**Lab 6**

**Introduction**

For this lab, we investigated arm movements and pointing accuracy under various conditions. Our goal was to determine if target size and target memorization influence the accuracy and kinematics of reaching. We predicted that as target size decreased, reaching speed would decrease. We also hypothesized that if targets needed to be memorized, this would reduce accuracy.

**Methods and Materials**

* Optitrack
* BPE script
* Electrodes

For the setup of the lab, we connected the EMG and Optitrack with the BNC cable to the amplifier port. In Optitrack Motive, we set the required sync parameters and set up the necessary electrodes and Optitrack markers to record our data in sync. We then used the provided BPE script to collect synchronized data for the following experiments.

Part A: Eyes Open, Wide Target

For this segment, we set rectangular outlines for the start and target positions for the subject’s fingers as seen in the figure below. The subject moved from the start position to the wide target as quickly and accurately as possible, before returning to the start position. This was repeated for 8 trials.

Part B: Eyes Open, Narrow Target

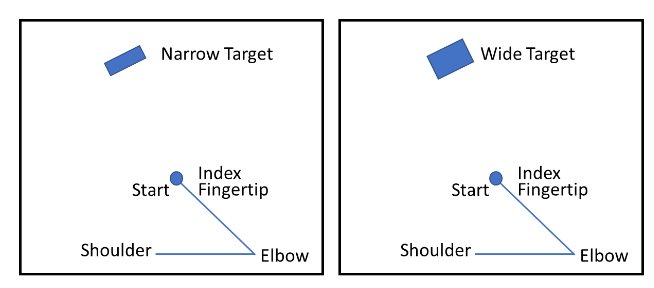
For this segment, the same 8 trials were performed with the narrow target.

Part C: Eyes Open

For this segment, the subject pointed to where the target was, then returned to the start position. This was performed 8 times.

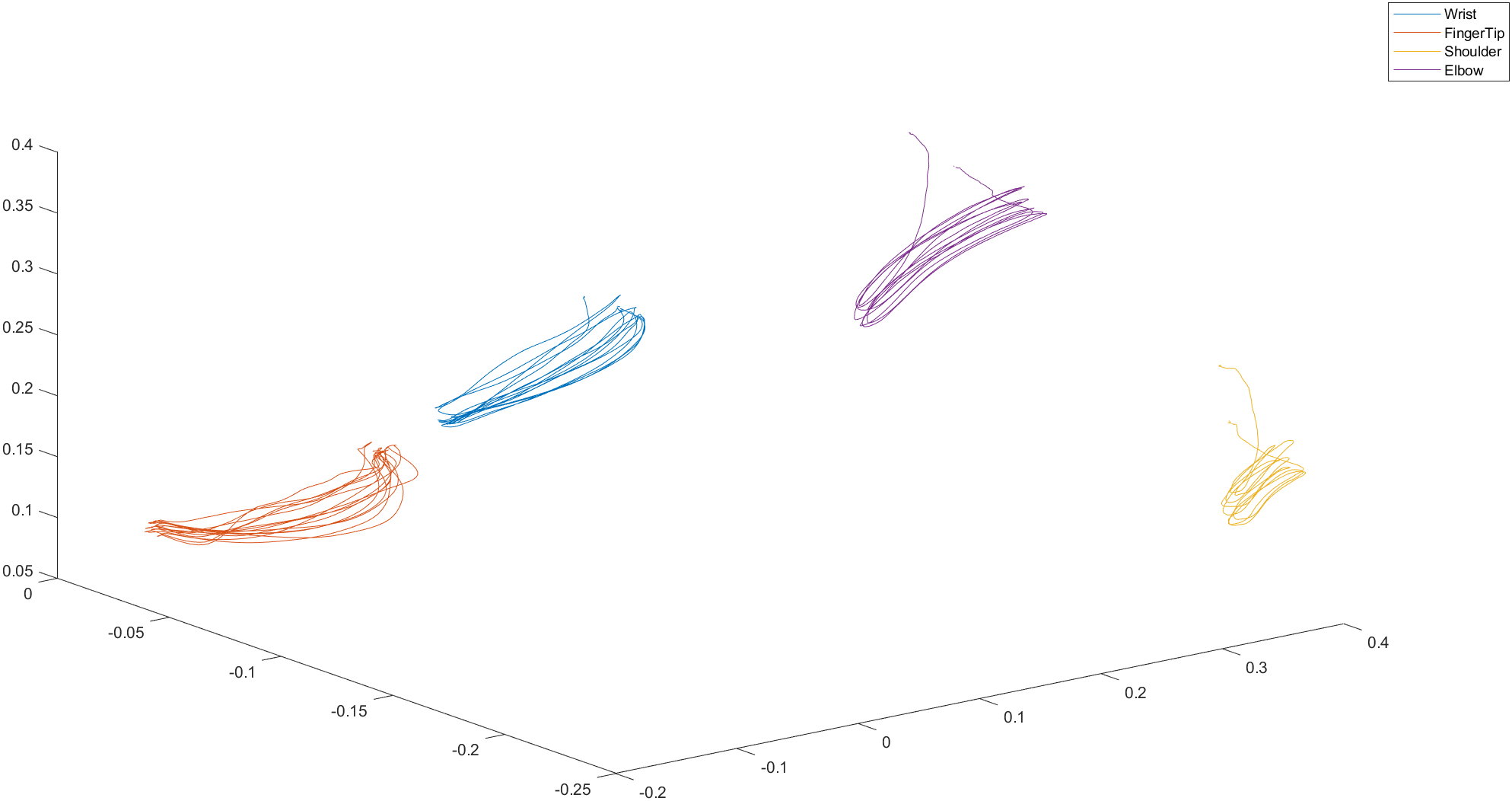
Part D: Eyes Closed

As soon as the target was removed, the subject closed their eyes, waited 8 seconds, and then pointed to where the target was, before returning to the start position. This was performed 8 times.

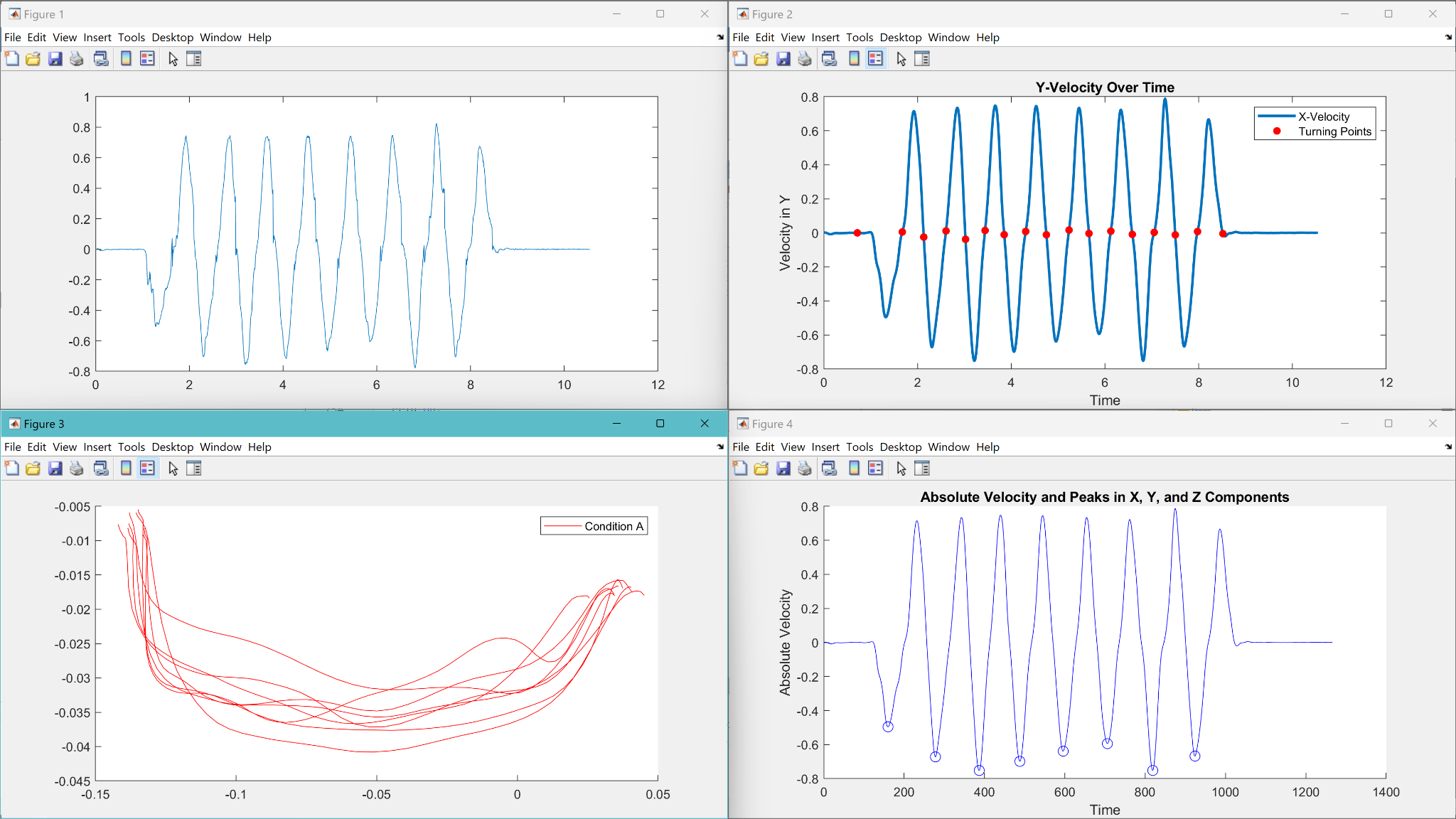


**Results**

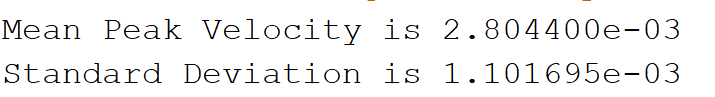
**Part A**

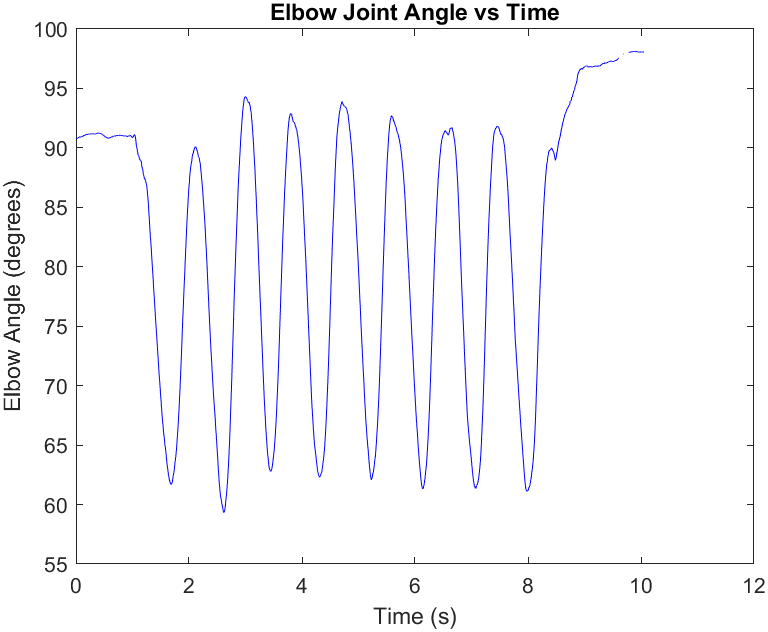


**Figure 1** is the positional data of Part A of the lab.

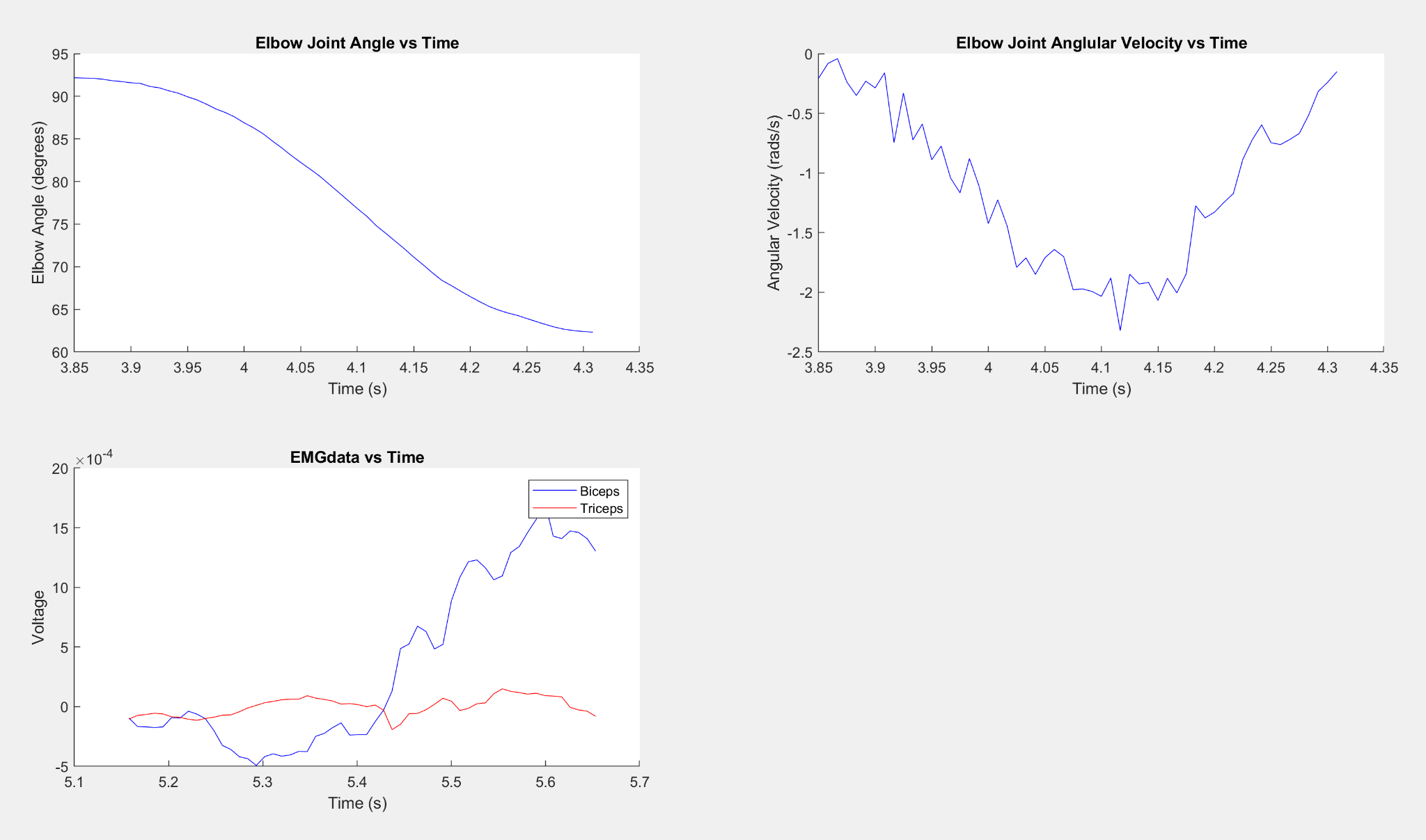
****

**Figure 2** Shows the Velocity in the X direction Unfiltered, Velocity Filtered with Turning Points, 8 Reach Lines, and Max Velocity Lines

