# Experiment-02: Potentiometer Data Acquisition and Pendulum Calibration

**ME-330**

Dr. Vibhav Durgesh

vdurgesh@uidaho.edu

Week-02

## Contents

## 1 Learning objective

The objectives of this experiment are to provide the student with an opportunity to: 1) Gain additional experience with Matlab. 2) Create a simple program for data acquisition using Matlab and National Instrument's USB-6001 or NI-MyDAQ. 3) Become familiar with breadboards, digital power supply, wiring, digital multi-meters, and other laboratory skills. 4) Calibrate a potentiometer-based angle measurement system and create a calibration plot. 5) Record and plot angle vs. time data for a swing motion of the pendulum system.

- Gain additional experience with Matlab data acquistion toolbox.
- Create a simple program for data acquisition using Matlab and National Instrument's USB-6001.
- Become familiar with breadboards, digital power supply, wiring, digital multi-meters, and other laboratory skills.
- Calibrate a potentiometer based angle measurement system and create a calibration plot.
- Learn more about pendulum system and
- Plotting data plot angle vs. time data for a swing motion of the pendulum system.

## 2 Introduction

In this experiment, you will connect a simple angle sensor (angular potentiometer) to the NI USB-6001 data acquisition device and collect digital data using a Matlab data acquisition toolbox. First you will collect data to create a calibration equation, then use the information from the calibration to convert the output voltage from the angular potentiometer to the pendulum angle.

**It is important that you read this entire document before the experiment. If you have question about the procedure, instruction provided, and the code then please stop by my office or talk to the TAs before Thursday.**

## 3 Part-A: Creating codes

*Creating data acquisition program using Matlab data acquisition toolbox*
The Matlab program is provided in the appendix section of the instruction. The main program sets the parameter for the acquisition and function file plots and record the data as data being acquired from the data acquisition device.

## 4 Part-B: Building circuit

*Connect an angular potentiometer to the NI My-DAQ using a breadborad.*

- Connect the potentiometer leads labeled "Power and Ground" to the 5 Volt reference signal and the reference ground from the power supply. See figure 1
- Connect the NI MyDAQ to the angular potentiometer leads labeled "Signal and Ground". The red wire should connect to AI0+ and the black wire to AI0- as shown in figure . See figure 1.

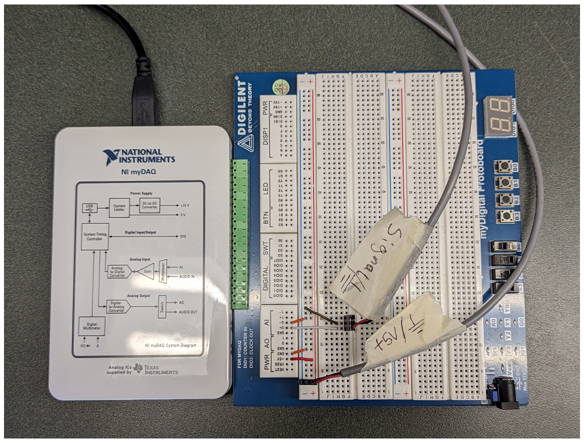
Figure 1: Connection to NI-DAQ system for experiment-2.

## 5 Part-C: Testing code and circuit

*Run your acquisition program and testing your connections.*

- Verify your circuit with the TAs
- Once your circuit is verified, run your acquisition program.
- Rotate the angular potentiometer and verify the results. The display should change with the movement of pendulum.
- Keep in mind that DAQ system takes a while to initialize. There may be a slight delay before data acquisition start.
- Use the Matlab program provided for testing your circuit connections. See section 11.1.

## 6 Part-D: Perform calibration

*Create a calibration equation for the potentiometer to output the angle of the pendulum.*

- Determine a repeatable method for measuring each angle with the supplied protractor (or your phone).
- You will need to perform calibration for 10 or more angles.
- For the calibration you will use the Matlab provided in the appendix 11.2.
- At each of 10 (or more) angles, ranging from $\pm 90$ degrees, collect the voltage at a sampling rate of 20 Hz for 5 seconds. Save each of these files with the following name Ang_(p/m)XXDeg.dat where xx represent angle and p/m represent plus or minus sign. For instance, the calibration file for $-45^o$ will be saved as "Ang_m45Deg.dat" and the calibration file for $+60^o$ will be saved as "Ang_p60Deg.dat".
- Repeat the above step for all the calibration angles.
  - Make sure to have the angle (in degrees).
  - **The zero degree angle is when the pendulum is straight down.**
- Create a table in log book similar to the table 1. You will use the values in the table for calibration.
- Plot the data collected calibration data using Matlab. The x-axis should be the recorded average voltages and the y-axis should be the angle in degrees. Be sure to properly annotate your plot. **Use the instruction from lab-01**
- Use Matlab's polyfit tool to determine a linear fit to the calibration data. This is your calibration equation. **Make sure to save your data in case you need to repeat any analysis**.
- Record the norm of the residuals from the linear regression provided by Matlab's polyfit function. Use this to calculate the standard error of the fit ($s_{yx}$).
- See the sample code in appendix to generate the calibration plot 11.3.
- You need to modify 'x' and 'y' variables values in the code based on your calibration data.
- Make sure to save the calibration file with at least 600 dpi resolution.

