UCI Samueli School of Engineering
Department of Mechanical and Aerospace Engineering

Experimental Verification #1

By Alex Amadeo-Ranch | MAE 106

Magnetometer Calibration

- Purpose:
  - Raw magnetometer readings are affected by sensor bias and axis scaling mismatch.
  - Calibration removes these systematic errors so magnetic field measurements map correctly to physical orientation.
- Setup:
  - Connected arduino to the LSM6DS33 and LIS3MDL Carrier (figure 1).
  - Spin magnetometer 360 degrees.
  - Arduino IDE programmed with Serial.ino by Pololu to collect data.
  - Saving data through CoolTerm.

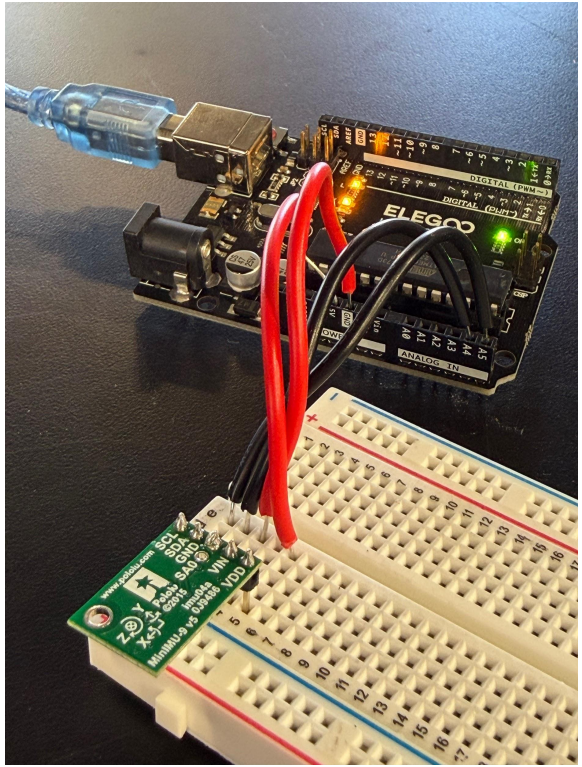

Figure 1: Magnetometer calibration circuit.

Magnetometer Calibration: Raw Data

- Raw Data
  - Arduino IDE reports text data to the serial monitor (figure 2).
  - Exported with CoolTerm as a .txt to be used with MATLAB.
  - Data extraction method used throughout this verification.
- Analyzing Data
  - Created a script in MATLAB to find offsets and scaling to calibrate the magnetometer (figure 3).
  - Plotted both uncalibrated data and calibrated data (next slide, figures 4 and 5).

```
M:  -2485   4123   -602
M:  -2491   4097   -655
M:  -2514   4117   -620
M:  -2469   4118   -604
M:  -2492   4113   -654
M:  -2483   4117   -578
M:  -2488   4119   -605
```

Figure 2: Magnetometer raw data in IDE serial monitor

a106verif1magCalib.m

C:\Users\alexa\OneDrive\Documents\MATLAB\a106verif1magCalib.m

```
% Load file
fname = "106verif1magRawData.txt";
lines = readlines(fname);

% Extract numbers using regex (robust)
tokens = regexp(lines, 'M:\s*(-?\d+)\s+(-?\d+)\s+(-?\d+)', 'tokens', 'once');
tokens = tokens(~cellfun('isempty', tokens));     % drop non-matching lines

A = vertcat(tokens{:});                           % Nx3 cell array of strings
data = str2double(A);                             % Nx3 double

Mx = data(:,1);
My = data(:,2);
Mz = data(:,3);

% raw data plot
figure
plot(Mx, My, '.', 'MarkerSize', 6)
axis equal
grid on
xlabel('M_x (raw)')
ylabel('M_y (raw)')
title('Raw Magnetometer Data (X vs Y)')

% Offsets
offset_x = (max(Mx) + min(Mx)) / 2;
offset_y = (max(My) + min(My)) / 2;

% Ranges
range_x = max(Mx) - min(Mx);
range_y = max(My) - min(My);

% Calibration
Mx_cal = (Mx - offset_x) / (range_x / 2);
My_cal = (My - offset_y) / (range_y / 2);

% calibrated data plot
figure
plot(Mx_cal, My_cal, '.', 'MarkerSize', 6)
axis equal
grid on
xlabel('M_x (calibrated)')
ylabel('M_y (calibrated)')
title('Calibrated Magnetometer Data')
```

