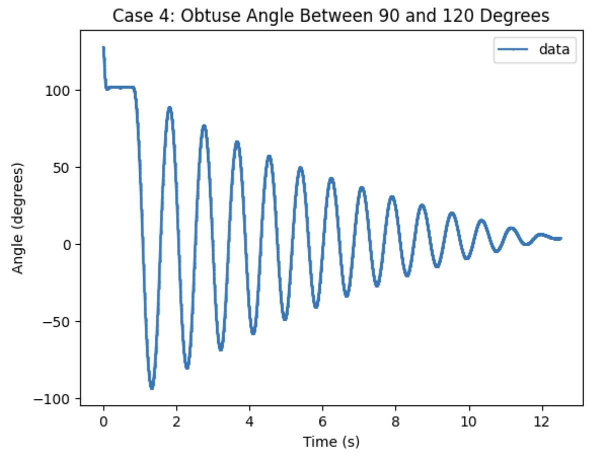
(a) Low Acute Angle



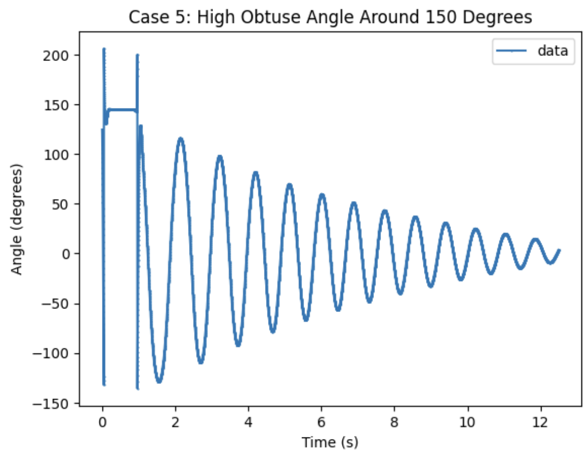
(b) Medium Acute Angle



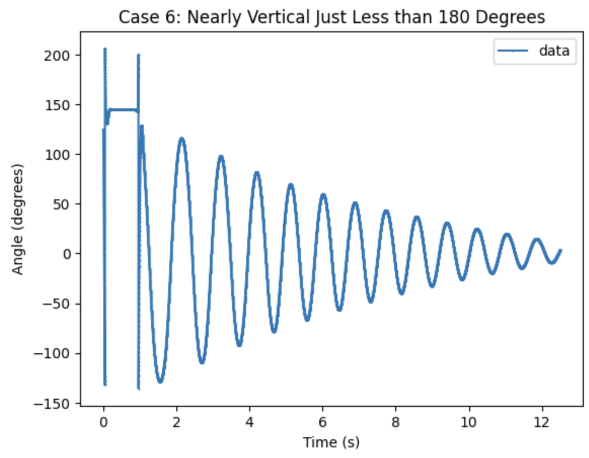
(c) High Acute Angle



(d) Obtuse Angle



(e) High Obtuse Angle



(f) Nearly Vertical

Figure 2: Calibrated Angles Plot Case 1 through Case 6

Figure 2 above illustrates the range of angles explored, from low acute angles to nearly vertical ones just shy of 180°. Our calibration effectively mapped the potentiometer voltage to these angles. However, for the first few seconds of the obtuse angle plots, the behavior was not entirely predictable. Still, the plot closely reflects the observations and measurements captured by the Analog Discovery 3, aligning with potentiometer voltage.

We determined the oscillation frequency by counting the successive intervals between distinct peaks in the plots (see Figures 2 and 3), with these counts recorded as cycles in Table 1. Dividing the total number of cycles by the elapsed time in seconds yielded the frequency value. This approach was chosen for its simplicity and reasonable accuracy given our data, and in every case, the error was less than 1% compared to the theoretical prediction.