## 7 Part-E: Acquire pendulum data

*Now add your calibration equation to acquisition program to display the pendulum angle in degrees during the data acquisition process.*

- You will have to change the slope and intercept values in the provided Matlab program in the section 11.4.
- Show your program and results to the TAs to verify your program.

Table 1: Sample calibration data for experiment 2.

| Pendulum angle (degree) | 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 |

- Change the acquisition time in the Matlab program to 15 seconds at a sampling rate of 200 Hz.
- You will need to save the data for this part. Change the output file name to "AngVs-Time.dat"
- Make sure your program is modified to acquire and store the data file.
- Run your Matlab program to and store the time and angle data.
- Execute the program and then release the pendulum from a horizontal position, as soon as the acquisition initiates.
- If everything is done correctly, the program should display the angle (in degrees from your calibration equation) vs. time.
- Confirm the file is stored correctly.
- Open the stored data file and determine the period, $T_d$, of the response oscillations and the corresponding frequency, $\omega_d$.

## 8 Part-F: Lab space clean up

Return lab space to prior condition

- Turn off the power supply.
- Remove wires connecting the power supply to the breadboard and return to the wall
- Remove all breadboard wires and place them back in the wire kit in an organized fashion.
- Remove the pendulum wires from the breadboard and set aside.
- Log off the computer.
- **Do this for every lab!**

## 9 Instruction

- Make sure to name the files your FirstName_ LastName_ QuestionNumber, use the format provided in the sample Matlab code. Make sure the submitted figure has the title. Include the apostrophe!
- Make sure to submit your post-lab assignment on the Canvas website.
- Here are some details to help.
  - Plot the experimental data using red circles with MarkerSize 8. Experimental data points should always be plotted using markers without lines connecting the points
  - Plot curve from theory (or curve fit line) using a solid colored line with LineWidth 2.
  - Set the x and y-axis limits for each figures.
  - Make sure the major grid lines are visible
  - All plot text should be in Times font. You will need to specify this for the title, labels, and legends. The axis labels and numbers should be in 10-point font
  - The title should be in 10-point font
  - Make sure to add the legend. Keep in mind that legend should not hide the plotted data
  - For full credit make sure that your submitted files look similar to the sample figures shown in the document
  - If in doubt make sure to request the TAs for help

## 10 Post lab (50 points)

**Postlab are due at Friday 5 pm.** For this experiment submit the following items on Canvas for your post-laboratory assessment.

1. Calibration plot. Submit a calibration plot showing the raw data points as markers and the calibration curve as a line. The plot should be 6.5" wide. Make sure to properly annotate your plots: axis labels, titles, legend, etc. Using the text command in Matlab, place the equations or numbers listed below on the plot. Use Greek symbols where appropriate. Make sure to include units.
   - The calibration equation on the plot with 4 decimals places for each number.
   - The norm of the residuals of your linear regression.
   - The standard error of the fit for your calibration equation.
2. Pendulum Swing Plot. Submit a time series plot showing the trajectory of your pendulum swinging from a horizontal starting position. The plot should be 6.5" wide. Make sure to properly annotate your plots: axis labels, titles, legend, etc. 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, $\omega_d$.
3. Answer all question in the post-lab assessment on Canvas

