

ME 3300 Experiment 02:

Potentiometer Data Acquisition and Pendulum Calibration

1. LABORATORY OBJECTIVES

Your objectives for this laboratory session are to:

- 1) Become familiar with MATLAB's data acquisition toolbox.
- 2) Create a simple program for data acquisition using MATLAB and SIMULINK.
- 3) Gain hands-on experience with breadboards, digital power supply, digital multi-meter, and laboratory tools.
- 4) Calibrate a potentiometer angle measurement system and create an angle vs. volts calibration plot.
- 5) Record and plot angle vs. time data for the swing motion of the pendulum system.
- 6) Compare your pendulum's recorded motion to the theoretical motion for a simple pendulum.

2. EQUIPMENT LIST

- | | |
|---|-------------------------------|
| 1) USB Data Acquisition Device (NI-MyDAQ) | 5) Small flathead screwdriver |
| 2) Breadboard (MyDigital protoboard) | 6) Digital multimeter (DMM) |
| 3) Pendulum with Potentiometer (10k Ω Bourns 6639s) | 7) Protractor |
| 4) Needle-nose pliers & wire strippers | 8) Power supply and wire kit |

3. INTRODUCTION

For this experiment, you will connect a simple angle sensor (angular potentiometer) to a data-acquisition (DAQ) device and collect digital data using MATLAB's data acquisition toolbox and a Simulink graphical program. You will collect data to create a calibration equation to convert the output voltage from the angular potentiometer to the pendulum angle. You will then validate your calibration by comparing the results of a free swing experiment to a theoretical prediction using a simple pendulum model solved using MATLAB's ODE45 solver.

3.1 Pendulum Model

An ideal planar pendulum consists of a body suspended from a light rod of length l pinned to a frictionless pivot, which allows it to swing back and forth under the influence of gravity. The dynamics of a pendulum can be simplified significantly by making the following assumptions:

- 1) The mass of the pendulum is concentrated at the swinging end.
- 2) The pendulum rod is rigid and massless.

These assumptions result in what is called the model of a *simple pendulum*. They will limit the accuracy of the model, but how much error will be introduced? We can use experimental measurements to compare.

To derive the differential equation of motion for the simple pendulum, refer to Figure 1.

The torque about the pivot point P is

$$\bar{\tau}_p = \bar{\mathbf{r}}_{p,m} \times m\mathbf{g} + \tau_f = L\hat{\mathbf{r}} \times m\mathbf{g}(\cos\theta\hat{\mathbf{r}} - \sin\theta\hat{\boldsymbol{\theta}}) - b\dot{\theta}\hat{\mathbf{k}} = -Lmg\sin\theta\hat{\mathbf{k}} - b\dot{\theta}\hat{\mathbf{k}}. \quad (1)$$

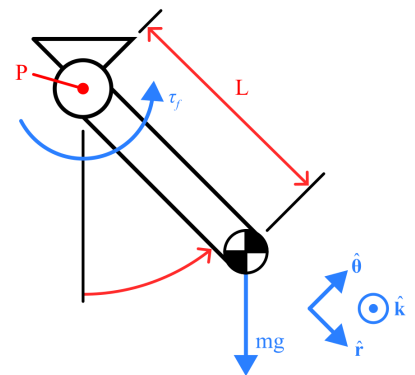


Figure 1: Coordinate System and Torque diagram for a simple pendulum.

So the z-component of torque about the pivot P may be written as

$$(\tau_p)_z = -mgL \sin \theta - b\dot{\theta}. \quad (2)$$

Applying the parallel axis theorem, the moment of inertia of a point mass about the pivot is $I_p = ml^2$ so the rotational equation of motion is found as

$$\begin{aligned} (\tau_p)_z &= I_p \alpha_z = I_p \ddot{\theta} \\ -mgL \sin \theta - b\dot{\theta} &= mL^2 \ddot{\theta}. \end{aligned} \quad (3)$$

One can rearrange to find the standard form as a second-order nonlinear ODE

$$\ddot{\theta} + b\dot{\theta} + \frac{g}{L} \sin \theta = 0. \quad (4)$$

This solution can also be represented in terms of the damping ratio, ζ , and natural frequency, ω_n , as follows

$$\begin{aligned} \ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2 \sin(\theta) &= 0 \\ \text{where:} & \\ \zeta = \frac{b}{2mL^2\omega_n}, \quad \omega_n &= \sqrt{\frac{g}{L}}. \end{aligned} \quad (5)$$

MATLAB's ODE 45 can be used to simulate the response of this system, given estimates of the parameters m, b, g and L as well as the initial condition (IC) for θ . [Pendulum_Simulation.m](#) is a sample script showing how to implement a simulation of this pendulum model.