Figure 3: MATLAB script to to plot magnetometer raw data

Magnetometer Calibration: Calibrated + Raw Data Plots

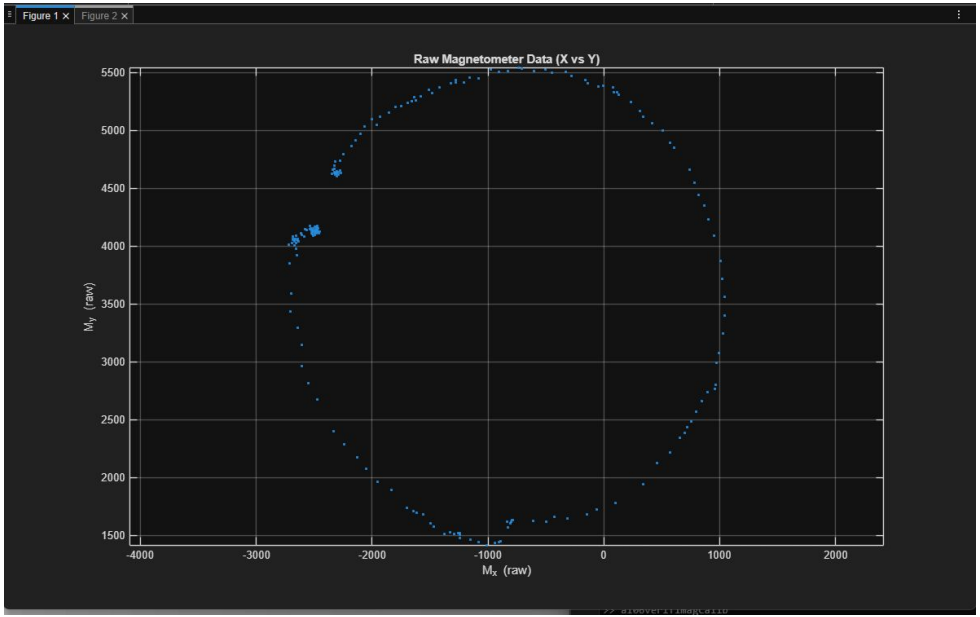

Figure 4: Magnetometer raw data plotted in MATLAB

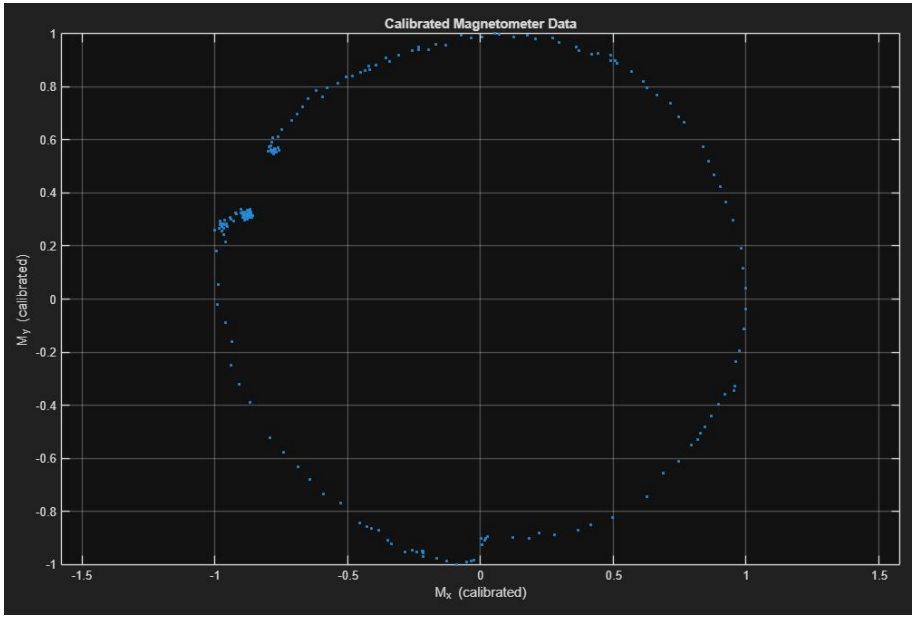

Figure 5: Magnetometer calibrated data plotted in MATLAB

Magnetometer Sensor Noise

- Purpose
  - Noise sets the minimum resolvable change in heading.
  - Understanding noise magnitude determines whether heading estimates are usable for control.
  -
- Setup
  - The same circuit from figure 1 is oriented at fixed bearings of 0°, 90°, 180°.
  - Used MATLAB magnetometer scaling and offsets to edit Arduino script to capture calibrated data (figure 6).

```
#include <Wire.h>
#include <LIS3MDL.h>

LIS3MDL mag;

void setup()
{
  Serial.begin(9600);
  Wire.begin();

  if (!mag.init())
  {
    Serial.println("Failed to detect and initialize magnetometer!");
    while (1);
  }
  mag.enableDefault();
}

// Calibration scales
const float MAG_OFFSET_X =  -838.5000;
const float MAG_OFFSET_Y = 3478;
const float MAG_SCALE_X  =   1.8835e+03;
const float MAG_SCALE_Y  =   2060;

void loop()
{
  mag.read();

  // Raw readings (int16)
  float mx = (float)mag.m.x;
  float my = (float)mag.m.y;
  float mz = (float)mag.m.z;
  // Calibrated (normalized) readings
  float mx_cal = (mx - MAG_OFFSET_X) / MAG_SCALE_X;
  float my_cal = (my - MAG_OFFSET_Y) / MAG_SCALE_Y;
  // Heading in degrees [0, 360)
  float heading = atan2f(my_cal, mx_cal) * 180.0f / PI;
  if (heading < 0) heading += 360.0f;
  // Print calibrated values + heading
  Serial.print("CAL:\t");