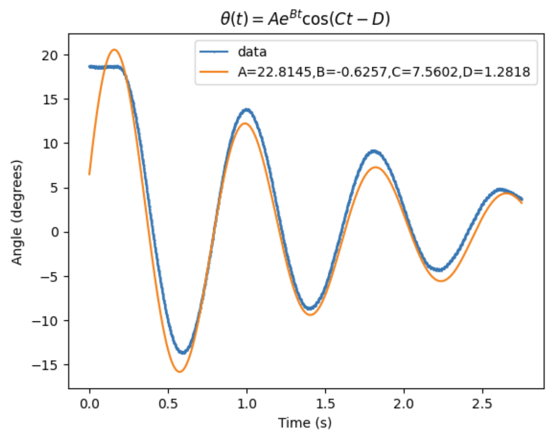
Table 1: Calculated Values of Oscillation Frequency

Case	Angle	Cycles	Seconds (s)	Frequency (Hz)	Error (%)
1	Low Acute	3	2.5	1.2	0.50
2	Medium Acute	9	7.5	1.2	0.50
3	High Acute	10	8.3	1.204	0.17
4	Obtuse	12	10	1.2	0.50
5	High Obtuse	12	10	1.2	0.50
6	Nearly Vertical	12	10	1.2	0.50

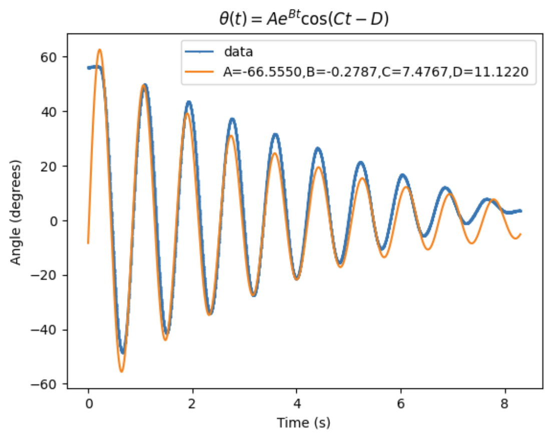
3.2.1 Data fitting

Figure 3 below displays our data alongside the fitted curves. Some post-processing was required to adjust the data for the fitting. In case 1, we observed an exponential decay when we included data from the end of the experiment, after the pendulum had settled and the angle was zero, so we truncated the data for a proper fit. For cases 4 through 6, we truncated the data from the beginning, as the release point was delayed at times, and in case 6, there was a sudden jump in values that caused irregular data capture. Cases 2 and 3 required truncation from both ends. Ultimately, data truncation resulted in the fitted curves shown in orange across all cases.

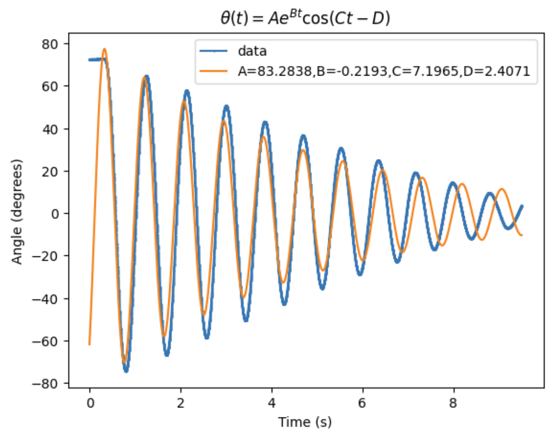
Visually, our model captured cases 1 and 2 very well, with slight variations in the amplitude that could be explained by friction and drag not easily captured by our model. However, as the initial angle increased, the fit increasingly deviated from the expected behavior over time. In the first few seconds, the fitted curves followed the trend in almost every case except for cases 5 and 6, where the frequency was also affected.