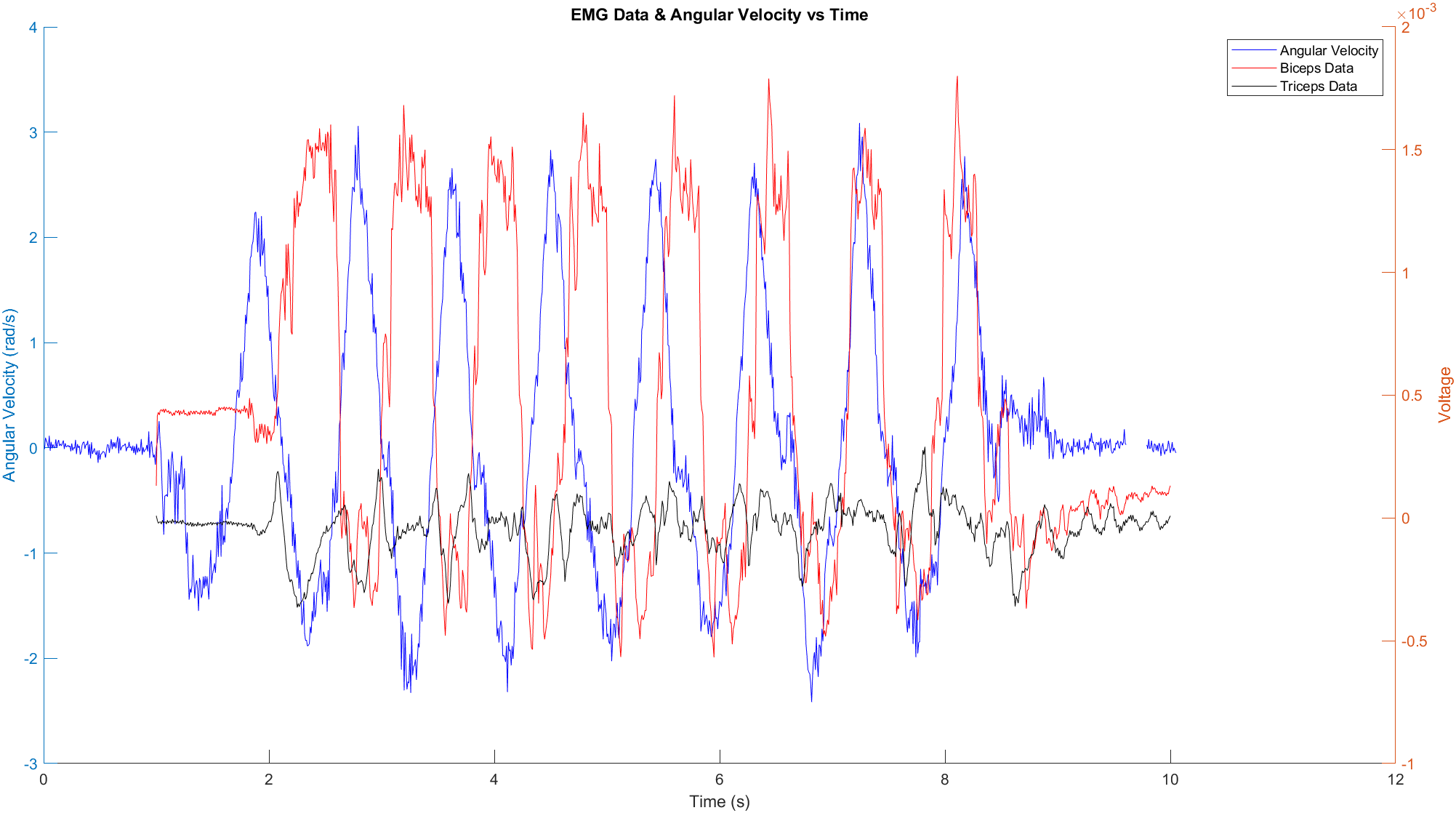
****

**Mean and standard deviation of Peak Velocities of Reach Lines**

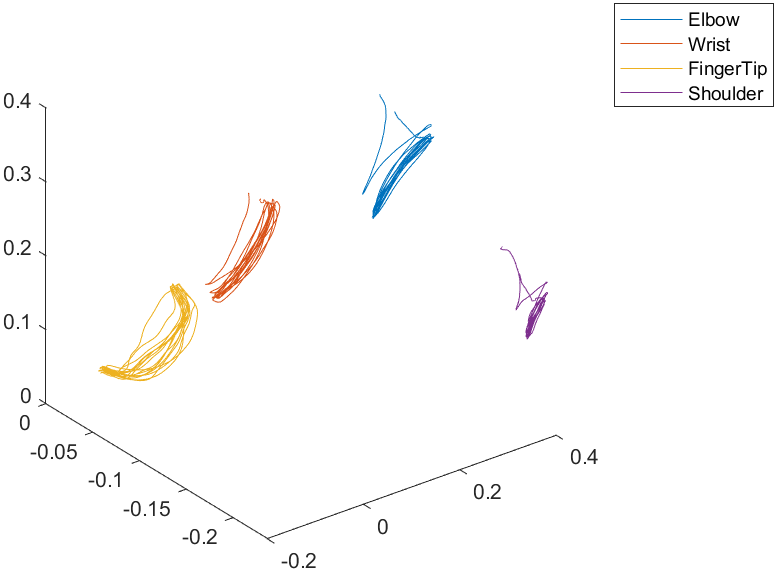
**Figure 3** Elbow Joint Angle vs Time

****

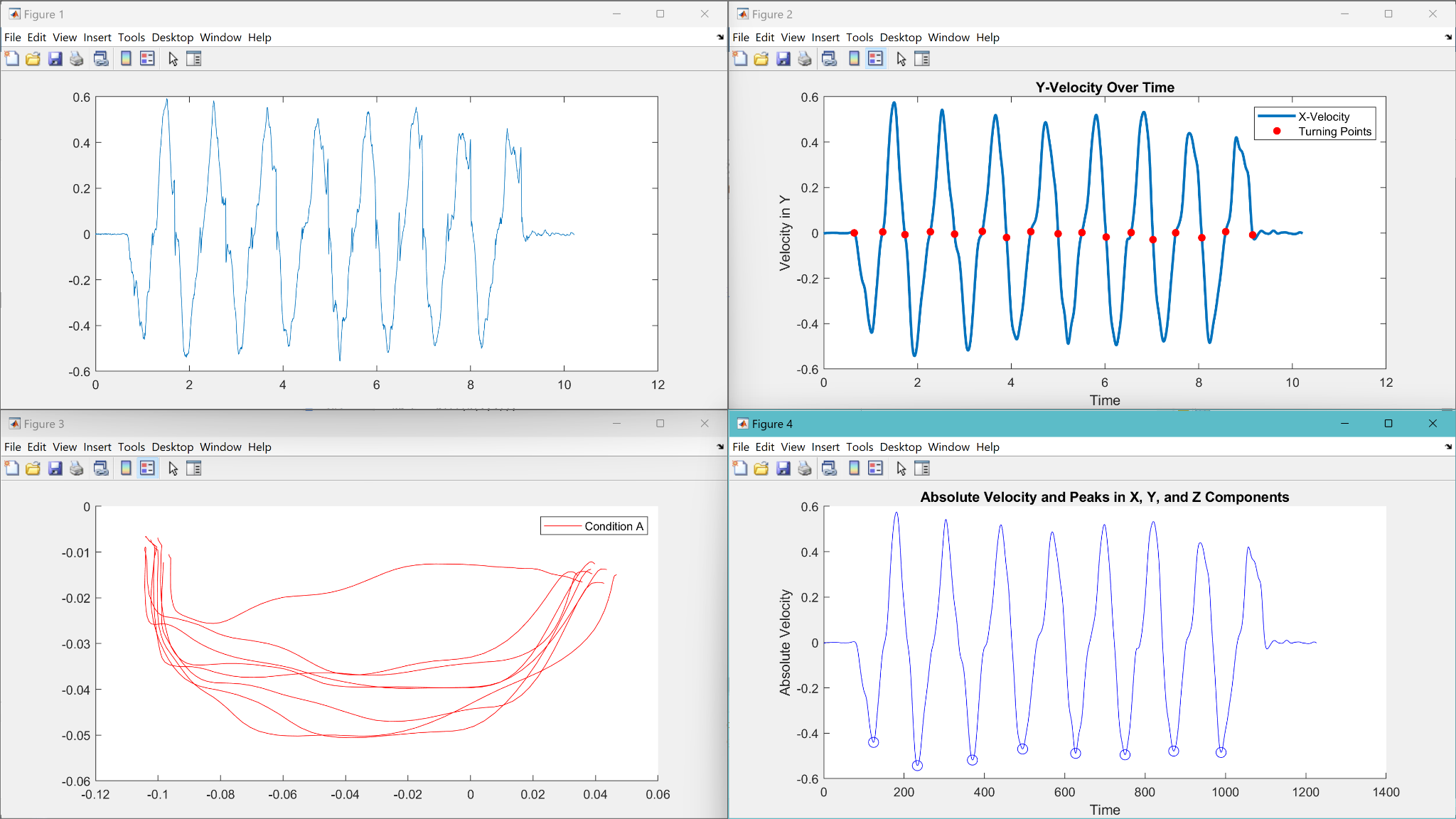
**Figure 4** Shows Elbow Joint Angle, Angular Velocity, and EMGdata vs Time