  Serial.print(mx_cal, 6); Serial.print("\t");
  Serial.print(my_cal, 6); Serial.print("\t");
  Serial.print(heading, 2); Serial.print("\t");
  Serial.print(mz, 0);
  Serial.println();
  delay(10);  // ~100 Hz
}
```

Figure 6: Arduino script for calibrated magnetometer data

Magnetometer Sensor Noise: Analysis

- Created a MATLAB script to calculate the average and standard deviation of the data at each heading (figure 7).
- Created a MATLAB script to plot the results of the analysis from figure 7 (figure 8).

```
figure

subplot(3,1,1)
plotNoise("106verif1noise0deg.txt", "Noise at ~0°")

subplot(3,1,2)
plotNoise("106verif1noise90deg.txt", "Noise at ~90°")

subplot(3,1,3)
plotNoise("106verif1noise180deg.txt", "Noise at ~180°")
```

Figure 8: MATLAB script to plot drift data

```
function plotNoise(fname, plotTitle)

    % Read file
    lines = readlines(fname);

    % Extract calibrated data safely
    tokens = regexp(lines, ...
        'CAL:\s*([-+]?\d*\.?\d+)\s+([-+]?\d*\.?\d+)\s+([-+]?\d*\.?\d+)', ...
        'tokens', 'once');
    tokens = tokens(~cellfun('isempty', tokens));
    A = vertcat(tokens{:});
    data = str2double(A);

    heading = data(:,3);

    % Unwrap to avoid 0/360 jumps
    heading = rad2deg(unwrap(deg2rad(heading)));

    % X-axis = sample index
    n = (1:length(heading))';

    % Stats
    fprintf('%s: mean = %.2f deg, std = %.2f deg\n', ...
        plotTitle, mean(heading), std(heading));

    % Plot
    plot(n, heading, '.')
    grid on
    xlabel('Sample Index')
    ylabel('Heading (deg)')
    title(plotTitle)

end
```