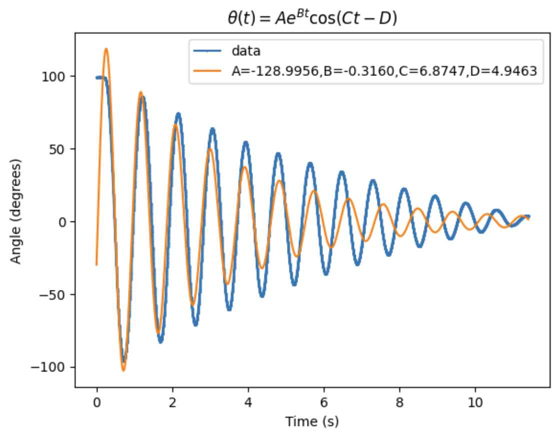
(a) Case 1: Low Acute Angle



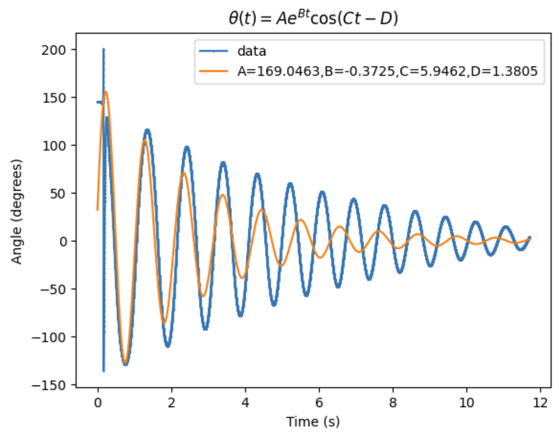
(b) Case 2: Medium Acute Angle



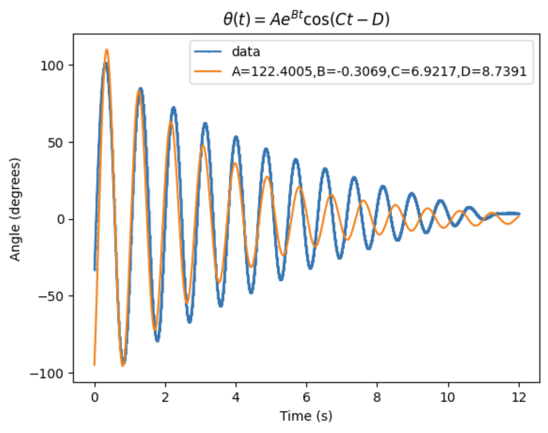
(c) Case 3: High Acute Angle



(d) Case 4: Obtuse Angle



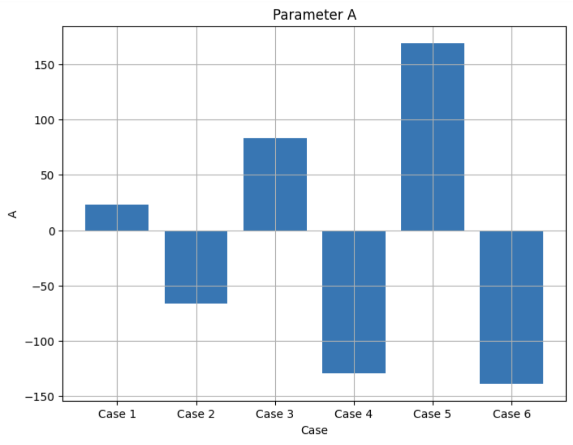
(e) Case 5: High Obtuse Angle



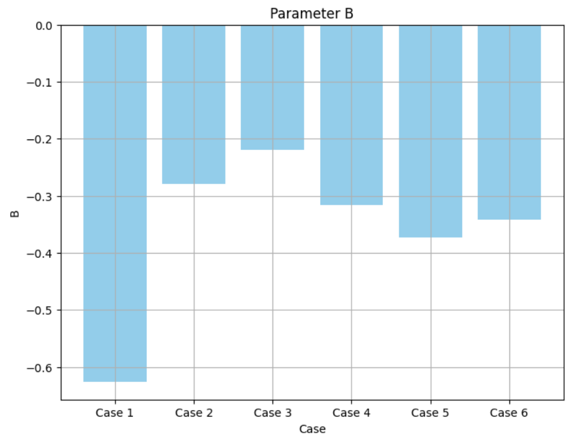
(f) Case 6: Nearly Vertical

Figure 3: Data Fitting Plots Case 1 through Case 6

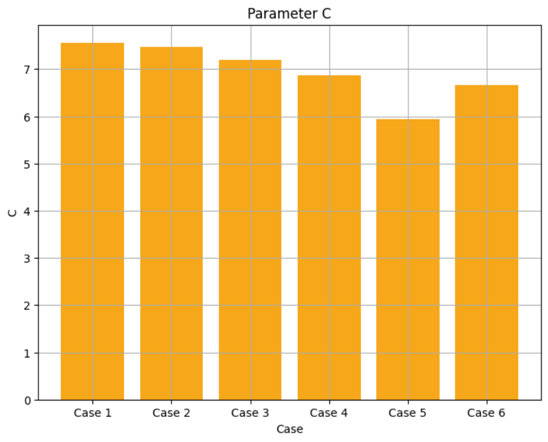
3.2.2 Parameters



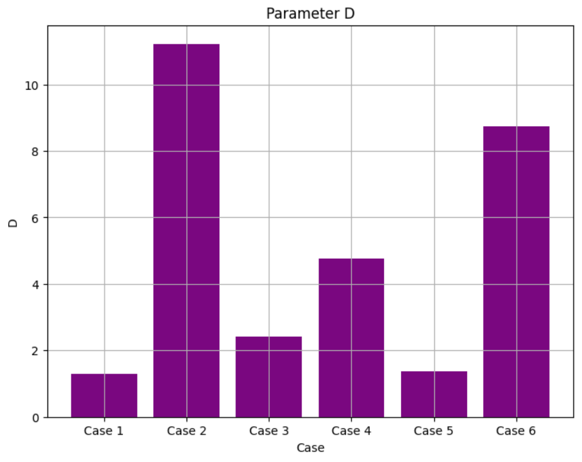
(a) Parameter A



(b) Parameter B



(c) Parameter C



(d) Parameter D

Figure 4: Parameter A through D for Each Case

Figure 4 above illustrates the variation of the different fitting parameters. As observed in Figure 3, the amplitude parameter A remains relatively low for cases 1 and 2 compared to cases 3 through 6, although A itself does not carry significant physical meaning.

Parameter B consistently appears as a negative value across all cases, with case 1 exhibiting the most negative value, corresponding to the quickest halt. In our experiment, case 6 took the longest to stop; however, due to data truncation during post-processing, its value for B was not the lowest. The negative nature of B underscores that our system's behavior is governed by initial conditions in the absence of an external energy source.

Parameter C, representing the angular frequency, shows minimal variation across different cases. It is estimated to be around 7 rad/s, which aligns well with the frequency reported in Table 1. Notably, in case 1, the estimated angular frequency precisely matches the measured value, with further details presented in Table 2, where case 5 (a high obtuse angle) exhibited the highest recorded error.

Finally, parameter D, like parameter A, lacks significant physical meaning. It does not display any consistent or observable pattern but serves to adjust the data along the horizontal axis, a result likely influenced by the data truncation process.

Table 2: Fit Frequency Compared to Actual Calculated Frequency

Case	Angle	Angular Freq(ω) - Fit	Fit Freq(Hz)	Actual Freq (Hz)	Error (%)
1	Low Acute	7.56	1.203	1.2	0.25
2	Medium Acute	7.48	1.19	1.2	0.84
3	High Acute	7.197	1.145	1.204	5.15
4	Obtuse	6.875	1.094	1.2	9.69
5	High Obtuse	5.946	0.946	1.2	26.85
6	Nearly Vertical	6.922	1.102	1.2	8.89

4 Conclusion

Our experiments validated the theoretical predictions of the system's dynamic behavior. The calibration process effectively mapped potentiometer voltage to corresponding measurements, and most of the data followed a linear trend.