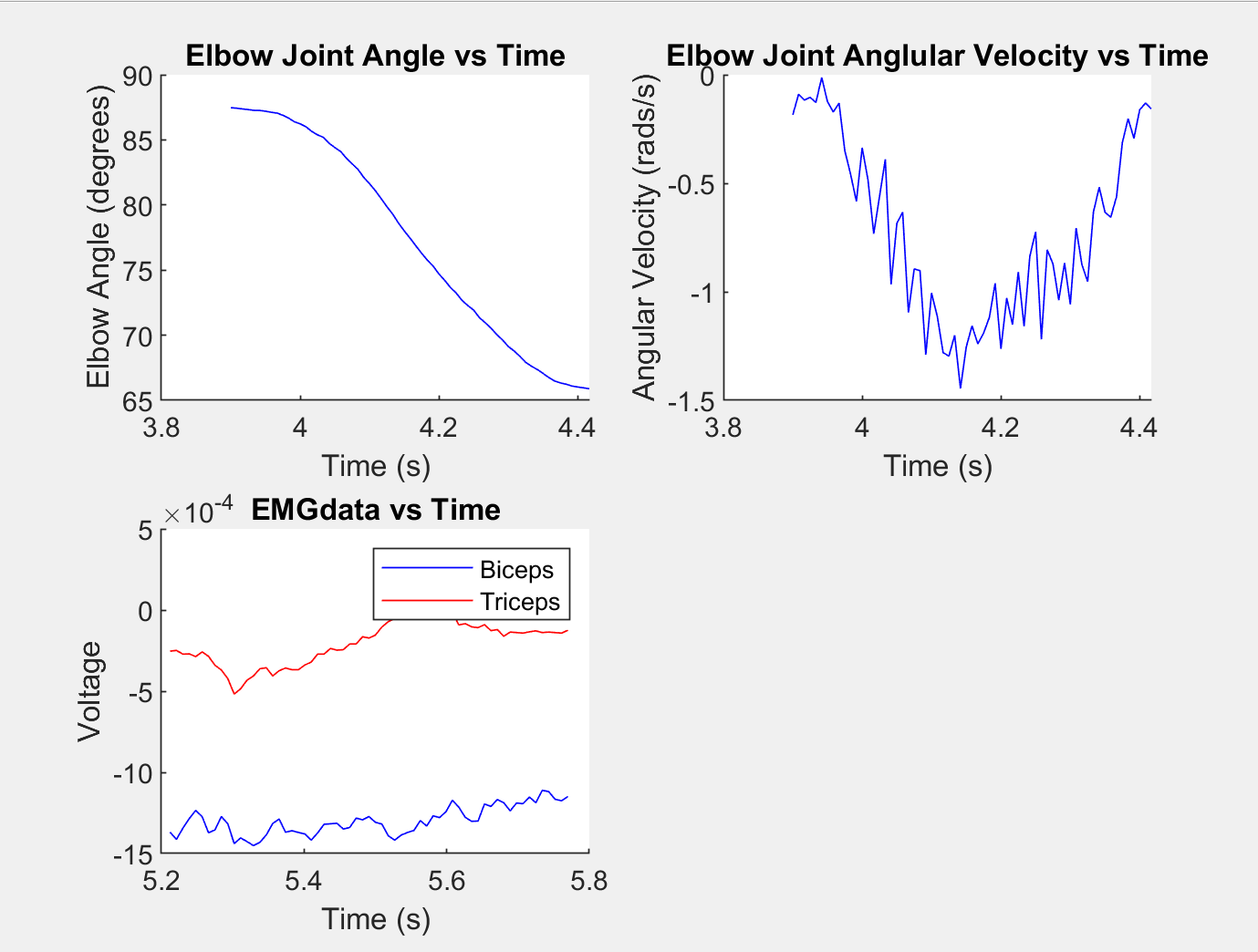
****

**Figure** **5** Is the Angular Velocity, Biceps, and Triceps EMG data on the same timescale

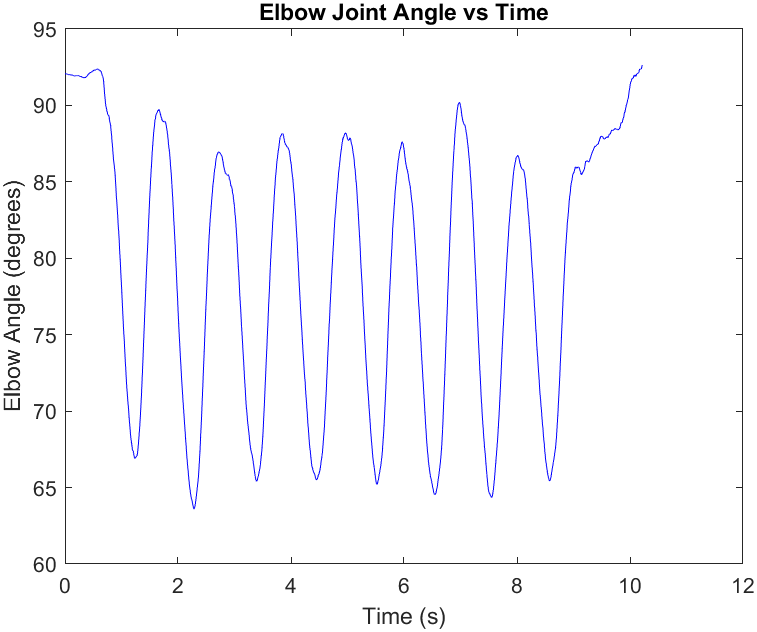
**Part B**

**Figure 6** is the positional data of Part B of the lab.

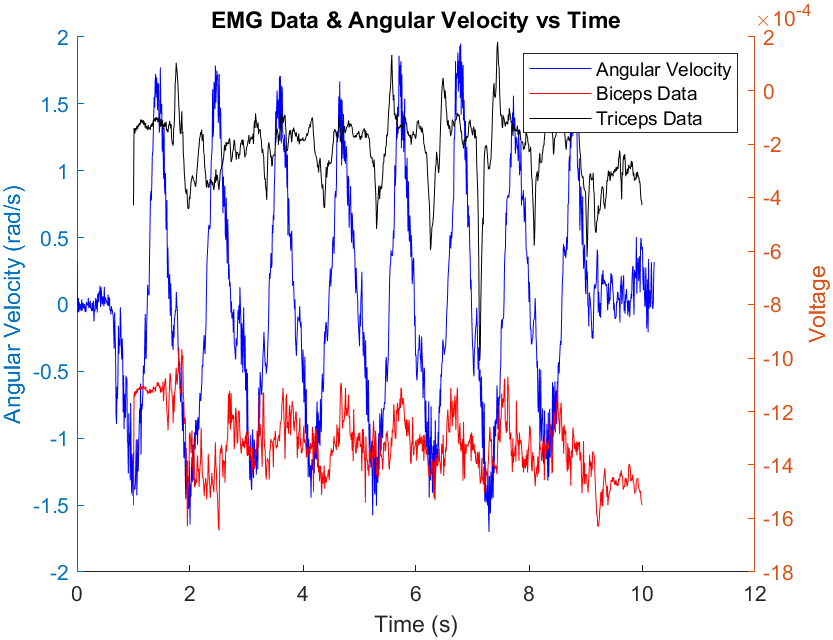
* **Figure 7** Shows the Velocity in the X direction Unfiltered, Velocity Filtered with Turning Points, 8 Reach Lines, and Max Velocity Lines
* Mean Peak Velocity is 3.783656e-03
* Standard Deviation is 7.008824e-04