Figure 7: MATLAB script to analyze drift data

Magnetometer Sensor Noise: Results

- Standard Deviation and Average of Magnetometer Data (figure 9).
  - Data suggests compass is accurate.
- Magnetometer readings plotted with respect to sample index (figure 10).

```
>> a106verif1noise
Noise at ~0°: mean = -0.56 deg, std = 0.41 deg
Noise at ~90°: mean = 89.69 deg, std = 0.44 deg
Noise at ~180°: mean = 179.39 deg, std = 0.73 deg
```

Figure 9: Standard Deviation and Average of Magnetometer Data at 0°, 90°, 180°.

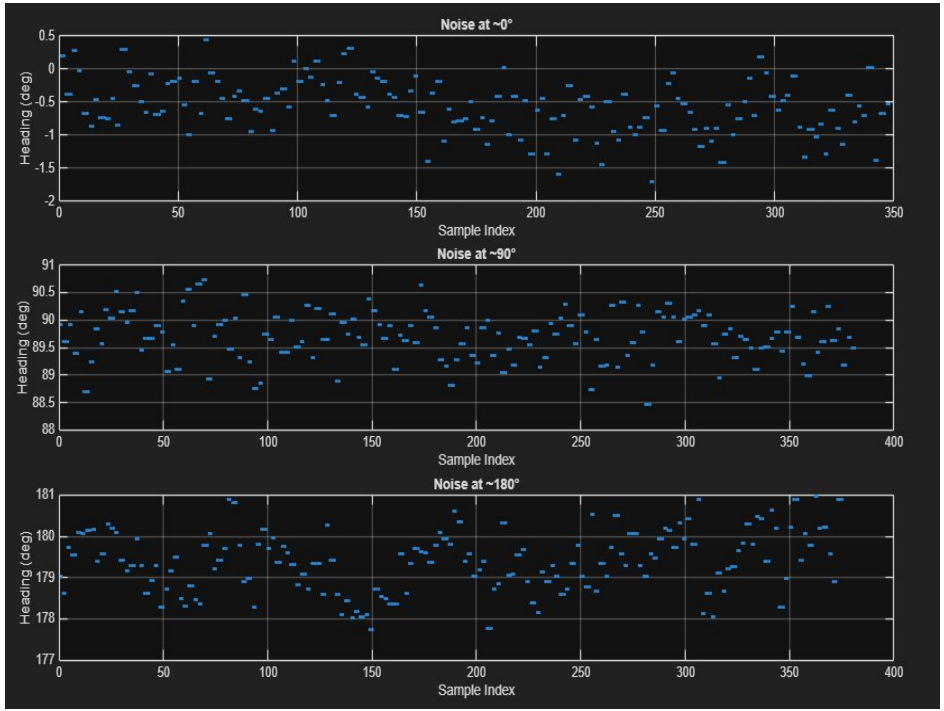

Figure 10: Magnetometer readings vs sample index

Magnetometer Sensor Drift

- Purpose
  - Drift causes heading estimates to change over time even when the system is stationary.
  - Quantifying drift determines whether recalibration or compensation is required during operation.
- Setup
  - Used the same magnetometer circuit from figure 1, but added tape (figure 11).
  - Sensor held at a fixed orientation for one hour.
  - Heading data recorded every 10 minutes using calibrated Arduino output.