## 11 Appendix

### 11.1 Matlab program to test the circuit

```
% This is a generic program to aquire the data from myDAQ
% THis can be used to test circuit and DAQ is working properly
% Sample data acquisition program
% Date: August 5th, 2020
% Dr. Vibhav Durgesh
% Rev 0.0
% User has to provide appropriate information - see beginning of code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS CODE OVERWRITES THE DATA. PLEASE MOVE THE FILES OR RENAME PRIOR TO
% RERUNNING THE CODE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
close all
clc
%% ----- Start user required information ----- %%%%%%%%%%%%%%%%%%%
Fs = 200; %Sampling rate data per sec
T = 10; %Time to aquire data
%% Here change the device name and input channel as needed; if not sure then talk to TAs
deviceName = 'myDAQ1';
inputChannel = 'ai0';
%% ----- End user required information ----- %%%%%%%%%%%%%%%%%%%%
d = daq.getDevices;
s = daq.createSession('ni');
addAnalogInputChannel(s,deviceName,[inputChannel],'Voltage');
s.Rate = Fs; %Sample rate modaify as required
s.DurationInSeconds = T; %Sampling time modify as required
lh = addlistener(s,'DataAvailable',@(src,event)plotAndLogData(src,event));
errorListener = addlistener(s, 'ErrorOccurred',@(src, event)plotAndLogData(src, event));
drawnow
s.IsContinuous = true;
startBackground(s);
pause(T)
delete(s)
delete(lh)

%% Plotting the voltage and time information
function plotAndLogData(src, event)
time = event.TimeStamps;
voltage = event.Data;%Potentiometer data in voltage
figure (1)
plot(time,voltage,'k.-');hold on
xlabel('time (s)')
ylabel('Voltage (v)')
end
```

## 11.2 Matlab program to acquire data for calibration

```
% This is a generic program to aquire the data from myDAQ
% Sample data acquisition program for calibration purpose it logs the data
% in a file for later use
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Date: August 5th, 2020
% Dr. Vibhav Durgesh
% Rev 0.0
% User has to provide appropriate information - see beginning of code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS CODE OVERWRITES THE DATA. PLEASE MOVE THE FILES OR RENAME PRIOR TO
% RERUNNING THE CODE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
close all
clc

%% Start of user information
Fs = 20; %Sampling rate data per sec
T = 5; %Time to aquire data
%% Here change the device name and input channel as needed; if not sure then talk to TAs
deviceName = 'myDAQ1';
inputChannel = 'ai0';
%% Change the filename where you will store the data
filename = '../Data/Ang_p90Deg.dat'; %To store calibration data file
fid1 = fopen(filename,'w');
%% End of user required information
d = daq.getDevices;
s = daq.createSession('ni');
addAnalogInputChannel(s,deviceName,[inputChannel],'Voltage');
s.Rate = Fs; %Sampling rate modify as required
s.DurationInSeconds = T; %Sampling time modify as required
lh1 = addlistener(s,'DataAvailable',@(src, event)plotAndLogData(src, event));
lh2 = addlistener(s,'DataAvailable',@(src, event)logData(src, event,fid1));
errorListener = addlistener(s, 'ErrorOccurred',@(src, event) disp(getReport(event.Error)));
drawnow
s.IsContinuous = true;
startBackground(s);
pause (T)
delete (s)
delete(lh1)
delete(lh2)
fclose(fid1)
data = readmatrix(filename);
avgVolt = mean(data(:,2));
gcf;
% Adding title with average values; you can note this value for later use
title(['Average voltage: ' sprintf('%3.4f',avgVolt) ' (v)'],'fontname','times','fontsize',14)

%% Function to plot the data as it is being acquired
function plotAndLogData(src,event)
time = event.TimeStamps;
voltage = event.Data; %Potentiometer data in voltage
figure(1)
plot(time, voltage,'k.-');hold on
xlabel('time (s)')
ylabel('Voltage (v)')
end

%% Function to store the data in a file as specidied by the user
function logData(src, evt, fid)
%Add the time stamp and the data values to data. To write data sequentially
%transpose the matrix
data = [evt.TimeStamps evt.Data];
fprintf(fid,'%f \t %f \n', data');
end
```