****

**Figure 8** Shows Elbow Joint Angle, Angular Velocity, and EMGdata vs Time

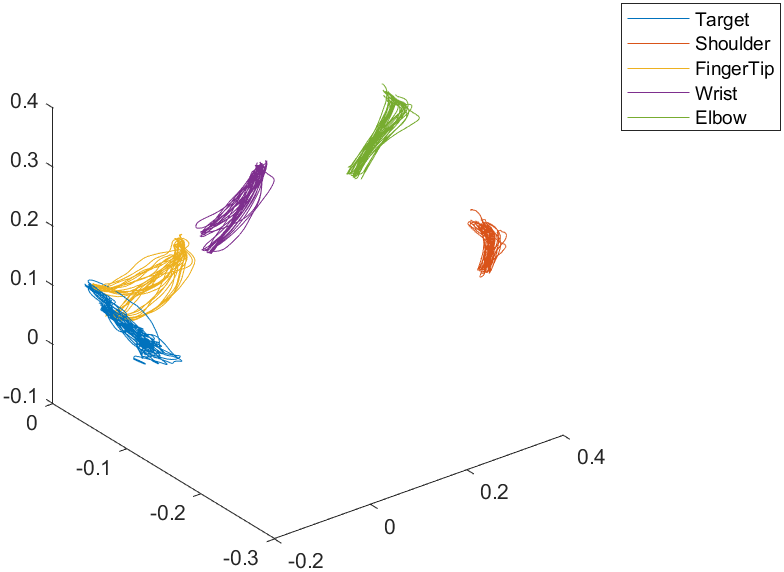
****

**Figure 9** Elbow Joint Angle vs Time

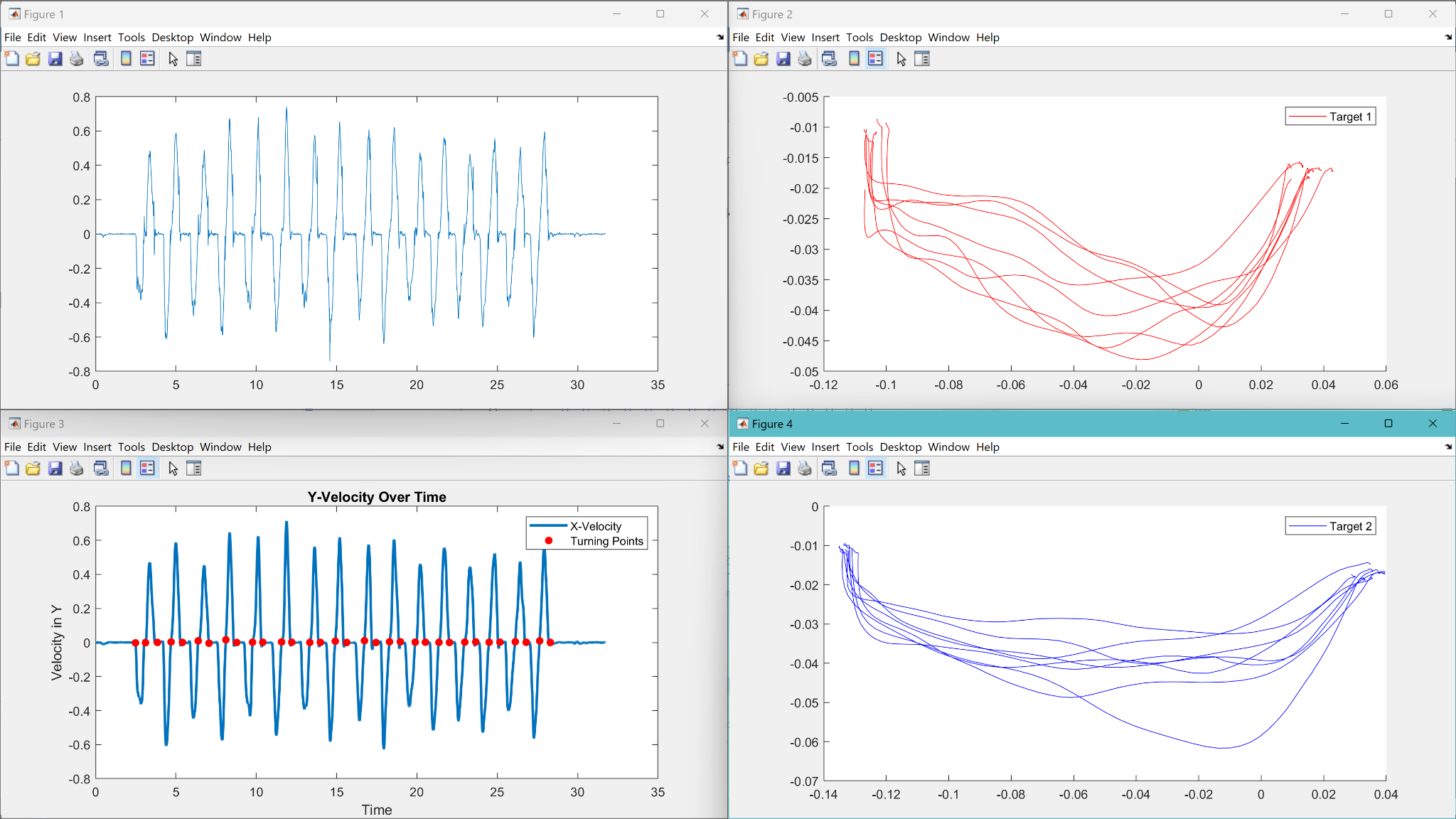
****

**Figure** **10** Is the Angular Velocity, Biceps, and Triceps EMG data on the same timescale

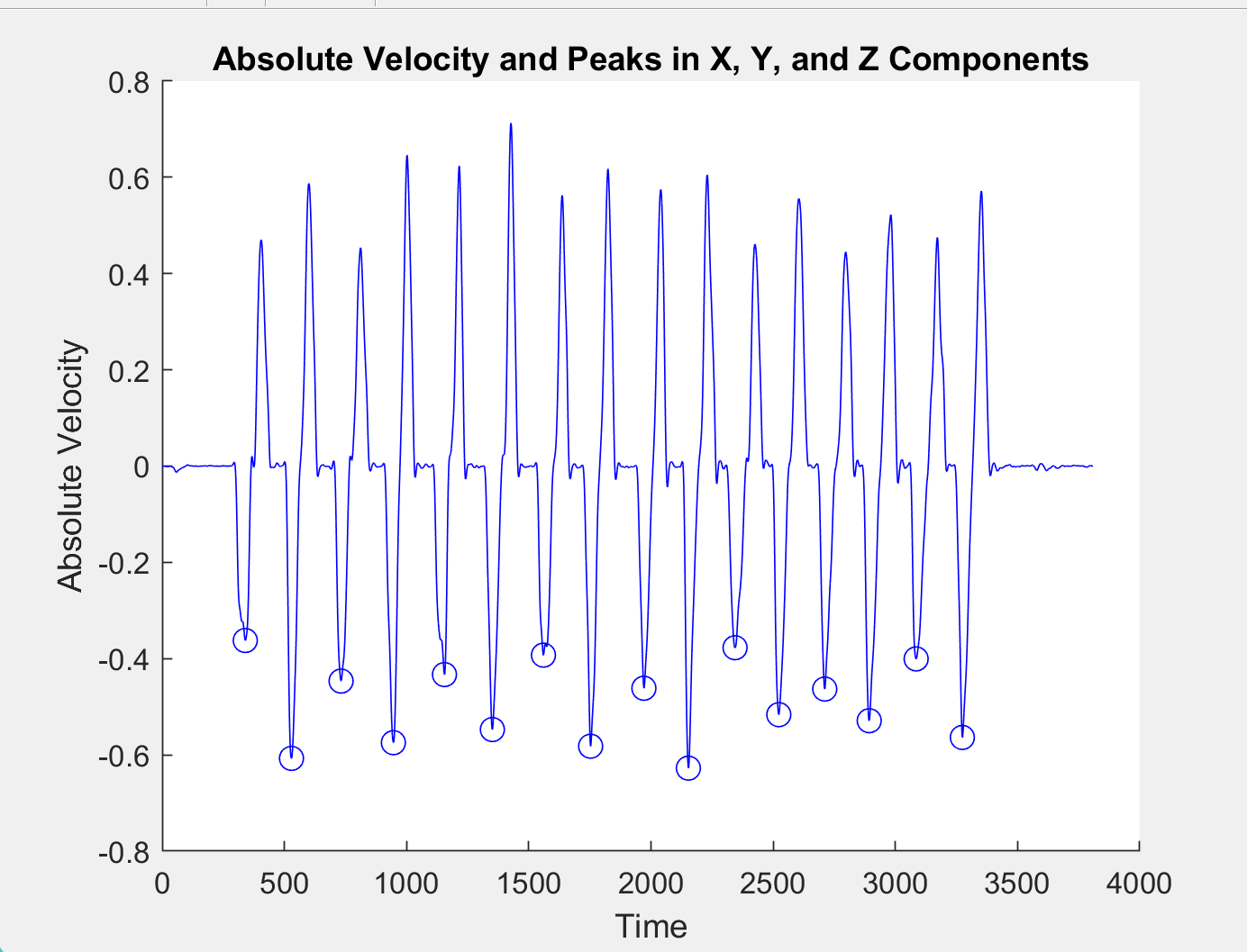
**Part C**

****

**Figure 11** is the positional data of Part C of the lab.

****

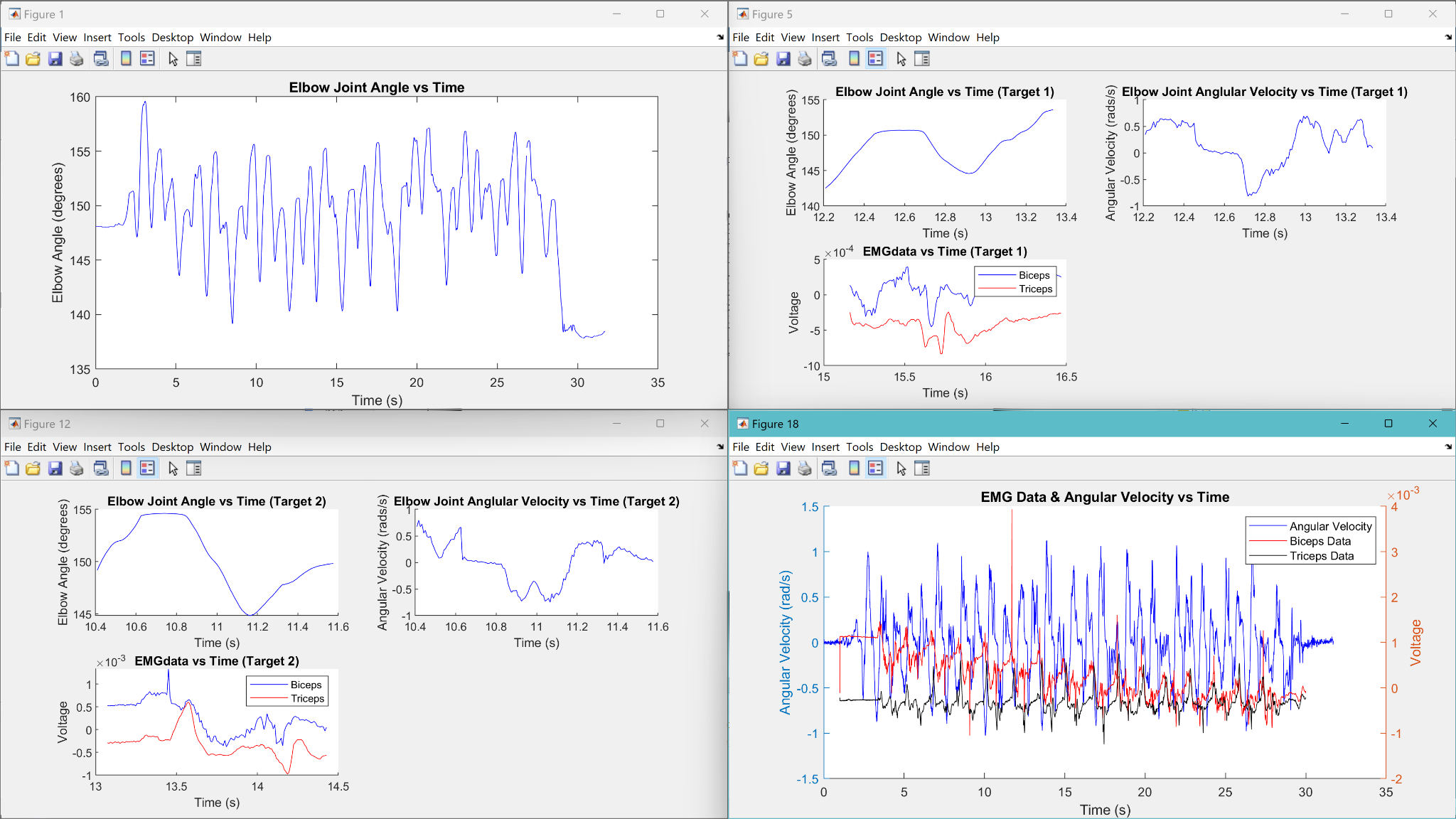
**Figure 12** Shows the Velocity in the X direction Unfiltered, Velocity Filtered with Turning Points, 8 Reach Lines For Target 1 and Target 2,

****

**Figure 13** Max Velocity Lines

Mean Peak Velocity is 8.977056e-02

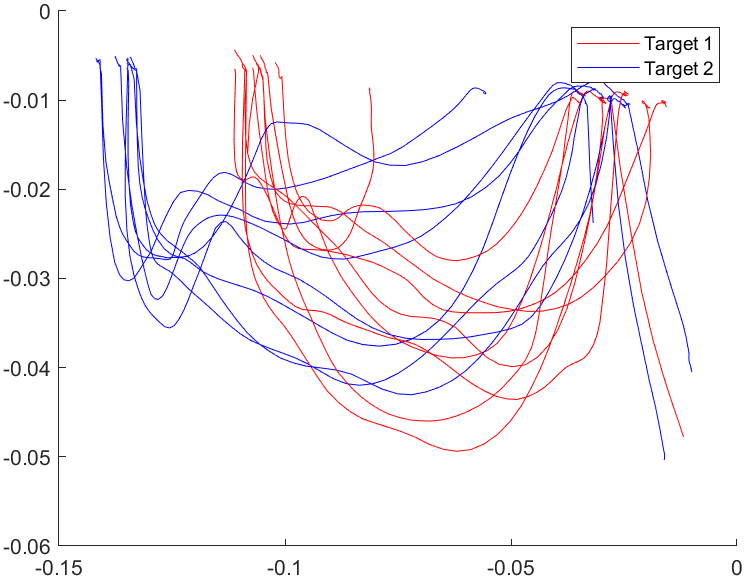
Standard Deviation is 1.118399e-01

****

**Figure 13** Shows Elbow Joint Angle vs Time, Elbow Joint Angle, Angular Velocity, and EMGdata vs Time For Target 1 and Target 2, And then Angular Velocity & EMG Data on one Figure

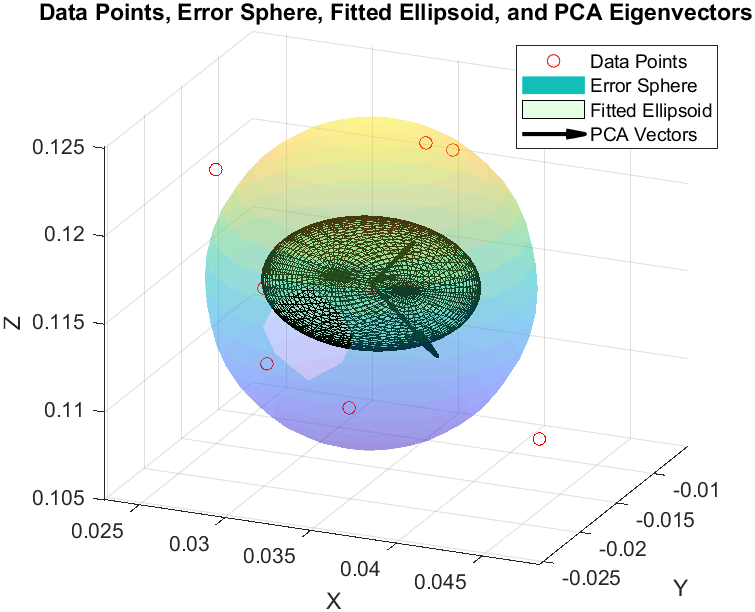
****

**Figure 14** Shows unfiltered Target Velocity, Filter Target Velocity and its Turning Points, and Target 1 and 2 plotted with its corresponding Movement Reversals

****

**Figure 15** is the path lines for both targets

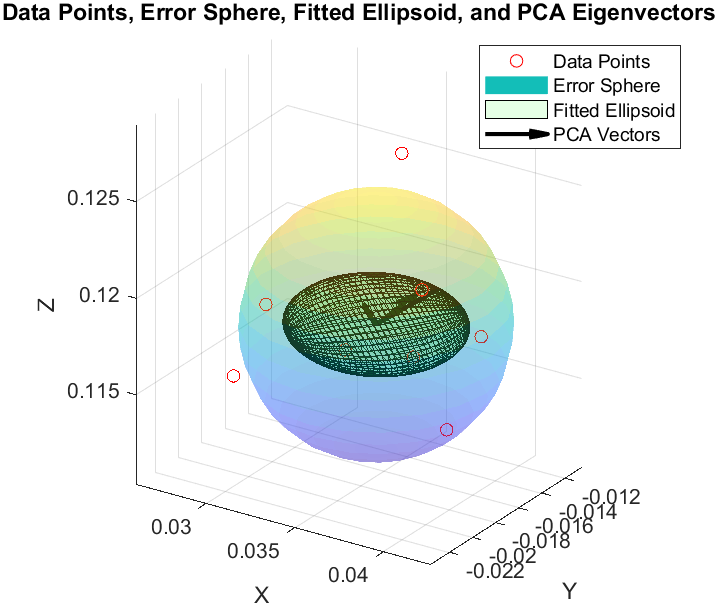
* Mean distance is 8.467165e-02
* Standard Deviation is 7.887275e-03

****

**Figure 16** is the error sphere showing the variability of the point attempts, and a fitted ellipsoid corresponding and rotated to the data’s eigenvalues for Target 1

Absolute Constant error for target 1 = 0.162094490123044

Constant Error for target 1 = 0.161955006972520

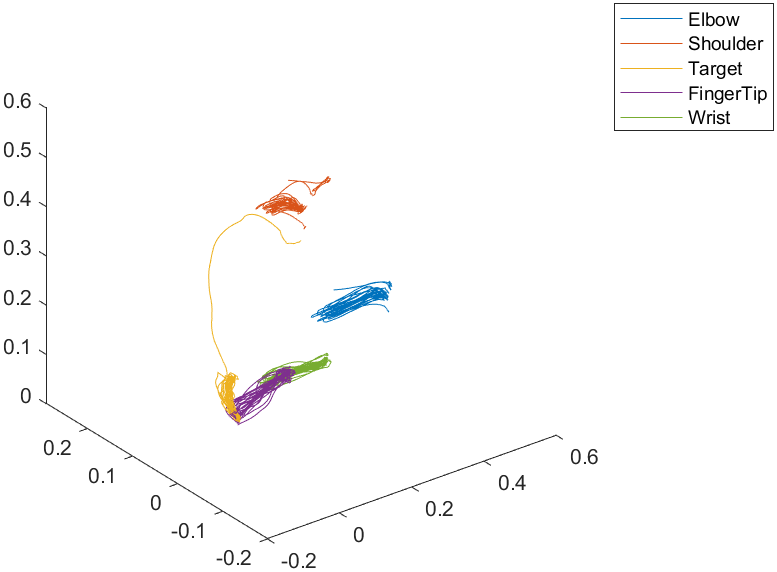
****

**Figure 17** is the error sphere showing the variability of the point attempts, and a fitted ellipsoid corresponding and rotated to the data’s eigenvalues for Target 2

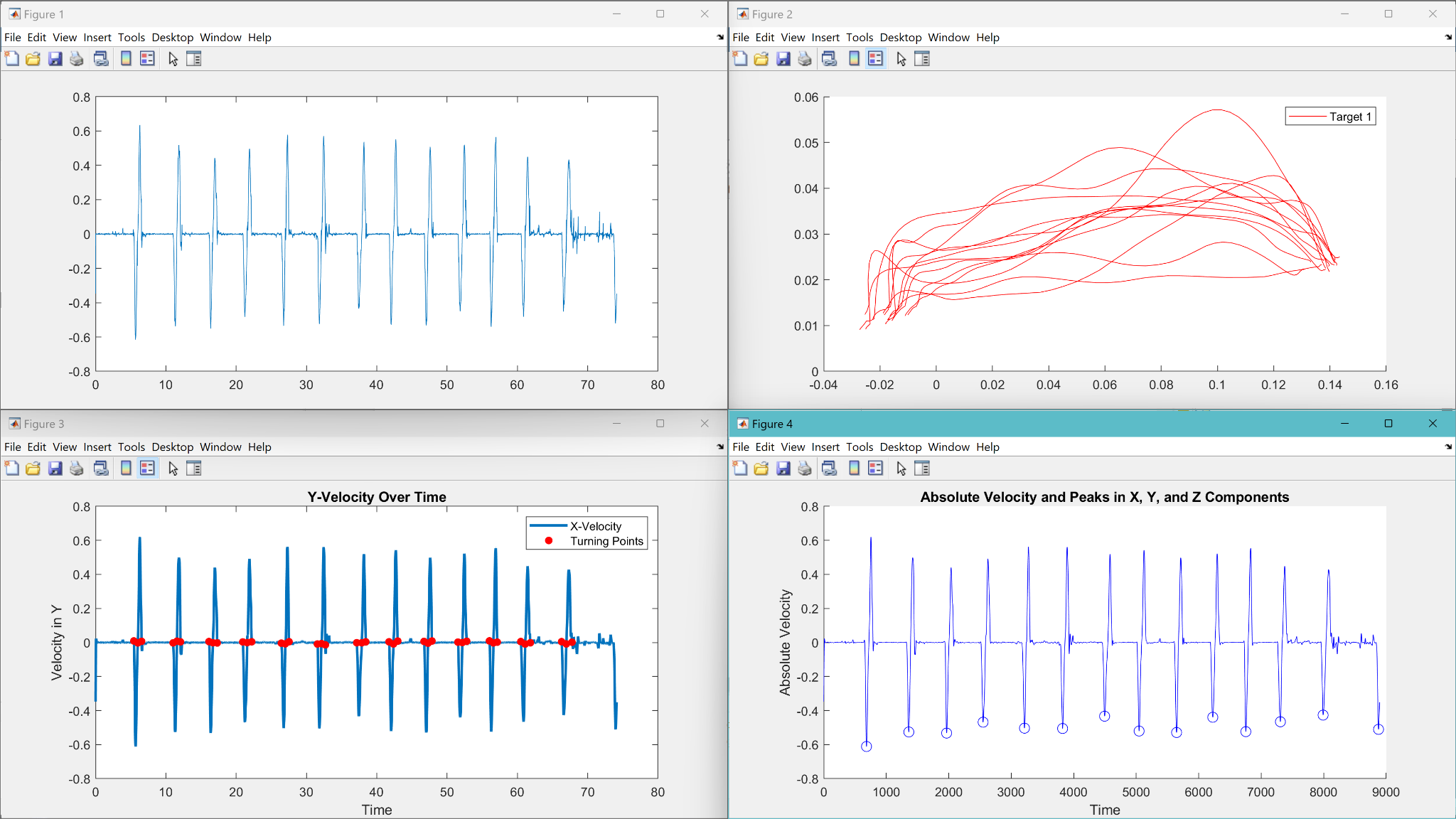
Absolute Constant error for target 2 = 0.175346321056379

Constant Error for target 2 = 0.175299079338400

**Part D1**

****

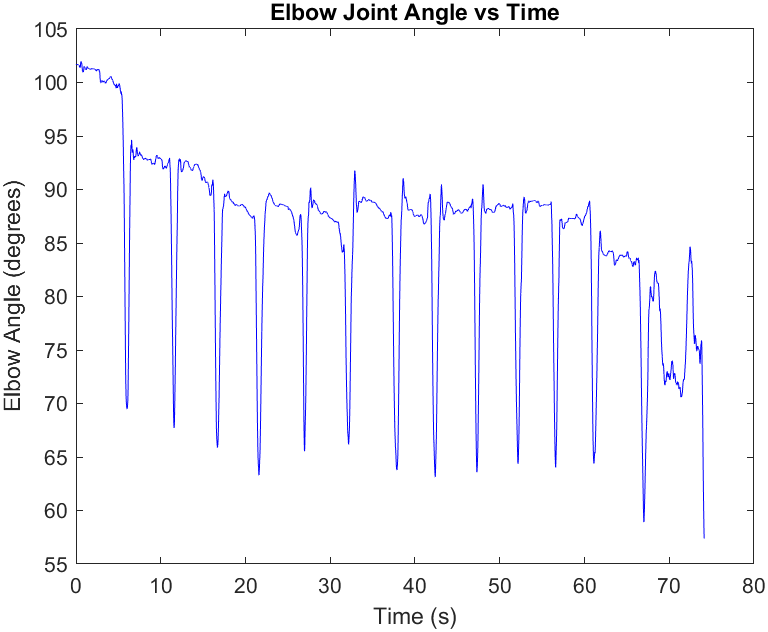
**Figure 18** is the positional data of Part D1 of the lab.