We fitted the experimental data from each dynamic experiment to a model defined by four parameters (A, B, C, and D). The overlay plots of the experimental data with the best-fit model curves show that the model captures the dynamics well for low and medium acute angles, corresponding to cases 1 and 2, where the system's behavior was most predictable. Parameters B and C are of significant physical meaning. Parameter B, consistently negative, indicates the system's deceleration, with case 1 exhibiting the most negative value and a rapid halt. Parameter C, representing the angular frequency, remains stable at approximately 7 rad/s and perfectly matches the measured value in case 1. In contrast, parameters A and D mainly adjust the scaling and horizontal positioning of the data and do not have direct physical interpretations.

Although the linear model provided an excellent fit for cases 1 and 2, its effectiveness diminished for experiments involving larger angles (cases 3 through 6), where additional factors such as friction, drag, alongside other non-linearities contributed to deviations from the expected behavior. Further refinement is needed to account for the complexities at larger angles.

5 Appendix

5.1 Code

```
import pandas as pd
from statistics import mean,median,stdev
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from scipy.signal import find_peaks

# Define a function to which data will be fit.
def theta_solution(t,a,b,c,d):
    return a*np.exp(b*t)*np.cos(c*t-d)

def linear_fit(v,m,c):
    return m*v +c

def quadratic_fit(v,a,b,c):
    return a*v**2+b*v + c

# Define a function that can read a data file.
def getData(file):
    tab1 = pd.read_csv(file,
                    index_col=None,
                    header=None,
                    names=['Time','Voltage','AveVoltage'],
                    skiprows=12,
                    )
    return tab1

#calibration data
arr = np.array([])
data = getData("data/Deg0.csv")
v_data_offset = data["Voltage"][3000]
for i in range(0,150,10):
    file_name = "data/Deg"+str(i)+".csv"
    data = getData(file_name)
    v_data = data["Voltage"]-v_data_offset
    avg = np.mean(v_data)
    arr = np.append(arr,avg)
degrees = np.arange(0,150,10)
```

```
linearfit = curve_fit(linear_fit,arr,degrees)
m = linearfit[0][0]
n = linearfit[0][1]
coef = [m,n]
voltage = np.linspace(-0.045,0,100)
linearfit_vals = linear_fit(voltage,m,n)
quad_fit = curve_fit(quadratic_fit,arr,degrees)
p = quad_fit[0][0]
q = quad_fit[0][1]
r = quad_fit[0][2]
quadraticfit_vals = quadratic_fit(voltage,p,q,r)

voltage = np.linspace(-0.045,0.045,100)
linearfit_vals = linear_fit(voltage,m,n)
quadraticfit_vals = quadratic_fit(voltage,p,q,r)
plt.plot(voltage,quadraticfit_vals, label="Quadratic Fit")
plt.plot(voltage,linearfit_vals, label="Linear Fit")
plt.scatter(arr, degrees, label = "Manual Calibration")
plt.xlabel("Potentiometer Voltage (V)")
plt.ylabel("Angles (degrees)")
plt.title("Potentiometer Calibration Plot -Extrapolated For Negative Angles")
plt.legend()
plt.show()

############################################################
#Angles greater than 150 degrees
arr2 = np.array([])
for i in range(150,180,10):
    file_name = "data/Deg"+str(i)+".csv"
    data = getData(file_name)
    v_data = data["Voltage"]-v_data_offset
    avg = np.mean(v_data)
    arr2 = np.append(arr2,avg)
degrees2 = np.arange(150,180,10)
#linear fit specific to greater than 150
l_150 = curve_fit(linear_fit, arr2, degrees2)
m_2 = l_150[0][0]
n_2 = l_150[0][1]
coef_150 = [m_2,n_2]
volt_150 = np.linspace(0.055,0.07)