3.2 Data Acquisition with MATLAB's Data Acquisition Toolbox

Throughout the semester, we will use MATLAB's Data Acquisition Toolbox for data collection. This package integrates DAQs with MATLAB and Simulink. Simulink provides graphical programming, which is helpful for prototyping. Text-based programming with MATLAB can also be used for more detailed control, depending on the application; however, for most labs, we will use Simulink. If you are interested in the text programming approach, an example MATLAB script, [MATLAB_Data_Aquisition.m](#), is available on the class GitHub repository. The following instructions will guide you through building and using your own Simulink data acquisition program.

4. LABORATORY INSTRUCTIONS

In this experiment, you will connect a simple angle sensor (angular potentiometer) to the NI USB-6008 data acquisition device and collect digital data using a Simulink graphical program. Then, you will collect data to create a calibration equation to convert the output voltage from the angular potentiometer to the pendulum angle. Finally, you will apply the calibration data to collect a recording of the pendulum swinging under the influence of gravity and compare the results to a model pendulum.


4.1 Part 1: Create a Simulink Program to Display and Record Data.

1. Launch Simulink

- Start MATLAB and enter **simulink** in the Command Window.
- In the Simulink start page, click **Blank Model** to open a new model.

2. Build a Basic Data Acquisition Program

You'll create a simple program to acquire and visualize voltage data from the potentiometer (see Figure 2).

 **Tip:** You can add blocks by using the Library Browser or by double-clicking on the canvas and typing the block name in the search bar.

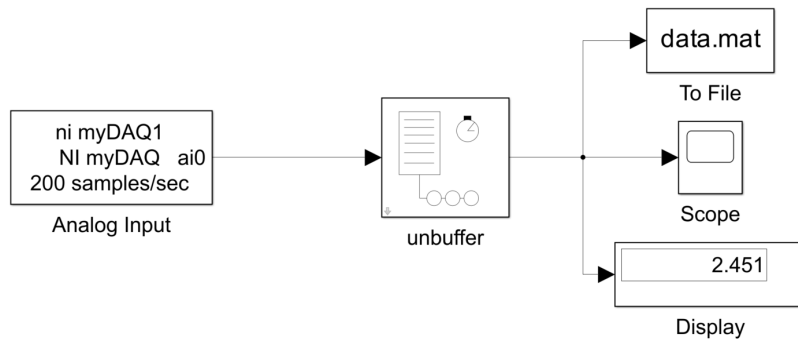


Figure 2. Simulink Model file for data acquisition.

Required Blocks and Configuration:

a. Analog Input Block (from Data Acquisition Toolbox)

- This block interfaces the NI MyDAQ and reads the analog input. Double-click the block to access its parameters.
- Set **Device** to your NI myDAQ.
- Select **Asynchronous** Acquisition Mode (this lets Simulink run to run while the DAQ collects data).
- Check the box to enable only analog input channel 0 (ai0).
- Set the **Measurement Type** to **Differential**.
- Set the **Input Range** on channel 0 to **-10 V to +10 V**.
- Set **Coupling** to on channel 0 to **DC**.
- Set **Number of ports** to **1 per channel**.
- Set **Input sample rate** to **200 samples/sec**.
- Set **Block size** to **10** (this sets the number of samples Simulink will read from the DAQ in a single “frame”. Smaller values allow quicker plot updates but increase CPU load.)
- Click **Apply**, then close the settings window.


b. Unbuffer Block (From the CustomBlocks.slx provided via GitHub).

This block converts the frame-based data from the ADC into a serial stream.

- Double-click the **Unbuffer** block to access its parameters.
- Set **Block Size** and **Sample Rate** to match the **Analog Input** block (10 and 200, respectively).

c. Scope Block (From Simulink\Sinks)

Used to plot the incoming voltage signal.

- Double-click on the **Scope** and click the  (gear) icon in the top bar to configure:

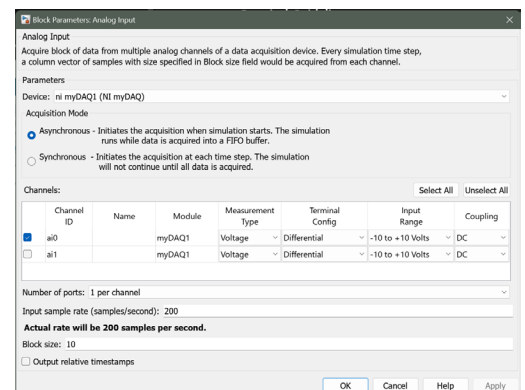


Figure 3: Analog Input Parameter Settings.