****

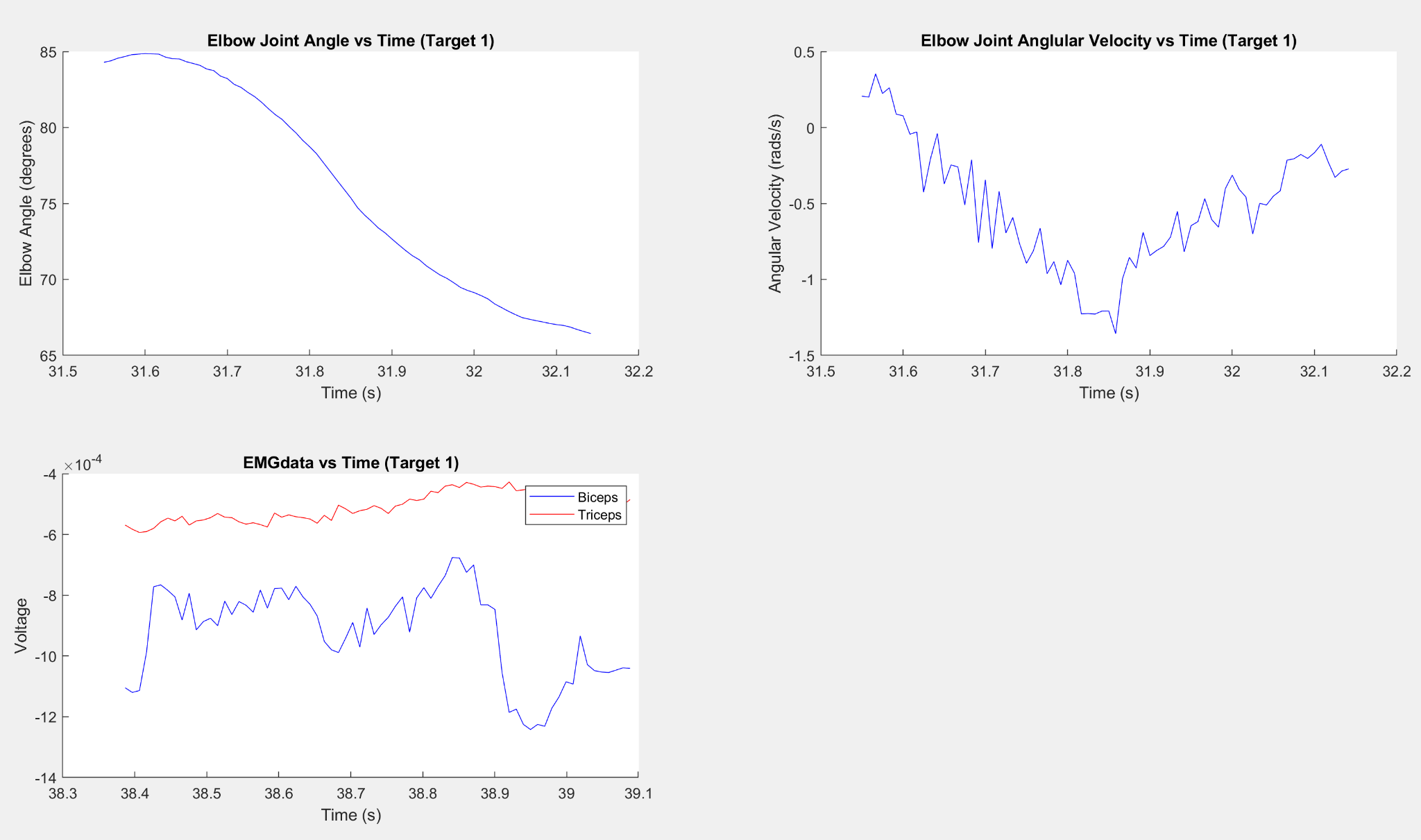
* **Figure 19** Shows the Velocity in the X direction Unfiltered, Velocity Filtered with Turning Points, 8 Reach Lines, and Max Velocity Lines

Mean distance is 8.467165e-02

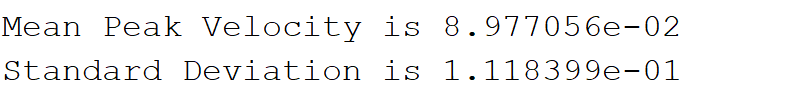
Standard Deviation is 7.887275e-03

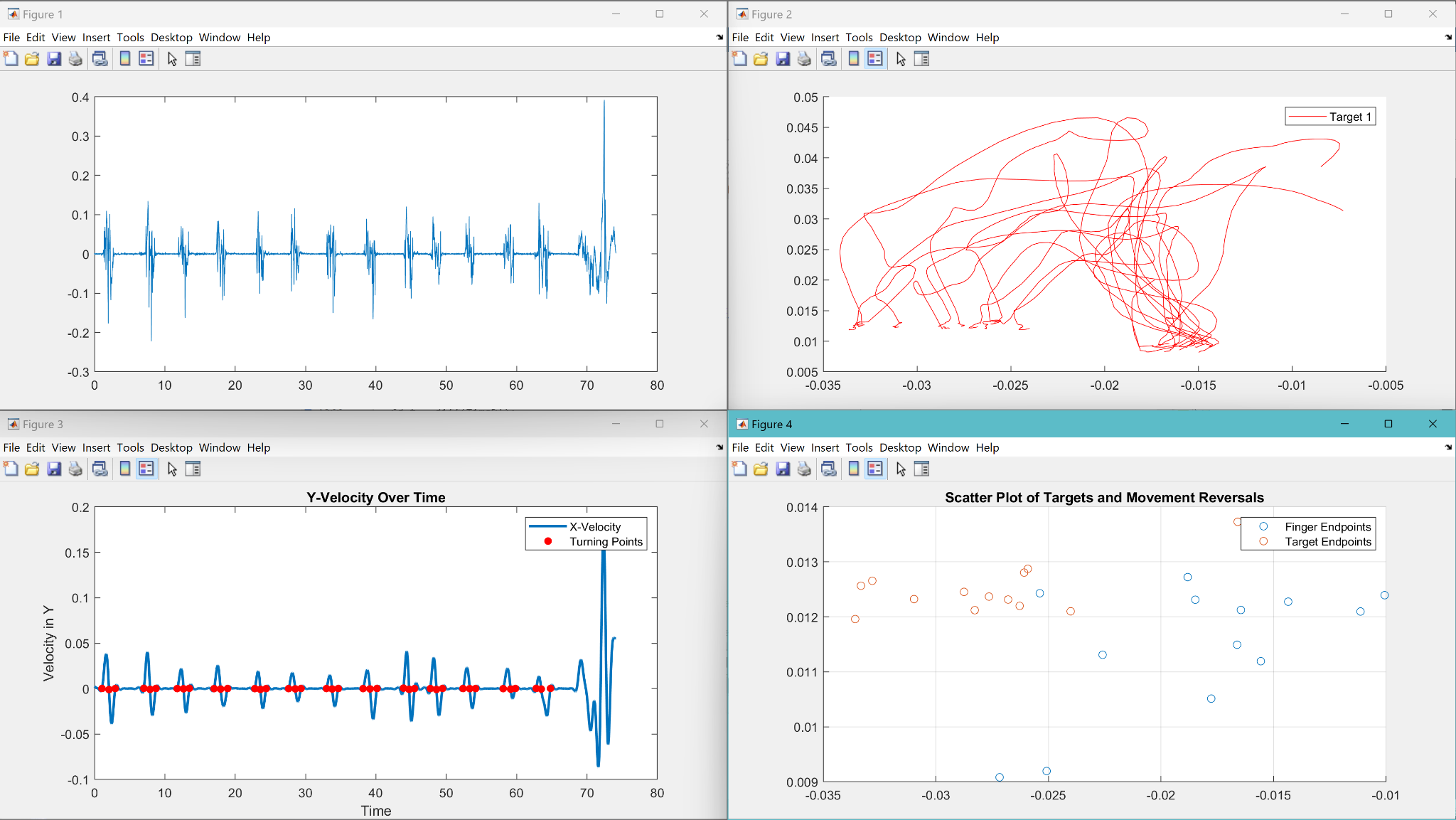
****

**Figure 20** Shows elbow joint angle vs time

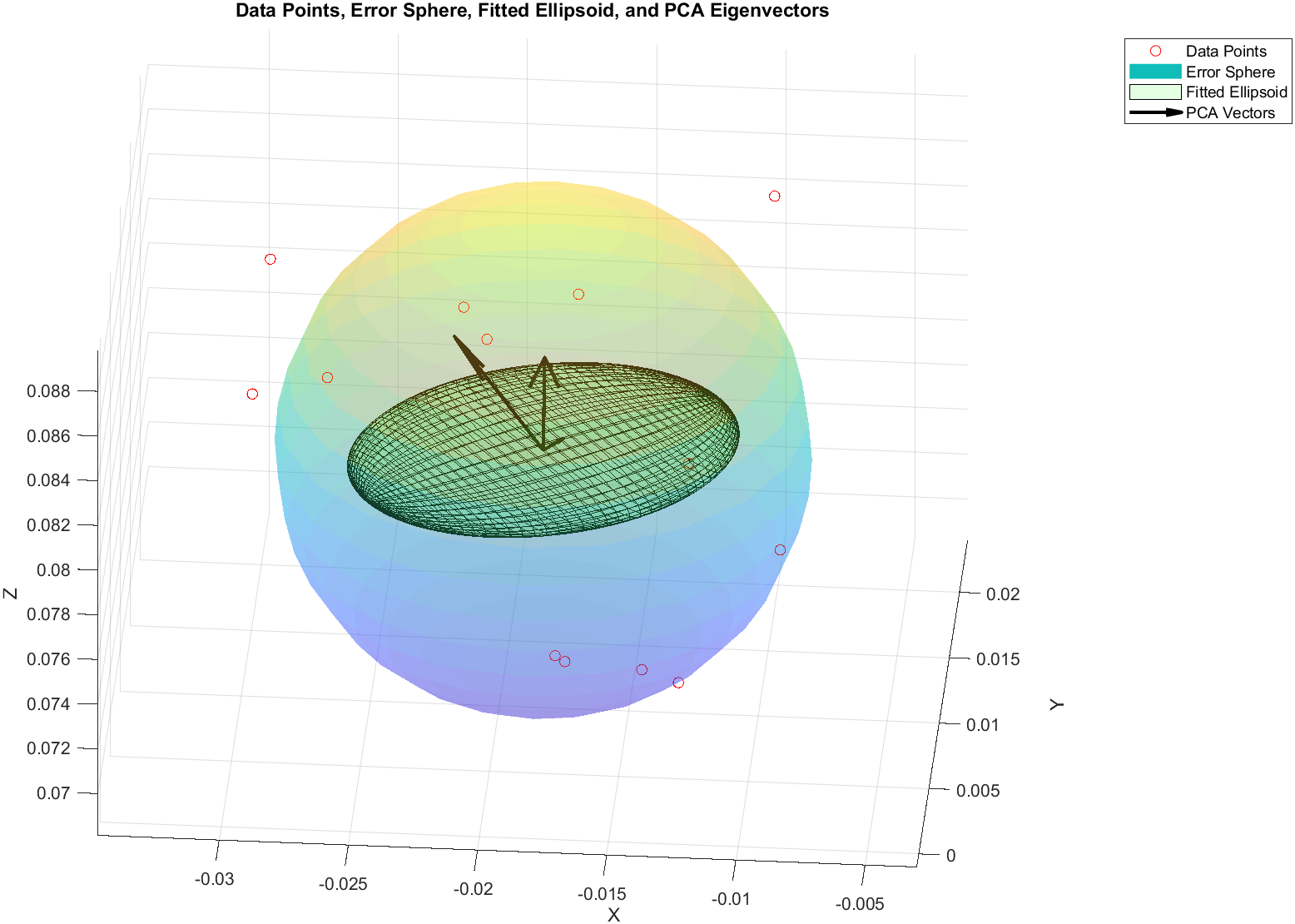
****

**Figure 21** Shows Elbow Joint Angle, Angular Velocity, and EMGdata vs Time

****

****

**Figure 22** Shows unfiltered Target Velocity, Filter Target Velocity and its Turning Points, 8 Target Path lines, and Target 1 plotted with its corresponding Movement Reversals