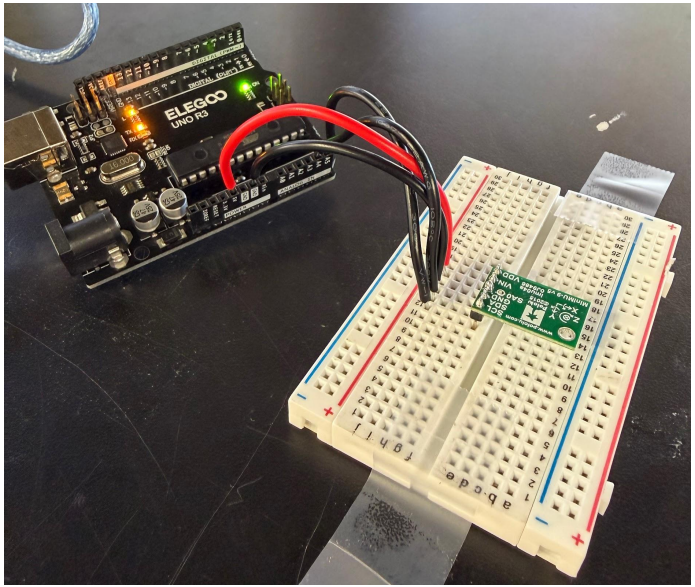

Figure 11: Taped down circuit

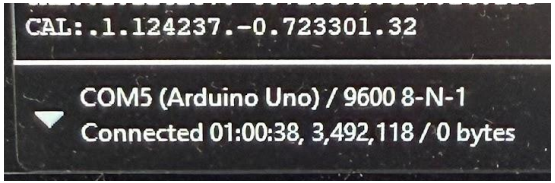

Figure 12: Arduino IDE serial monitor at 1 hour.

Magnetometer Sensor Drift: Analysis

- Created a MATLAB function to compute the mean heading at each 10-minute interval (figure 13).
- Applied angle unwrapping to prevent 0°/360° discontinuities.
- Calculated drift relative to the initial heading.
- Performed a linear fit to estimate drift rate in degrees per hour.

```
% Drift files (10 min spacing)
files = ["106verif1driftT00.txt","106verif1driftT10.txt","106verif1driftT20.txt", ...
         "106verif1driftT30.txt","106verif1driftT40.txt","106verif1driftT50.txt", ...
         "106verif1driftT60.txt"];

t_min = (0:10:60)';      % minutes
t_hr  = t_min/60;        % hours

% Mean heading at each timepoint
H = zeros(numel(files),1);
for i = 1:numel(files)
    H(i) = meanHeadingFromDriftFile(files(i));
end

% Drift relative to t=0
drift_deg = H - H(1);

% Plot drift over time
figure; grid on
plot(t_hr, drift_deg, '-o')
xlabel('Time (hours)')
ylabel('Heading drift (deg)')
title('Magnetometer Heading Drift (relative to t=0)')

% Drift rate (deg/hr) via linear fit
p = polyfit(t_hr, drift_deg, 1);
drift_rate_deg_per_hr = p(1);

fprintf("Drift rate = %.3f deg/hour\n", drift_rate_deg_per_hr);

% plot mean heading
figure; grid on
plot(t_hr, H, '-o')
xlabel('Time (hours)')
ylabel('Mean heading (deg)')
title('Mean Heading vs Time (Unwrapped)')
```

```
function hmean = meanHeadingFromDriftFile(fname)
    lines = readlines(fname);

    tokens = regexp(lines, ...
        'CAL:\s*([-+]?\d*\.?\d+)\s+([-+]?\d*\.?\d+)\s+([-+]?\d*\.?\d+)', ...
        'tokens', 'once');

    tokens = tokens(~cellfun('isempty', tokens));
    A = vertcat(tokens{:});
    data = str2double(A);

    heading = data(:,3);                % deg, may wrap near 0/360
    heading = rad2deg(unwrap(deg2rad(heading)));  % unwrap to remove wrap jumps

    hmean = mean(heading);
end
```

Figure 13: MATLAB script to compute mean heading and drift

Magnetometer Sensor Drift: Results

- Heading drift plotted as a function of time in hours (figure 14).
- Linear regression used to estimate drift rate (figure 15).
  - Acceptable drift rate.

```
>> a106verif1drift
Drift rate = 0.002 deg/hour
```

Figure 15: MATLAB-calculated drift rate.