- **Main Tab:** check *Open at simulation start* and set *Axes scaling* to **Auto**. This will open the plot when the simulation runs and automatically scales the x and y axes to fit the acquired data.
- **Time Tab:** Set *Time units* to **seconds** and check *Show time-axis label*.
- **Display Tab:** Check *Show Legend*.
- **Logging Tab:** Make sure **no** options are checked. (Note that this tab allows logging to MATLAB, but we will use a separate block for that.)
- Click **Apply** and close the scope settings window.
- *Scopes offer additional formatting and logging tools. Feel free to explore as you gain familiarity with Simulink.*
- d. **Display Block** (From **Simulink\Sinks**)
Used to show the current value of the incoming signal numerically during simulation.
- e. **To File Block** (From **Simulink\Sinks**)
Used to save collected data to **.mat** files for later analysis.
 - Set the **File name** to something short and descriptive (e.g., **Angle_m45Deg.mat**).
⚠ Always change the file name between runs to avoid overwriting old data!
 - Set **Save format** to **Array**.
 - Leave **Decimation** at **1** and **Sample time** as **-1**.
- 3. **Save Your Model**
 - Save your Simulink Model with a descriptive name.
e.g. **ME_3300_Pendulum_Experiment_Part1_Lab_01.slx**
 Your program is now ready to acquire data.

4.2 Part 2: Build Angular Potentiometer Circuit and Connect to DAQ

Next, we will wire connect the potentiometer to power and the DAQ for data acquisition.

1. Connect the Potentiometer to Power

- Connect the leads labeled **+5V** / $\frac{1}{\text{ground}}$ to the 5-Volt reference signal and the reference ground.
- Refer to Figure 4 for wiring layout.

2. Connect Signal to NI MyDAQ

- Connect potentiometer leads labeled **Signal**/ $\frac{1}{\text{ground}}$ as follows:
 - **Red wire** → AI0+ (positive analog input)
 - **Black wire** → AI0- (negative analog input)
- See Figure 4 to confirm the correct pin connections.

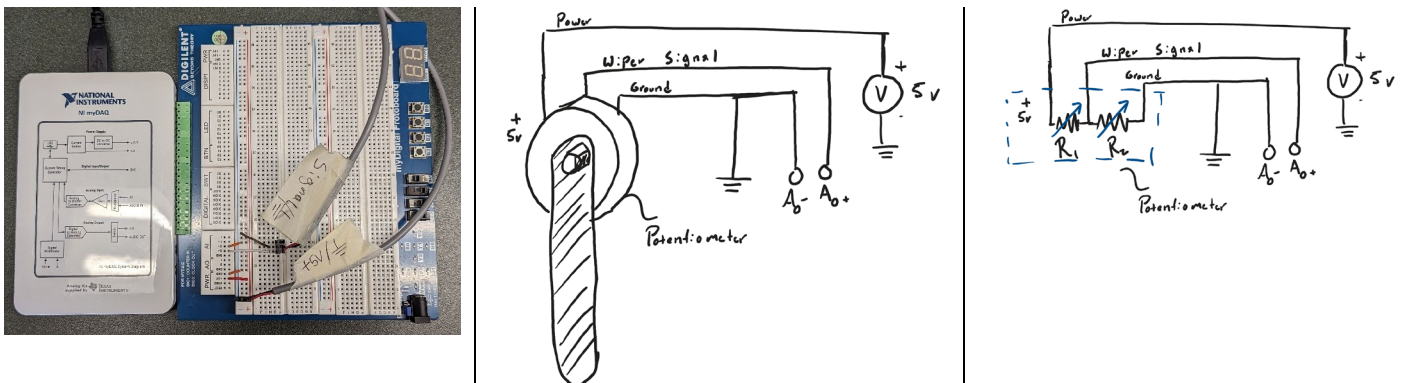


Figure 4. Circuit built on NIMyDAQ, LEFT, Potentiometer Circuit Diagram, CENTER. This setup forms a voltage divider where R_1 and R_2 vary with wiper angle, RIGHT.