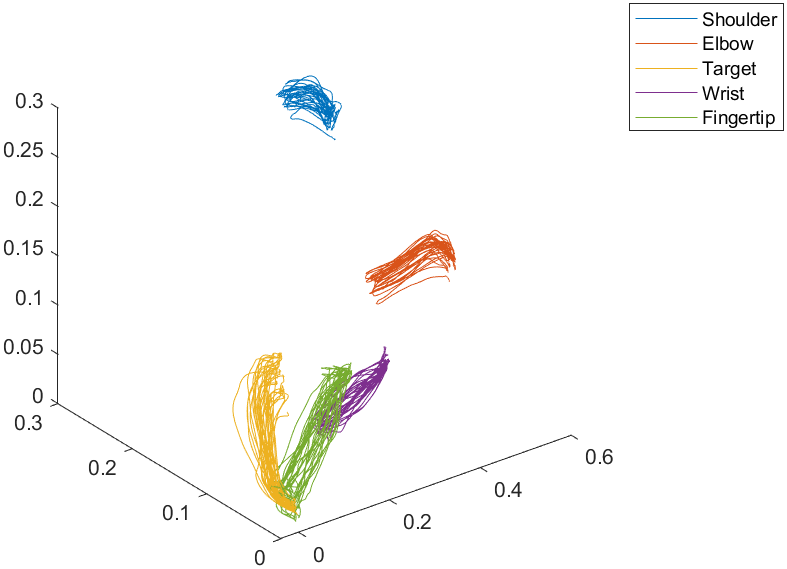
****

**Figure 23** is the error sphere showing the variability of the point attempts, and a fitted ellipsoid corresponding and rotated to the data’s eigenvalues for Target 1

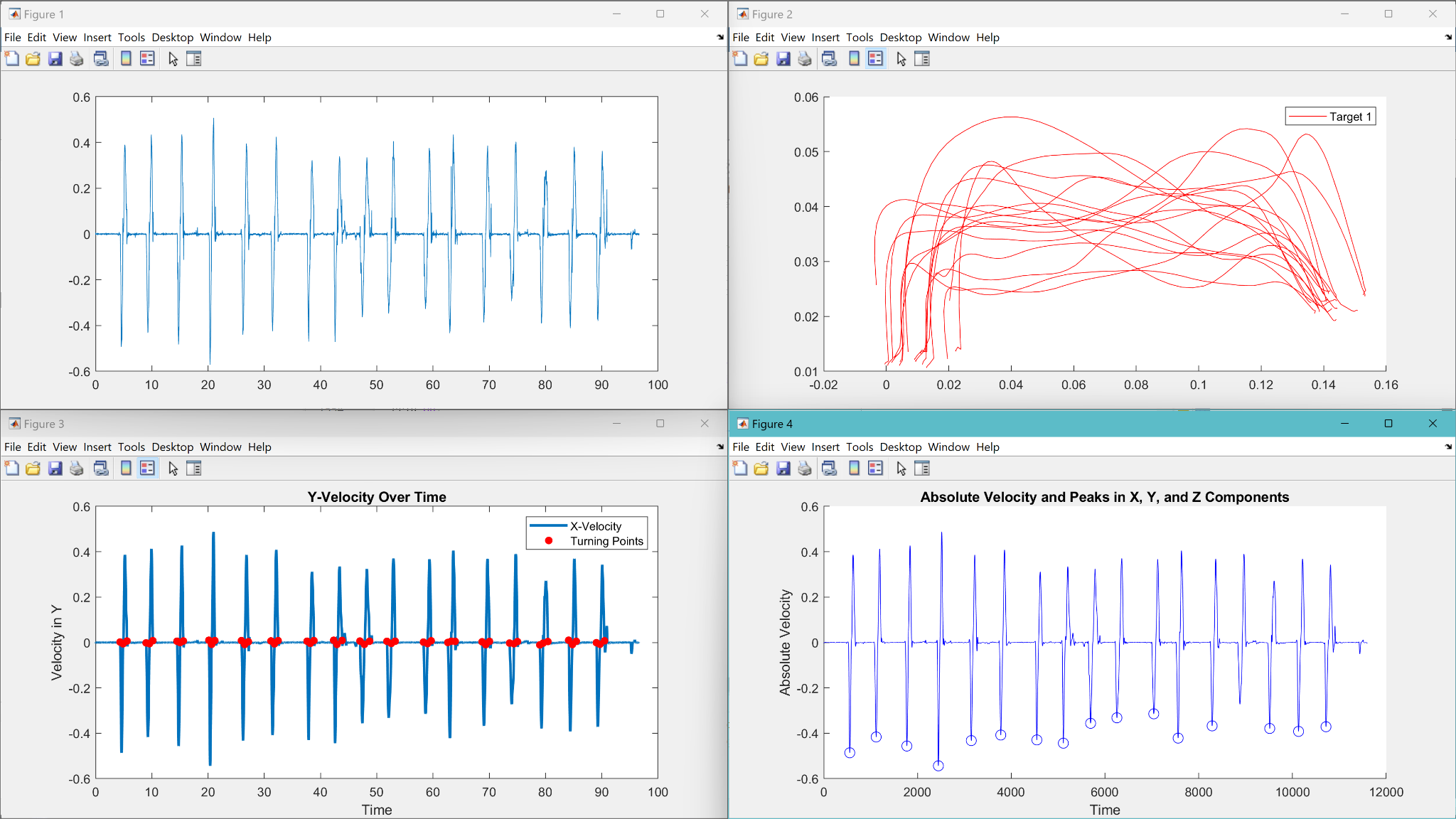
* absolute constant error is 0.084671649033201

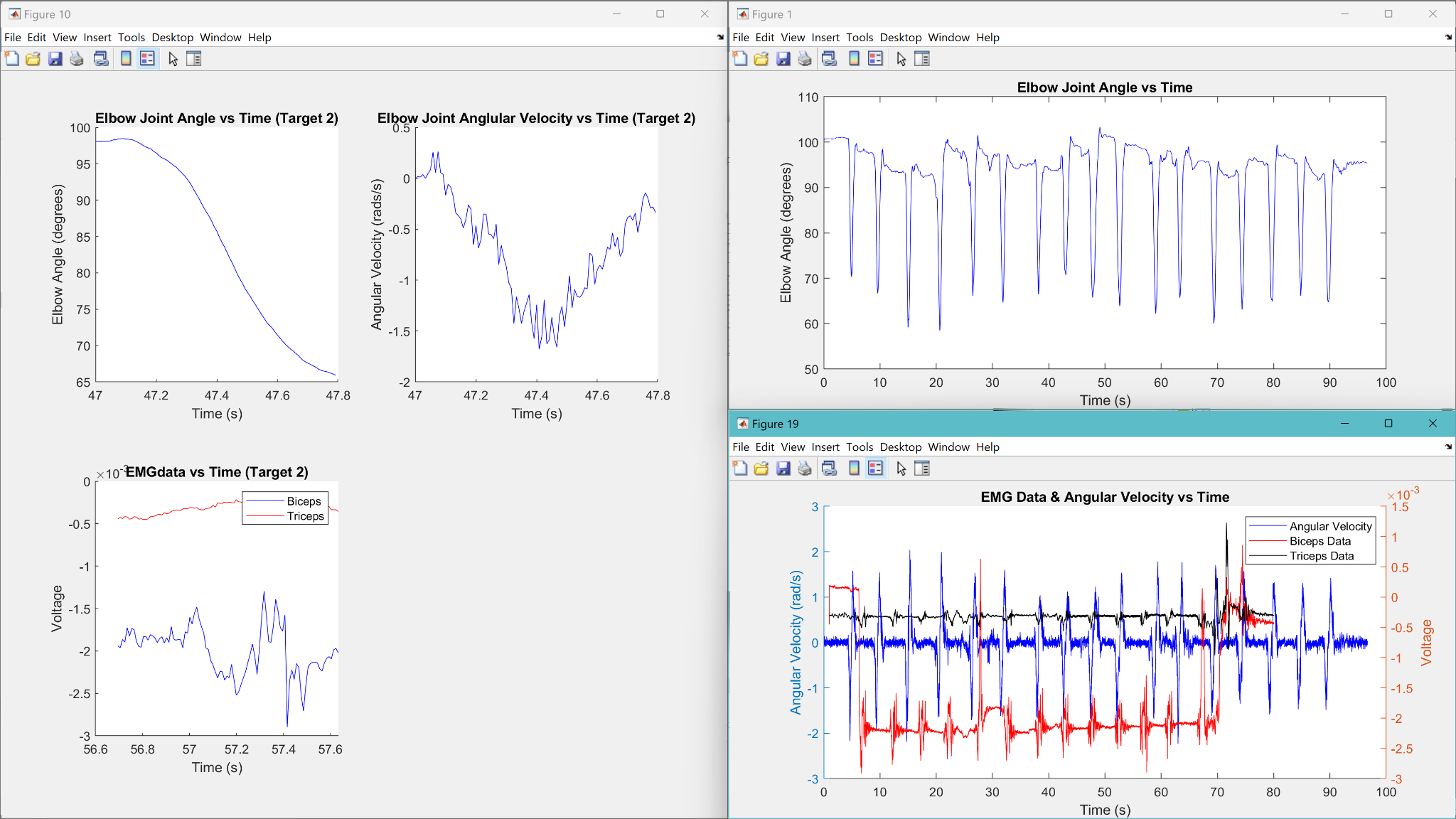
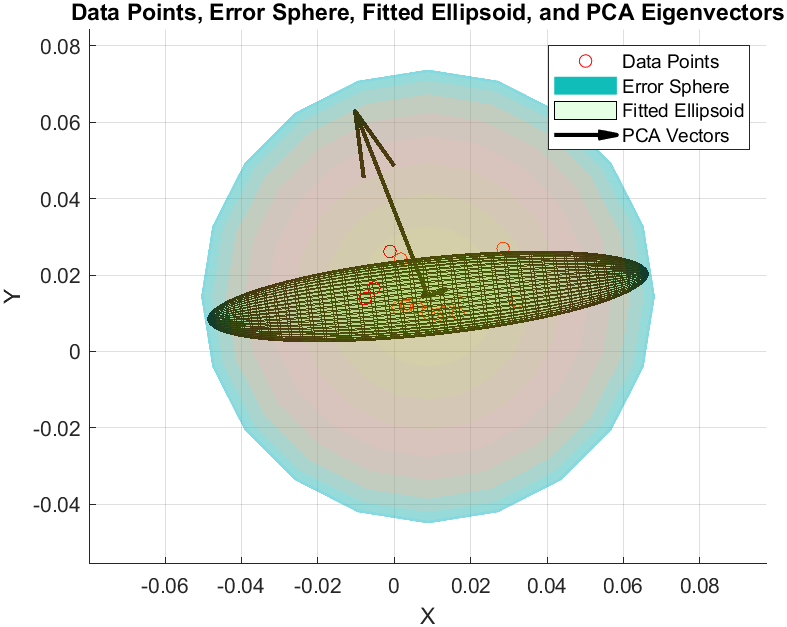
constant error 0.084419954849437

**Part D2**

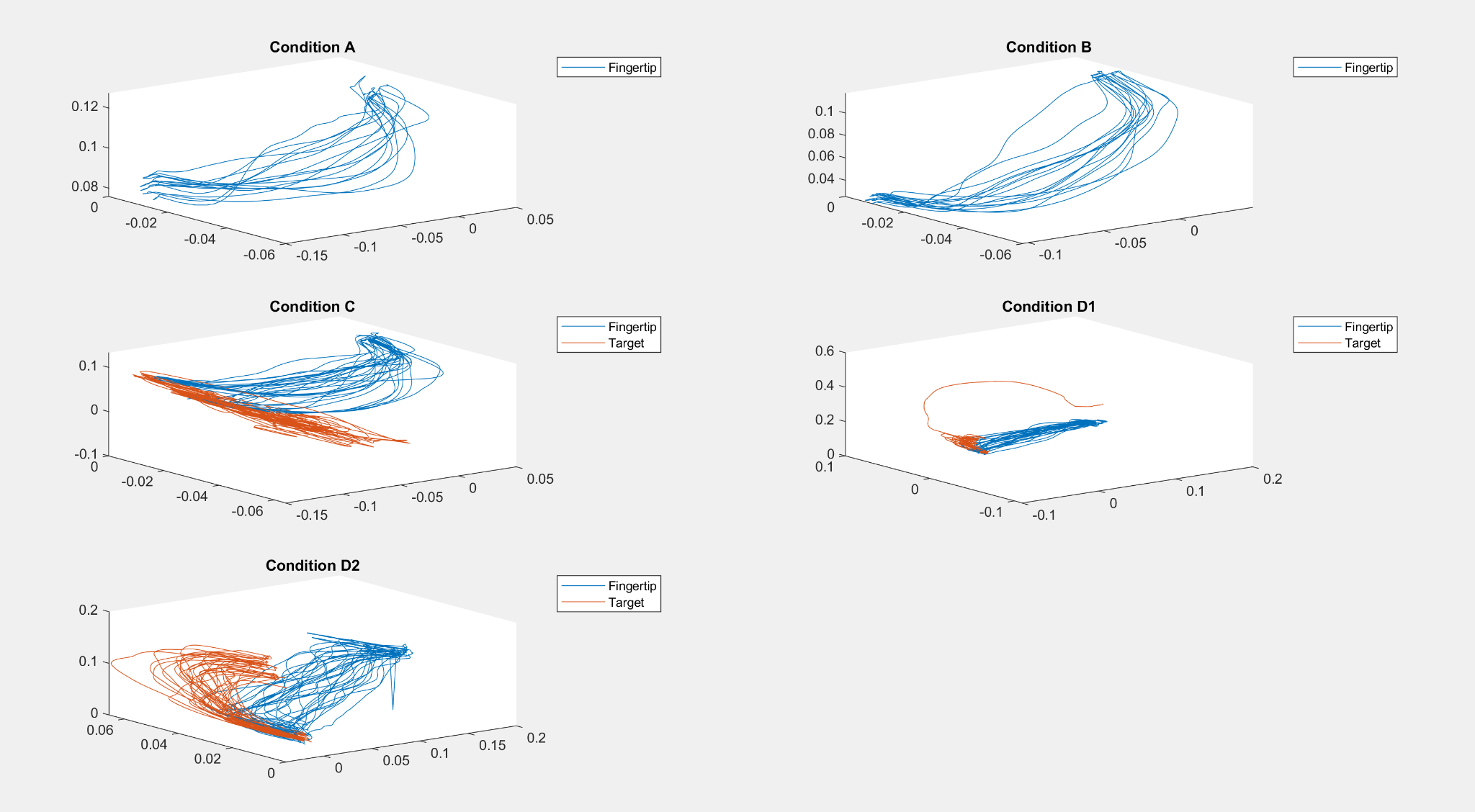
****

**Figure 24** is the positional data of Part D1 of the lab.

****

* **Figure 25** Shows the Velocity in the X direction Unfiltered, Velocity Filtered with Turning Points, 8 Reach Lines, and Max Velocity Lines
* Mean Peak Velocity is 4.853461e-03
* Standard Deviation is 3.158308e-03
* 
* **Figure 26** Shows Elbow Joint Angle, Angular Velocity, and EMGdata vs Time
* Mean distance is 1.184678e-01
* Standard Deviation is 5.187356e-02
* 
* **Figure 27** is the error sphere showing the variability of the point attempts, and a fitted ellipsoid corresponding and rotated to the data’s eigenvalues for Target 1
* absolute constant error for d2 0.118467792711645
* constant error is 0.115072721522727