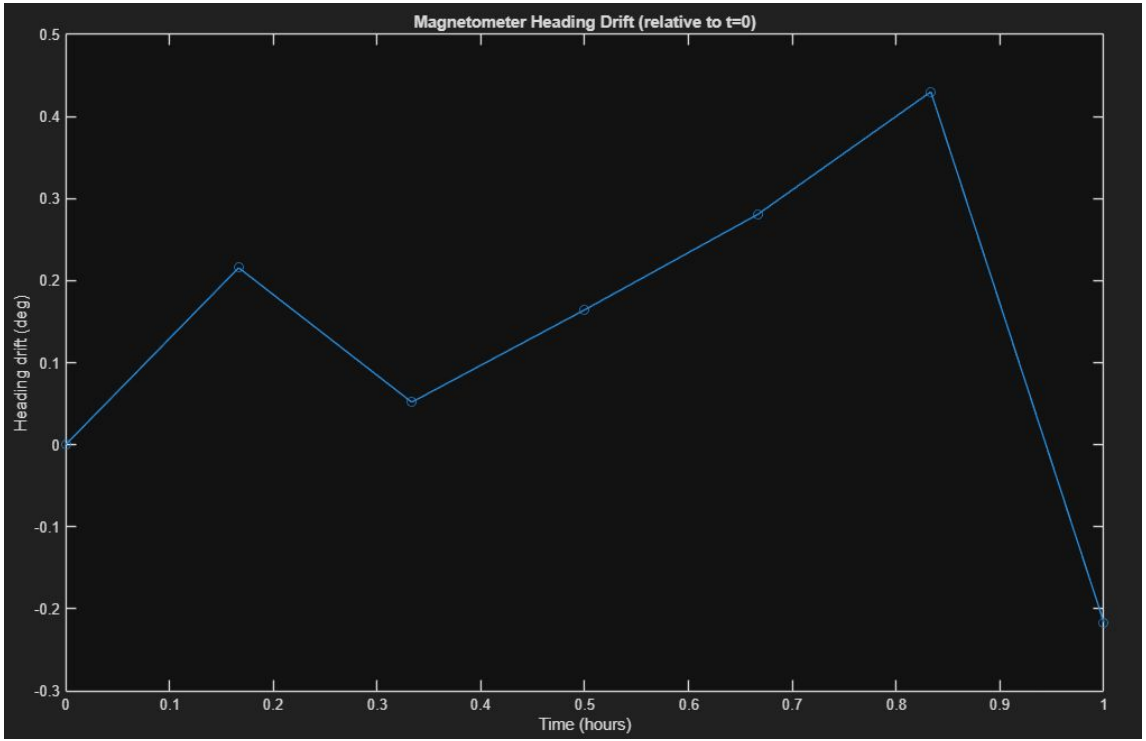

Figure 14: MATLAB plots showing angular drift per hour relative to theta = 0.

Servo Position Accuracy

- Purpose
  - Steering performance depends on the servo reaching commanded angles accurately.
  - Position error directly translates to steering error in the chosen Ackermann geometry.
- Setup
  - Servo powered by an external regulated supply with a common ground to the Arduino (figure 16).
  - Servo commanded to discrete angles: 0°, 30°, 60°, 90°, 120°, 150° (figure 17).

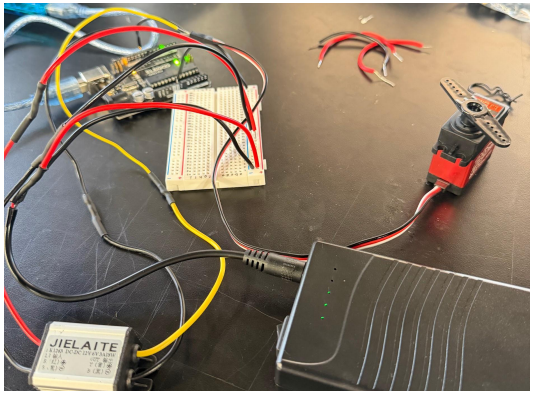

Figure 16: Servo circuit

```
#include <Servo.h>

Servo servo;

int angles[] = {0, 30, 60, 90, 120, 150};
int idx = 0;

void setup() {
  servo.attach(9);
  servo.write(angles[0]);
}

void loop() {
  servo.write(angles[idx]);
  delay(3000);          // 3 seconds per position

  idx++;
  if (idx >= 6) {
    idx = 0;            // wrap around
  }
}
```