4.3 Part 3: Use Simulink Program to Test Circuit and Test Data Acquisition

To verify that your circuit is implemented correctly, test it by rotating the potentiometer and confirming the voltage changes as expected at each position.

1. Adjust Simulation Duration

- Change the “**Stop Time**” from its default (10 seconds) to a value that fits your testing needs:
 - Use a **short duration** for quick checks
 - Use a **longer duration** or **inf** (infinite) if you want the simulation to run continuously until you manually stop it with the **Stop Simulation** button (Figure 5).

2. Run the Simulation

- Start your Simulink program by clicking the **Start Simulation** button arrow (Figure 5).

3. Observe Output While Rotating the Pendulum

- While the simulation is running:
 - **Rotate** the angular potentiometer
 - **Observe** the real-time output on the **Scope** and **Display** blocks

4. Verify Smooth Voltage Response

- Ensure the voltage changes **smoothly** over the $\pm 90^\circ$ range.
- If you notice a sudden jump or drop in voltage:
 - The potentiometer may be **misaligned**.
 - To fix it, loosen the grub screw, rotate the potentiometer until the voltage varies smoothly (jump should occur around $+180^\circ$), then re-tighten it.
 - Re-test to confirm smooth output over the test range.

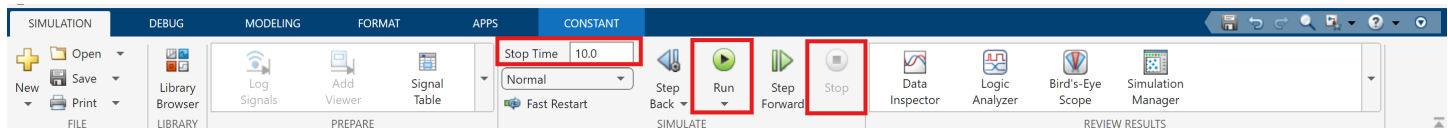


Figure 5. Locations of simulation time, simulation run, and simulation stop buttons in the Simulink interface.

4.4 Part 4: Create Calibration Equation to Output Pendulum Angle

Next, you will perform a calibration with **10 or more angles**. At each angle, you'll hold the pendulum steady and collect a short data set. You will repeat this process to sweep the pendulum through the full $\pm 90^\circ$ range incrementally.

1. Choose a Repeatable Angle Measurement Method

Use a reliable method to measure pendulum angle. Options include:

- The supplied protractor
- A smartphone app like [Bubble Level](#).

2. Configure your Simulink Model

Set your model to record:

- Duration: 5 seconds
- Sample Rate: 20 samples per second

This short time & coarse sample rate is appropriate for static measurements and helps reduce file size.

3. Create a table in your logbook

Use the format shown in Table 1 to log your measurements.

4. Record data at each angle:

Repeat this step repeatedly while incrementing through each angular position. Each run:

a. Run your Simulink Model:

- Save each data file using the "to_file" block.
- Use this name convention: **Ang_(p/m)XXDeg.mat**
 - **XX** indicates angle.
 - **p/m** indicates sign (positive or negative).
 - For example: for -45° , use **Ang_m45Deg.mat**.
- **Update the file name before each run** to avoid overwriting data.

b. Record the angular position (in degrees):

- Use the **right-hand rule**: point your right thumb out from the potentiometer shaft (see Figure 6) to determine the positive direction.
- Zero degrees is defined as the pendulum pointing straight down.

c. Compute the average voltage:

- Write a MATLAB Script to calculate the mean voltage from each data file
- Record the result with at **least two decimal places**.

Table 1 Sample calibration data for experiment 2.

Pendulum Angle (Degrees)	Average Output Voltage (V)
-90°	1.21
-70°	1.43
-50°	1.65
-30°	2.00
-10°	2.31
0°	2.50
10°	2.55
30°	2.82
50°	3.20
70°	3.51
90°	3.74

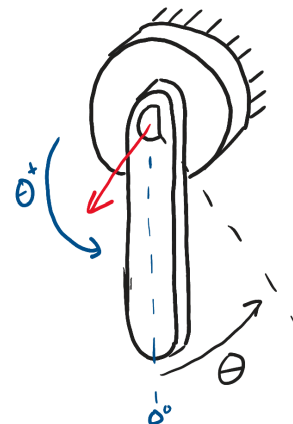


Figure 6. Right hand rule and pendulum zero position.

5. Plot Your Calibration Data

- Use MATLAB to create a scatter plot:
 - X-axis: average voltage
 - Y-axis: angle in degrees
- Format the plot clearly and include appropriate axis labels, title, and units

Refer back to formatting guidelines from **Lab 01** and the requirements below under **Post Laboratory Assignment**.