**Fingertip and Target**

****

**Figure 28** Is the fingertip and target position data for all trials on one figure.

**Discussion/Conclusion**

From our results, we found that it was easier to point to the narrow targets, which contradicts our hypothesis. The standard deviation of our larger target was higher, suggesting that subjects were less consistent in their pointing with larger targets available. The average velocity was also higher in part B, which was from the narrow target again. This suggests that the subject was faster in pointing with a smaller target, which again contradicts both intuition and our initial hypothesis. Throughout this lab, we were plagued with issues syncing the EMG data with the Optitrack data, running the BPE script, filtering issues, markers not being recorded, and other technical issues. As a result, we had to rely on data collected from another group. This data did not include recorded positions for the targets, so this meant that it is not possible to assess the accuracy of the pointing between trials other than comparing the trials to themselves. According to these metrics, the trials using the narrow target were more precise, and possibly more accurate. For parts C and D, we predicted that the eyes being open would lead to faster, more precise, and more accurate pointing. In part D, we see that the error is reduced across both targets when compared to part C. There were several factors which might have led to these unexpected results. Firstly, there is the questionable quality of the data. Some segments of the data had to be heavily filtered in order to isolate turning points in the data, to the point of it heavily distorting the signal, but in addition, there were missing points from the Optitrack data as markers would constantly move out of the field of view of the cameras. On top of all the issues we were seeing with collecting data normally, there were issues with the syncing the EMG data using Optitrack, so we omitted the portions of this lab requiring synchronization for the sake of time. Another possible explanation for the “interesting results” we had might be the trials from A and C acting as a “warm-up” of sorts for the trials in B and D, which might have led to the increased consistency and speed that we observed. In conclusion, matlab is cringe. Use a real language like python + matplotlib + projectileboost + reverse mayo% + L + ratio + rocket return + parry + chargeback.

**Matlab Code:**

clc;

clear;

close all;

data = readtable("Data 2\Sahmet1A.csv");

data1 = readtable("Data 2\Sahmet1A.csv");

data1(1:4,:) = [];

data1(:,3:26) = [];

data1(:,15:17) = [];

Position = table2array(data1);

TimeA = Position(:,2);

X1 = Position(:,3);

Y1 = Position(:,4);

Z1 = Position(:,5);

X2 = Position(:,6);

Y2 = Position(:,7);

Z2 = Position(:,8);

X3 = Position(:,9);

Y3 = Position(:,10);

Z3 = Position(:,11);

X4 = Position(:,12);

Y4 = Position(:,13);

Z4 = Position(:,14);

figure

plot3(X1,Y1,Z1); %wrist

hold on

plot3(X2,Y2,Z2); %fingertip

plot3(X3,Y3,Z3); %shoulder

plot3(X4,Y4,Z4); %elbow

legend('Wrist', 'FingerTip', 'Shoulder', 'Elbow')

hold off

A = [X2,Y2,Z2]; %fingertip

V1A = [X3-X4, Y3-Y4, Z3-Z4]; % Shoulder to Elbow

V2A = [X4-X1, Y4-Y1, Z4-Z1]; % Elbow to Wrist

data = readtable("Data 2\Sahmet1B.csv");

data1 = readtable("Data 2\Sahmet1B.csv");

data1(1:4,:) = [];

data1(:,3:26) = [];

Position = table2array(data1);

TimeB = Position(:,2);

X1 = Position(:,3);

Y1 = Position(:,4);

Z1 = Position(:,5);

X2 = Position(:,6);

Y2 = Position(:,7);

Z2 = Position(:,8);

X3 = Position(:,9);

Y3 = Position(:,10);

Z3 = Position(:,11);

X4 = Position(:,12);

Y4 = Position(:,13);

Z4 = Position(:,14);

figure

plot3(X1,Y1,Z1); %elbow

hold on

plot3(X2,Y2,Z2); %wrist

plot3(X3,Y3,Z3); %fingertip

plot3(X4,Y4,Z4); %shoulder

legend('Elbow', 'Wrist', 'FingerTip', 'Shoulder')

hold off

B = [X3,Y3,Z3]; %fingertip

V1B = [X4-X1, Y4-Y1, Z4-Z1]; % Shoulder to Elbow

V2B = [X1-X2, Y1-Y2, Z1-Z2]; % Elbow to Wrist

data = readtable("Data 2\Sahmet2C.csv");

data1 = readtable("Data 2\Sahmet2C.csv");

data1(1:4,:) = [];

data1(:,3:26) = [];

data1(:,18:20) = [];

Position = table2array(data1);

TimeC = Position(:,2);

X1 = Position(:,3);

Y1 = Position(:,4);

Z1 = Position(:,5);

X2 = Position(:,6);

Y2 = Position(:,7);

Z2 = Position(:,8);

X3 = Position(:,9);

Y3 = Position(:,10);

Z3 = Position(:,11);

X4 = Position(:,12);

Y4 = Position(:,13);

Z4 = Position(:,14);

X5 = Position(:,15);

Y5 = Position(:,16);

Z5 = Position(:,17);

figure

plot3(X1,Y1,Z1); %Target

hold on

plot3(X2,Y2,Z2); %Shoulder

plot3(X3,Y3,Z3); %FingerTip

plot3(X4,Y4,Z4); %Wrist

plot3(X5,Y5,Z5); %Elbow

legend('Target', 'Shoulder', 'FingerTip', 'Wrist','Elbow')

hold off

C = [X3,Y3,Z3]; %fingertip

CT = [X1,Y1,Z1];%Target

V1C = [X2-X1, Y2-Y1, Z2-Z1]; % Shoulder to Elbow

V2C = [X4-X2, Y4-Y2, Z4-Z2]; % Elbow to Wrist

data = readtable("Data 2\Sahmet2DTarget1.csv");

data1 = readtable("Data 2\Sahmet2DTarget1.csv");

data1(1:4,:) = [];

data1(:,3:26) = [];

data1(:,18:20) = [];

Position = table2array(data1);

TimeD1 = Position(:,2);

X1 = Position(:,3);

Y1 = Position(:,4);

Z1 = Position(:,5);

X2 = Position(:,6);

Y2 = Position(:,7);

Z2 = Position(:,8);

X3 = Position(:,9);

Y3 = Position(:,10);

Z3 = Position(:,11);

X4 = Position(:,12);

Y4 = Position(:,13);

Z4 = Position(:,14);

X5 = Position(:,15);

Y5 = Position(:,16);

Z5 = Position(:,17);

figure

plot3(X1,Y1,Z1); %Elbow

hold on

plot3(X2,Y2,Z2); %Shoulder

plot3(X3,Y3,Z3); %Marker

plot3(X4,Y4,Z4); %FingerTip

plot3(X5,Y5,Z5); %Wrist

legend('Elbow', 'Shoulder', 'Target', 'FingerTip','Wrist')

hold off

D1 = [X4,Y4,Z4]; %fingertip

DT1 = [X3,Y3,Z3]; % Target

V1D1 = [X2-X1, Y2-Y1, Z2-Z1]; % Shoulder to Elbow

V2D1 = [X1-X5, Y1-Y5, Z1-Z5]; % Elbow to Wrist

data = readtable("Data 2\Sahmet2DTarget2.csv");

data1 = readtable("Data 2\Sahmet2DTarget2.csv");

data1(1:4,:) = [];

data1(:,3:26) = [];

Position = table2array(data1);

TimeD2 = Position(:,2);

X1 = Position(:,3);

Y1 = Position(:,4);

Z1 = Position(:,5);

X2 = Position(:,6);

Y2 = Position(:,7);

Z2 = Position(:,8);

X3 = Position(:,9);

Y3 = Position(:,10);

Z3 = Position(:,11);

X4 = Position(:,12);

Y4 = Position(:,13);

Z4 = Position(:,14);

X5 = Position(:,15);

Y5 = Position(:,16);

Z5 = Position(:,17);

figure

plot3(X1,Y1,Z1); %Shoulder

hold on

plot3(X2,Y2,Z2); %Elbow

plot3(X3,Y3,Z3); %Target

plot3(X4,Y4,Z4); %Wrist

plot3(X5,Y5,Z5); %Fingertip

legend('Shoulder', 'Elbow', 'Target', 'Wrist','Fingertip')

hold off

D2 = [X5,Y5,Z5]; %fingertip

DT2 = [X3,Y3,Z3]; %Marker

V1D2 = [X1-X2, Y1-Y2, Z1-Z2]; % Shoulder to Elbow

V2D2 = [X2-X4, Y2-Y4, Z2-Z4]; % Elbow to Wrist

D2(8432:8442,1) = 0.14;

D2(8432:8442,2) = 0.0245;

D2(8432:8442,3) = 0.0142;

figure

subplot(3,2,1)

plot3(A(:,1),A(:,2),A(:,3))

title('Condition A')

legend('Fingertip')

hold on

subplot(3,2,2)

plot3(B(:,1),B(:,2),B(:,3))

title('Condition B')

legend('Fingertip')

subplot(3,2,3)

plot3(C(:,1),C(:,2),C(:,3));

hold on

plot3(CT(:,1),CT(:,2),CT(:,3))

title('Condition C')

legend('Fingertip','Target')

subplot(3,2,4)

plot3(D1(:,1),D1(:,2),D1(:,3))

hold on

plot3(DT1(:,1),DT1(:,2),DT1(:,3))

title('Condition D1')

legend('Fingertip','Target')

subplot(3,2,5)

plot3(D2(:,1),D2(:,2),D2(:,3))

hold on

plot3(DT2(:,1),DT2(:,2),DT2(:,3))

title('Condition D2')

legend('Fingertip','Target')

%% cri ;(

close all

Xvalue = C(:,1);

Yvalue = C(:,2);

Zvalue = C(:,3);

Cd\_x = diff(C(:,1));

Cd\_y = diff(C(:,2));

Cd\_z = diff(C(:,3));

Cd\_t = diff(TimeC);

velC = Cd\_x./Cd\_t;

velCy = Cd\_y./Cd\_t;

velCz = Cd\_z./Cd\_t;

velC = velC';

velCy = velCy';

velCz = velCz';

velC = [velC(end),velC];

velCy = [velCy(end),velCy];

velCz = [velCz(end),velCz];

velC = velC';

velCy = velCy';

velCz = velCz';

figure

plot(TimeC,velC)

[A, B1] = butter(2, 50/500, "low");

velC = filtfilt(A, B1, velC);

turningPoints = [];

i = 1;

while i <= length(velC) - 1

% Check if current time is within desired range before evaluating velC

if TimeC(i) > 2.4 && TimeC(i) < 28.5

if (velC(i) <= 0 && velC(i + 1) > 0)||(velC(i) >= 0 && velC(i + 1) < 0) % Condition for turning point

turningPoints = [turningPoints; i + 1]; % Collect turning point

i = i + 75; %Search for turning Points in set interval

continue;

end

end

i = i + 1; % Otherwise, proceed to the next index

end

% Extract every other value

everyOther = Xvalue(turningPoints(1:2:end));

everyOther2 = Yvalue(turningPoints(1:2:end));

everyOther3 = Zvalue(turningPoints(1:2:end));

% Calculate the average

% average = mean(everyOther);

% average2 = mean(everyOther2);

% Extract every other value starting from the second element

everyOtherOpposite = flip(Xvalue(turningPoints(2:2:end)));

everyOtherOpposite2 = flip(Yvalue(turningPoints(2:2:end)));

everyOtherOpposite3 = flip(Zvalue(turningPoints(2:2:end)));

% Calculate the average

% averageOpposite = mean(everyOtherOpposite);

% averageOpposite2 = mean(everyOtherOpposite2);

% TY = [average, average2];

% TZ = [averageOpposite, averageOpposite2];

% turningPointsmean = plot(TY,TZ,'Color','g');

timeForvelC = TimeC(1:end); % Drop the first time value

% Plot Y-velocity over time

figure(3);

plot(timeForvelC, velC, 'LineWidth', 2);

xlabel('Time');

ylabel('Velocity in Y');

title('Y-Velocity Over Time');

hold on;

% Highlight turning points on the plot

figure(3);

scatter(timeForvelC(turningPoints), velC(turningPoints), 'r', 'filled');

turningPointsX = timeForvelC(turningPoints(1:end));

turningPointsX = diff(turningPointsX);

% labeling

legend('X-Velocity', 'Turning Points');

figure;

hold on;

i1 = 1;

addedLegend1 = false;

for i2 = 2:4:length(turningPoints)

h1 = plot3(C(turningPoints(i1):turningPoints(i2),1), ...