## 11.3 Matlab program to generate calibration plot

```
% This is a generic program to generate calibration plot and provide slope
% and intercept data; This code can be used for calibration for other
% experiments too.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Date: August 5th, 2020
% Dr. Vibhav Durgesh
% Rev 0.0
% This is a basic program to demonstrate the use of polyfit command and
% calculate norm and standard error of fit.
% User has to provide appropriate information - see beginning of code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
close all
clc

%% User have to manually add information- change x, y, and t_nuP values as required.
x = [3.8005 3.5127 3.2527 2.9704 2.6897 2.5767 2.465 2.206 1.9007 1.6289 1.3384]; %Voltage Data
y = [-90 -70 -50 -30 -10 0 10 30 50 70 90]; %Angular Location of Pendulum
t_nuP= 2.262; %Value read from t-student's table
plotTitle = 'Student''s Name calibration Plot';
%% End of user information
[p,s] = polyfit (x,y,1); % Curve fitting with 1st order fit
xfit = x;% Generating x-data for curve fitting
yfit = polyval(p,xfit);
nu = s.df; % Getting degree of freedom for the curve fit
norm = s.normr; % Getting the norm of the curve fitting
syx = norm/sqrt(nu);
%% Generating figure with specific size
figure(1)
set(gcf,'unit','inches','position',[0.50 0.50 6.50 3.50],...
    'defaultaxesfontsize',10,'defaultaxesfontname','times');
% Plotting data
plot(xfit,yfit,'b-','linewidth',2);hold on
plot(x,y,'ro','markersize',9,'markerfacecolor','r')
xlabel('Volts (v)')
ylabel('Angle (^{o})')
%% Adding 95% confidence level lines in the plot
y_cl_low = yfit - t_nuP*syx; % Using regression analysis
y_cl_up  = yfit + t_nuP*syx;
plot(xfit,y_cl_low,'--','color',[0.2 0.2 0.2], 'HandleVisibility','off');
plot(xfit,y_cl_up,'--','color',[0.2 0.2 0.2]);
%% Adding legend and title
legend('Curve fit','Expt. data','95% Cl range','location','Northeast')
title(plotTitle)
%% Adding text with relevant curvefitting information
text(3,10,sprintf('y = %3.4fx+%3.4f',p(1),p(2)),'Fontname','times')
text(3,0,sprintf('Norm = %3.4f',s.normr),'Fontname','times')
text(3,-10,sprintf('s_{yx} = %3.4f',syx),'Fontname','times')
%% Saving the files in png and pdf format
figName = ['../Figures/Student_Name_Exp02_Part1'];
set(gcf,'PaperPositionMode','auto')
print(figName,'-dpng','-r600')
set(gcf,'PaperUnits','inches','Units','inches');
figpos = get(gcf,'Position');
set(gcf,'Papersize',figpos(3:4),'Units','inches');
print(figName,'-dpdf','-r600')
```

## 11.4 Matlab program to acquire data

```
% This is a generic program to aquire the data from myDAQ
% Sample data acquisition program for collecting the data to capture
% pendulum motion file for later use
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Date: August 5th, 2020
% Dr. Vibhav Durgesh
% Rev 0.0
% User has to provide appropriate information - see beginning of code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS CODE OVERWRITES THE DATA. PLEASE MOVE THE FILES OR RENAME PRIOR TO
% RERUNNING THE CODE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
close all
clc

%% Start of user information
Fs = 200; %Sampling rate data per sec
T = 20; %Time to aquire data
%% Here change the device name and input channel as needed; if not sure then talk to TAs
deviceName = 'myDAQ1';
inputChannel = 'ai0';
%% Input the filename and directory where to store the data
filename = '../Data/Student_Name_AngVsTime.dat'; %To store calibration data file
fid1 = fopen(filename,'w');
%% Add the slope and intercept values to convert volts to physical varaible (angle)
slope = -73.915; %Add your calibration slope for potentiometer
intercept = 190.4438; %Add you calibration intercept for potentioment
plotTitle = 'Firstname LastName''s plot - Experiment#2';
fprintf(fid1,'%s \t %s \n','Times (s)','Angular Position (degrees)')
%% End of user information
d = daq.getDevices;
s = daq.createSession('ni');
addAnalogInputChannel(s,deviceName,[inputChannel],'Voltage');
s.Rate = Fs; %Sample rate modaify as required
s.DurationInSeconds = T; %Sampling time modify as required
lh1 = addlistener(s,'DataAvailable',@(src, event)plotAndLogData(src, event,slope,intercept));
lh2 = addlistener(s,'DataAvailable',@(src, event)logData(src, event,fid1,slope,intercept));
errorListener = addlistener(s, 'ErrorOccurred',@(src, event) disp(getReport(event.Error)));
drawnow
s.IsContinuous = true;
startBackground(s);
pause(T)
delete(s)
delete(lh1)
delete(lh2)
fclose(fid1)
title(plotTitle,'fontname','times','fontsize',14)
%% Function to plot the data as it is being acquired
function plotAndLogData(src,event,m,c)
time = event.TimeStamps;
voltage =  m*event.Data + c; %Potentiometer data in voltage
figure(1)
plot(time, voltage,'k.-');hold on
xlabel('time (s)')
ylabel('Angular position (degrees)')
end

%% Function to store the data in a file- values are converted to physical variable using slope
    and intercept
function logData(src,evt, fid,m,c)
%Add the time stamp and the data values to data. To write data sequentially
%transpose the matrix
data = [evt.TimeStamps m*evt.Data+c];
fprintf(fid,'%f \t %f \n', data');
```

```
end
```