6. Fit a Linear Calibration Model

Use the **polyfit** command in MATLAB to compute a linear regression of the form: $y = a_0 + a_1x$.

7. Ensure Fit Quality

- Record the **norm of the residuals** from **polyfit**.
- Use this value to calculate the **standard error of the fit**, S_{yx} .
- What do these values tell you about the fit? (Answer briefly in your lab logbook).

8. Create and Save Calibration Plot

- Refer to the sample code in the appendix to create a calibration plot.
- Update the **x** and **y** variable values to match your data.
- Save your plot with **at least 600 dpi resolution**.

4.5 Part 5: Add Calibration Equation to Simulink and Prepare for Swing Experiment

Next, we will modify your data acquisition program to **convert the raw voltage into pendulum angle (in degrees)** using your calibration equation.

1) Implement the Calibration Equation

You will add blocks to your Simulink program to use the calibration equation (See Figure 7). There are several ways to do this, but the following Blocks are necessary for a common approach:

- Two **Constant** blocks (from **Sources**)
 - One stores the **calibration slope** and the other stores the **bias**.
- **Product** block (from **Math Operations**)
 - Multiplies acquired potentiometer voltage by the calibration slope.
- **Add** Block (from the **Math Operations**)
 - Adds the calibration bias to the output on the multiplication.

2) Append Timestamps to the Output

- Add a **Clock** (from Sources) and **Mux** (from Signal Routing)
- Wire to append sample times to the recorded data (See Figure 7).

3) Update Output File Settings

- Change the output **file name** to “**AngVsTime.mat**”.

4) Save a New Version of the Simulink Program

- Save your updated data acquisition program with a new, descriptive file name

Keeping a copy of each Simulink/MATLAB program you create helps with debugging and for future reference.

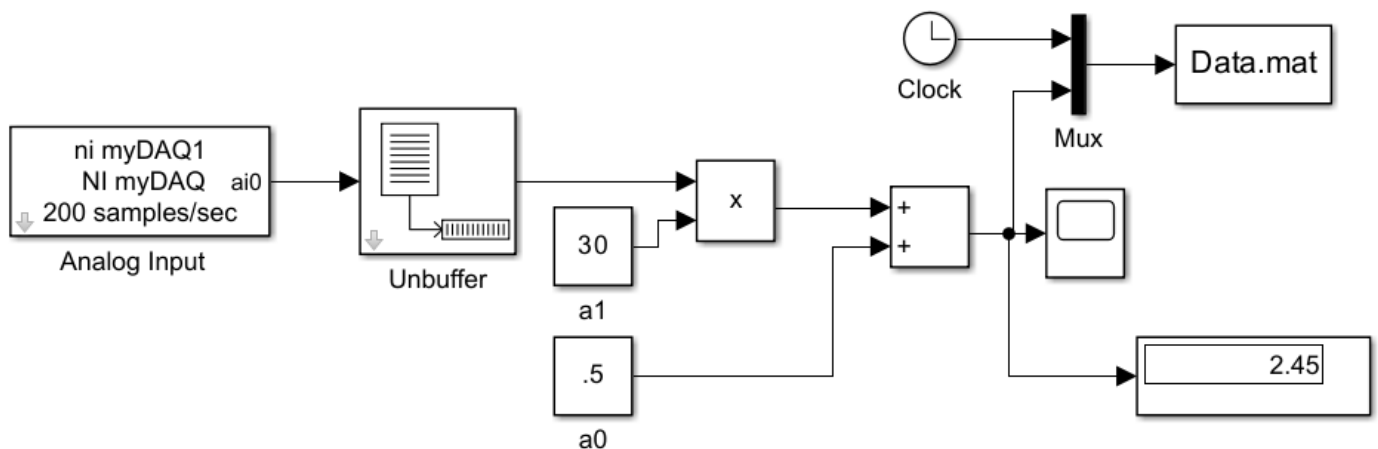


Figure 7. Example Simulink program including calibration and sample time logging.

4.6 Part 6: Acquire Calibrated Pendulum Swing Data

In this step, you will record and plot the pendulum's free oscillation using your calibrated Simulink program.

1) Set an Appropriate Recording Duration

- Choose a **simulation time** long enough to capture the transient response (the full oscillation time) of the pendulum.
- Start with **at least 15 seconds**, and adjust as needed to allow the pendulum to settle during your recording.