Figure 17: IDE program to command rotations

Servo Position Accuracy: Analysis

- Actual angles measured using a protractor (table 1).
- Created a MATLAB script to plot position error versus commanded angle (figure 18).

Table 1: Servo commanded vs actual rotation.

Commanded (degrees)	Actual (degrees)
0	8
30	39
60	71
90	101
120	132
150	162

```
% Commanded vs actual angles (degrees)
cmd = [0 30 60 90 120 150];
actual = [8 39 71 101 132 162];

% Position error
error = actual - cmd;

% Plot error vs commanded angle
figure; grid on
plot(cmd, error, '-o', 'LineWidth', 1.5)
xlabel('Commanded Angle (deg)')
ylabel('Position Error (deg)')
title('Servo Position Accuracy')

% Optional stats (nice to mention)
fprintf('Mean error = %.2f deg\n', mean(error));
fprintf('Max error  = %.2f deg\n', max(abs(error)));

```

Figure 18: MATLAB script to compute and plot position error.

Servo Position Accuracy: Results

- Servo position error plotted as a function of commanded angle (figure 19).
- Mean error and maximum error computed (figure 20).

```
>> a106verif1servo
Mean error = 10.50 deg
Max error  = 12.00 deg
```

Figure 20: MATLAB-computed average and maximum servo command discrepancy.

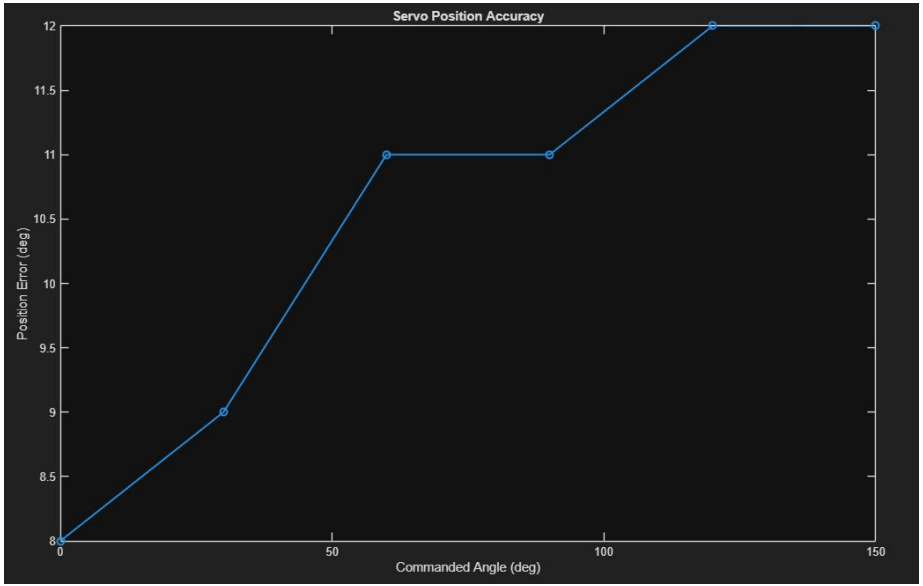

Figure 19: MATLAB plot of servo position error.

Key Takeaways

- Sensor calibration is essential for meaningful heading estimation.
- Noise and drift define the limits of sensor reliability.
- Servo accuracy defines the limits of steering precision for the chosen ackerman system.
    - The accuracy of the commanded angles is of concern, on average 10 degrees off, overshooting. This issue will be escalated to the TA for assisting prognosis.
- Together, these results establish a validated foundation for system-level tuning in Verification #2.