lin_150  = linear_fit(volt_150,coef_150[0],coef_150[1])
plt.scatter(arr2,degrees2, label='manual calibration')
plt.plot(volt_150,lin_150, label='linear fit')
plt.xlabel('Potentiometer Voltage (V)')
plt.ylabel('Angle (degrees)')
plt.title("Calibration Plot - Greater than 150 Degrees")
plt.legend()
plt.show()

def angles_data_elementwise(v_data, coef_150, coef):
    mask = (-0.045 <= v_data) & (v_data <= 0.045)
    output = np.empty_like(v_data, dtype=float)
    output[mask] = linear_fit(v_data[mask], coef[0], coef[1])
    output[~mask] = linear_fit(v_data[~mask], coef_150[0], coef_150[1])

    return output


def process_and_plot(start,finis,file_name):

    experiment = getData(file_name)
    v_offset = experiment["AveVoltage"][finis]
    t_offset = experiment["Time"][start]

    # Create data vectors
    t_data = np.array(experiment["Time"][start:finis]) - t_offset
    v_data = np.array(experiment["AveVoltage"][start:finis]) - v_offset


    angles_data = angles_data_elementwise(v_data, coef_150, coef)


    popt, _ = curve_fit(theta_solution, t_data, angles_data)
    a, b, c, d = popt
    theta_model_vals = theta_solution(t_data, a, b, c, d)


    labelstring = f"A={a:.4f},B={b:.4f},C={c:.4f},D={d:.4f}"
    titlestring = r"$\theta(t) = A e^{Bt} \cos(Ct -D)$"

    peaks, properties = find_peaks(angles_data, height=None, distance=None)
    peak_times = np.array(t_data)[peaks]
    periods = np.diff(peak_times)
    avg_period = np.mean(periods) if periods.size > 0 else np.nan
    frequency = 1 / avg_period if avg_period > 0 else np.nan

    # Plot the data and the fitted model
    plt.plot(t_data, angles_data, marker='.', markersize=1, label='data')
    plt.xlabel("Time (s)")
    plt.ylabel("Angle (degrees)")
    plt.plot(t_data, theta_model_vals, label=labelstring)
    plt.title(titlestring)
    plt.legend()
    plt.show()
    return np.array([a,b,c,d])

case_1 = process_and_plot(250,3000, 'data/Dynamic/case1.csv')
case_2 = process_and_plot(680,9000, 'data/Dynamic/case2.csv')
case_3 = process_and_plot(500,10000, 'data/Dynamic/case3.csv')
case_4 = process_and_plot(650,12000, 'data/Dynamic/case4.csv')
case_5 = process_and_plot(800,12500, 'data/Dynamic/case5.csv')
case_6 = process_and_plot(3000,14999, 'data/Dynamic/case6.csv')

cases = np.vstack([case_1, case_2, case_3, case_4, case_5, case_6])

# Parameter names corresponding to indices 0,1,2,3
param_names = ['A', 'B', 'C', 'D']
n_cases = cases.shape[0]

# Create a plot for parameter A (index 0)
plt.figure(figsize=(8, 6))
plt.bar(np.arange(n_cases), cases[:, 0])
plt.title("Parameter A")
plt.xlabel("Case")
plt.ylabel("A")
plt.xticks(np.arange(n_cases), [f"Case {j+1}" for j in range(n_cases)])
plt.grid(True)
plt.show()

# Create a plot for parameter B (index 1)
plt.figure(figsize=(8, 6))
plt.bar(np.arange(n_cases), cases[:, 1], color='skyblue')
plt.title("Parameter B")
plt.xlabel("Case")
plt.ylabel("B")
plt.xticks(np.arange(n_cases), [f"Case {j+1}" for j in range(n_cases)])
plt.grid(True)
plt.show()

# Create a plot for parameter C (index 2)
plt.figure(figsize=(8, 6))
plt.bar(np.arange(n_cases), cases[:, 2], color='orange')
plt.title("Parameter C")
plt.xlabel("Case")
plt.ylabel("C")
plt.xticks(np.arange(n_cases), [f"Case {j+1}" for j in range(n_cases)])
plt.grid(True)
plt.show()

# Create a plot for parameter D (index 3)
plt.figure(figsize=(8, 6))
plt.bar(np.arange(n_cases), cases[:, 3], color='purple')
plt.title("Parameter D")
plt.xlabel("Case")
plt.ylabel("D")
plt.xticks(np.arange(n_cases), [f"Case {j+1}" for j in range(n_cases)])
plt.grid(True)
plt.show()
```