2) Run the Simulation and Release the Pendulum


- Start your Simulink program and release the pendulum from a horizontal position ($+90^\circ$), recording the angle (in degrees from your calibration equation) vs. time.

3) Analyse the Oscillation in MATLAB

- Load your recorded **.mat** file in MATLAB.
- Use the angle data to determine:
 - The **oscillation period**, T_d
 - The corresponding **damped oscillation frequency**, ω_d .

4) Plot Angle vs. Time

- Use MATLAB to plot the pendulum angle vs. time.
- Use the following formatting:
 - **Line color**: solid red
 - **Line width** line with a 2 pt
- **Trim the data** so that the plot starts at the moment the pendulum is released.
 - Set this point as **time = 0**

 You can use your plot to identify the oscillation period.

4.7 Part 7: Overlay a model estimate of pendulum oscillation and estimate pendulum parameters.

For this part, you will match a model pendulum simulation to your recorded swing data.

1. Open the provided MATLAB Script

- Refer to the **Pendulum_Simulation.m** for this part.
- This script uses **ode45** to simulate a model pendulum's motion

2. Set Pendulum Length

- Measure your pendulum's length using a **ruler or tape measure**.

3. Set Initial Parameter Estimates

- Without measuring directly, we don't have direct knowledge of the values of mass or the damping coefficient. For now, start with a reasonable guess of $m = 0.05$ kg and $b = 0.005$ Nms/rad.

4. Simulate Response with ODE45


- Use ODE 45 to simulate a model version of the response using these values.

5. Overlay the Model on your Pendulum Swing Plot

- On the previous pendulum swing plot, overlay the simulation result:
 - Use a blue dashed line with a 2 pt. linewidth.

6. Refine Model Parameters

- Iteratively adjust parameter values for **m** and **b** to bring the model into close alignment with your experimental data.
- You may need to adjust your model's initial conditions as well if your recording didn't start with the pendulum at exactly $+90^\circ$.

 Note: It is normal for the model and data to show some differences. Aim for approximate alignment, not perfection.

4.8 Part 8: Return lab space to prior condition.

- 1) Remove all breadboard wires and place them back in the wire kit in an organized fashion.
- 2) Remove the pendulum wires from the breadboard and set aside.
- 3) Confirm your data files are saved to OneDrive.
 - a. Check that files are available online using your phone or personal Laptop to confirm.
 - b. Make sure that all team members have access to experiment files, including data and scripts.
- 4) Log off the computer.

5. POST-LABORATORY ASSIGNMENT

For this experiment, submit the following items as pdf files on Canvas:

1. Calibration Plot.

- The calibration plot should include:
 - Raw data points as **red circles** markers with `markersize = 9`
 - A solid blue line showing the **linear calibration curve** with a 2 pt. linewidth.
 - Dashed black lines showing the 95th CI (you can compute this using the standard error of the fit S_{yx})
- The plot should be 5" tall and 6.5" wide.
- Make sure to properly annotate your plot: axis labels, titles, legend, etc.
 - Use title: "FirstName LastName's Calibration Plot"
- Set axis grid lines **on**, box **off**, and figure background color to **white**.
- Using the **text** command, place the equations or numbers listed below on the plot. Use Greek symbols where appropriate.
 - The **calibration equation** on the plot has 4 decimal places for each number. Also include the norm of the residuals and the standard error of the fit. Make sure to include units.
 - The **norm of the residuals** of your linear regression.
 - The **standard error of the fit** for your calibration equation.

2. Pendulum Swing Plot.

- The Pendulum Swing plot is the time series plot showing the trajectory of your pendulum swinging from a horizontal starting position. It should include:
 - Experimental data plotted with a solid red line with 2 pt. linewidth.
 - Your best-fitting model as a dashed blue line with a 1 pt. linewidth.
- The plot should be 5" tall and 6.5" wide.
- Set axis grid lines **on**, box **off**, and figure background color to **white**.
- Make sure to properly annotate your plots: axis labels, titles, legend, etc.
 - Use title: "FirstName LastName's Pendulum Swing Plot"
- Using the text command in MATLAB, place the equations or numbers listed below on the plot. Use Greek symbols where appropriate.
 - The period of the oscillations, T_d .
 - The frequency of the oscillations, ω_d .

3. Answer all questions in the post-lab assessment.

- Compare your plots to the example plots shown below to confirm that you have completed the assignment correctly. Note that your values will differ, but your formatting and annotation should match.