C(turningPoints(i1):turningPoints(i2),2), ...

C(turningPoints(i1):turningPoints(i2),3), 'Color','r');

if ~addedLegend1

set(h1, 'DisplayName', 'Target 1');

addedLegend1 = true;

else

set(h1, 'HandleVisibility', 'off');

end

i1 = i1+4;

end

legend show;

hold off

figure

hold on

i1 = 3;

addedLegend2 = false;

for i2 = 4:4:length(turningPoints)

h2 = plot3(C(turningPoints(i1):turningPoints(i2),1), ...

C(turningPoints(i1):turningPoints(i2),2), ...

C(turningPoints(i1):turningPoints(i2),3), 'Color','b');

if ~addedLegend2

set(h2, 'DisplayName', 'Target 2');

addedLegend2 = true;

else

set(h2, 'HandleVisibility', 'off');

end

i1 = i1+4;

end

legend show;

hold off

endpointfinger = turningPoints(1:2:end);

for i = 1:length(endpointfinger)

epf(i,1) = C(endpointfinger(i),1);

epf(i,2) = C(endpointfinger(i),2);

epf(i,3) = C(endpointfinger(i),3);

end

%finds peaks

[Onepiece, locationx] = findpeaks(-velC, 'MinPeakHeight', 0.3, 'MinPeakDistance', 50);

figure;

hold on;

% Plot velocity components

plot(1:length(velC), velC, 'b', 'DisplayName', 'Velocity Component X');

% Plot peak locations

plot(locationx, velC(locationx), 'bo', 'MarkerSize', 8, 'DisplayName', 'Peaks X');

xlabel('Time');

ylabel('Absolute Velocity');

title('Absolute Velocity and Peaks in X, Y, and Z Components');

hold off;

for i = 1:length(locationx)

magnV(i) = sqrt(velC(i)^2+velCy(i)^2+velCz(i)^2);

end

meanPeakV = mean(magnV);

stdPeakV = std(magnV);

fprintf('Mean Peak Velocity is %d\n', meanPeakV)

fprintf('Standard Deviation is %d\n', stdPeakV)

%% Elbow Angle Vs Time and angular velocity

close all

dot = sum(V1C .\* V2C, 2);

magV1 = sqrt(sum(V1C.^2, 2));

magV2 = sqrt(sum(V2C.^2, 2));

angrads = acos(dot./(magV1.\*magV2));

elbow\_angle = angrads\*(180/pi);

delbow\_ang = diff(angrads);

angularvel = delbow\_ang./Cd\_t;

angularvel = angularvel';

angularvel = [angularvel(end),angularvel];

angularvel = angularvel';

bicepdata = load("Data 2\SahmetBicep2C.mat");

tricepdata = load("Data 2\SahmetTricep2C.mat");

biceps = bicepdata.ch2data;

triceps = tricepdata.ch1data;

biceps = [biceps(end),biceps];

triceps = [triceps(end),triceps];

biceps = biceps(1:10:end);

triceps = triceps(1:10:end);

figure

plot(TimeC, elbow\_angle, 'b-');

title("Elbow Joint Angle vs Time");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

EMGTime = linspace(1,30,3001);

p = 1;

for i = 1:4:length(turningPoints)-1

figure

subplot(2,2,p)

hold on

plot(TimeC(turningPoints(i):turningPoints(i+1)), elbow\_angle(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Angle vs Time (Target 1)");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

hold off

p = p+1;

subplot(2,2,p)

hold on

plot(TimeC(turningPoints(i):turningPoints(i+1)), angularvel(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Anglular Velocity vs Time (Target 1)");

xlabel('Time (s)');

ylabel('Angular Velocity (rads/s)');

hold off

p = p+1;

if turningPoints(i) <= 3000

subplot(2,2,p)

hold on

plot(EMGTime(turningPoints(i):turningPoints(i+1)), biceps(turningPoints(i):turningPoints(i+1)), 'b-');

plot(EMGTime(turningPoints(i):turningPoints(i+1)), triceps(turningPoints(i):turningPoints(i+1)), 'r-');

title("EMGdata vs Time (Target 1)");

xlabel('Time (s)');

ylabel('Voltage');

legend('Biceps','Triceps')

hold off

else

continue

end

p = 1;

end

p = 1;

for i = 3:4:length(turningPoints)-1

figure

subplot(2,2,p)

hold on

plot(TimeC(turningPoints(i):turningPoints(i+1)), elbow\_angle(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Angle vs Time (Target 2)");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

hold off

p = p+1;

subplot(2,2,p)

hold on

plot(TimeC(turningPoints(i):turningPoints(i+1)), angularvel(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Anglular Velocity vs Time (Target 2)");

xlabel('Time (s)');

ylabel('Angular Velocity (rads/s)');

hold off

p = p+1;

if turningPoints(i) <= 3000

subplot(2,2,p)

hold on

plot(EMGTime(turningPoints(i):turningPoints(i+1)), biceps(turningPoints(i):turningPoints(i+1)), 'b-');

plot(EMGTime(turningPoints(i):turningPoints(i+1)), triceps(turningPoints(i):turningPoints(i+1)), 'r-');

title("EMGdata vs Time (Target 2)");

xlabel('Time (s)');

ylabel('Voltage');

legend('Biceps','Triceps')

hold off

else

continue

end

p = 1;

end

figure

hold on

yyaxis left

plot(TimeC,angularvel, 'b-')

ylabel('Angular Velocity (rad/s)')

yyaxis right

plot(EMGTime,biceps, 'r-')

plot(EMGTime(1:length(triceps)),triceps, 'k-')

title('EMG Data & Angular Velocity vs Time')

ylabel('Voltage')

xlabel('Time (s)')

legend('Angular Velocity', 'Biceps Data', 'Triceps Data')

%% cri 2 ;(

close all

XTvalue = CT(:,1);

YTvalue = CT(:,2);

ZTvalue = CT(:,3);

Cd\_x = diff(CT(:,1));

Cd\_t = diff(TimeC);

velC = Cd\_x./Cd\_t;

velC = velC';

velC = [velC(end),velC];

velC = velC';

figure

plot(TimeC,velC)

[A, B1] = butter(2, 50/500, "low");

velC = filtfilt(A, B1, velC);

turningPoints = [];

i = 1;

while i <= length(velC) - 1

% Check if current time is within desired range before evaluating velC

if TimeC(i) > 1.6 && TimeC(i) < 29.5

if (velC(i) <= 0 && velC(i + 1) > 0)||(velC(i) >= 0 && velC(i + 1) < 0) % Condition for turning point

turningPoints = [turningPoints; i + 1]; % Collect turning point

i = i + 60; %Search for turning Points in set interval

continue;

end

end

i = i + 1; % Otherwise, proceed to the next index

end

% Extract every other value

everyOther = Xvalue(turningPoints(1:2:end));

everyOther2 = Yvalue(turningPoints(1:2:end));

everyOther3 = Zvalue(turningPoints(1:2:end));

% Calculate the average

% average = mean(everyOther);

% average2 = mean(everyOther2);

% Extract every other value starting from the second element

everyOtherOpposite = flip(Xvalue(turningPoints(2:2:end)));

everyOtherOpposite2 = flip(Yvalue(turningPoints(2:2:end)));

everyOtherOpposite3 = flip(Zvalue(turningPoints(2:2:end)));

% Calculate the average

% averageOpposite = mean(everyOtherOpposite);

% averageOpposite2 = mean(everyOtherOpposite2);

% TY = [average, average2];

% TZ = [averageOpposite, averageOpposite2];

% turningPointsmean = plot(TY,TZ,'Color','g');

timeForvelC = TimeC(1:end); % Drop the first time value

% Plot Y-velocity over time

figure(3);

plot(timeForvelC, velC, 'LineWidth', 2);

xlabel('Time');

ylabel('Velocity in Y');

title('Y-Velocity Over Time');

hold on;

% Highlight turning points on the plot

figure(3);

scatter(timeForvelC(turningPoints), velC(turningPoints), 'r', 'filled');

turningPointsX = timeForvelC(turningPoints(1:end));

turningPointsX = diff(turningPointsX);

% labeling

legend('X-Velocity', 'Turning Points');

figure;

hold on;

i1 = 1;

addedLegend1 = false;

for i2 = 2:4:length(turningPoints)

h1 = plot3(CT(turningPoints(i1):turningPoints(i2),1), ...

CT(turningPoints(i1):turningPoints(i2),2), ...

CT(turningPoints(i1):turningPoints(i2),3), 'Color','r');

if ~addedLegend1

set(h1, 'DisplayName', 'Target 1');

addedLegend1 = true;

else

set(h1, 'HandleVisibility', 'off');

end

i1 = i1+4;

end

i1 = 3;

addedLegend2 = false;

for i2 = 4:4:length(turningPoints)

h2 = plot3(CT(turningPoints(i1):turningPoints(i2),1), ...

CT(turningPoints(i1):turningPoints(i2),2), ...

CT(turningPoints(i1):turningPoints(i2),3), 'Color','b');

if ~addedLegend2

set(h2, 'DisplayName', 'Target 2');

addedLegend2 = true;

else

set(h2, 'HandleVisibility', 'off');

end

i1 = i1+4;

end

legend show;

hold off

endpointtarget = turningPoints(2:2:end);

for i = 1:length(endpointtarget)

ept(i,1) = CT(endpointtarget(i),1);

ept(i,2) = CT(endpointtarget(i),2);

ept(i,3) = CT(endpointtarget(i),3);

end

figure

hold on

scatter3(epf(1:2:end,1), epf(1:2:end,2), epf(1:2:end,3))

scatter3(ept(1:2:end,1), ept(1:2:end,2), ept(1:2:end,3))

title('Scatter Plot of Targets and Movement Reversals');

legend("Finger Endpoints", "Target Endpoints")

grid on

hold off

figure

hold on

scatter3(epf(2:2:end,1), epf(2:2:end,2), epf(2:2:end,3))

scatter3(ept(2:2:end,1), ept(2:2:end,2), ept(2:2:end,3))

title('Scatter Plot of Targets and Movement Reversals');

legend("Finger Endpoints", "Target Endpoints")

grid on

hold off

for i = 1:size(epf,1)

disx(i) = epf(i,1)-ept(i,1);

disy(i) = epf(i,2)-ept(i,2);

disz(i) = epf(i,3)-ept(i,3);

dis(i) = sqrt(disx(i)^2+disy(i)^2+disz(i)^2);

end

dismean = mean(dis);

disstd = std(dis);

fprintf('Mean distance is %d\n', dismean)

fprintf('Standard Deviation is %d\n', disstd)

p = 1;

for i = 1:2:size(epf,1)

epf1(p,1) = epf(i,1);

epf1(p,2) = epf(i,2);

epf1(p,3) = epf(i,3);

p = p+1;

end

p = 1;

for i = 2:2:size(epf,1)

epf2(p,1) = epf(i,1);

epf2(p,2) = epf(i,2);

epf2(p,3) = epf(i,3);

p = p+1;

end

p = 1;

for i = 1:2:size(ept,1)

ept1(p,1) = ept(i,1);

ept1(p,2) = ept(i,2);

ept1(p,3) = ept(i,3);

p = p+1;

end

p = 1;

for i = 2:2:size(ept,1)

ept2(p,1) = ept(i,1);

ept2(p,2) = ept(i,2);

ept2(p,3) = ept(i,3);

p = p+1;

end

%%

close all

% Define the target point

target = [mean(ept1(:,1)), mean(ept1(:,2)), mean(ept1(:,3))];

X = epf1(:,1);

Y = epf1(:,2);

Z = epf1(:,3);

Xt = ept1(:,1);

Yt = ept1(:,2);

Zt = ept1(:,3);

% 1. Absolute Constant Error

distancesToTarget = sqrt((X - Xt).^2 + (Y - Yt).^2 + (Z - Zt).^2);

absoluteConstantError = mean(distancesToTarget);

X = (X-Xt)+target(1);

Y = (Y-Yt)+target(2);

Z = (Z-Zt)+target(3);

% 2. Constant Error

dataCenter = [mean(X), mean(Y), mean(Z)];

constantError = norm(dataCenter - target);

% 3. Variable Error

stdX = std(X);

stdY = std(Y);

stdZ = std(Z);

variableError = sqrt(stdX^2 + stdY^2 + stdZ^2);

% Perform PCA to get the eigenvectors and eigenvalues

[coeff, ~, ~] = pca([X, Y, Z]);

dataCenter = mean([X, Y, Z]);

% Plot data, sphere, and ellipsoid

figure;

hold on;

plot3(X, Y, Z, 'ro');

[sphereX, sphereY, sphereZ] = sphere;

radius = variableError;

sphereX = radius \* sphereX + dataCenter(1);

sphereY = radius \* sphereY + dataCenter(2);

sphereZ = radius \* sphereZ + dataCenter(3);

surf(sphereX, sphereY, sphereZ, 'FaceAlpha', 0.3, 'EdgeColor', 'none');

% Plot ellipsoid

scale = [stdX, stdY, stdZ]; % Scaling factors for the ellipsoid

[ellipsoidX, ellipsoidY, ellipsoidZ] = ellipsoid(0, 0, 0, scale(1), scale(2), scale(3), 50);

transformedEllipsoid = [ellipsoidX(:), ellipsoidY(:), ellipsoidZ(:)] \* coeff;

ellipsoidX = reshape(transformedEllipsoid(:,1), size(ellipsoidX)) + dataCenter(1);

ellipsoidY = reshape(transformedEllipsoid(:,2), size(ellipsoidY)) + dataCenter(2);

ellipsoidZ = reshape(transformedEllipsoid(:,3), size(ellipsoidZ)) + dataCenter(3);

surf(ellipsoidX, ellipsoidY, ellipsoidZ, 'FaceColor', 'green', 'FaceAlpha', 0.1);

% Plot eigenvectors as arrows starting from the center of the data

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,1)\*scale(1), coeff(2,1)\*scale(1), coeff(3,1)\*scale(1), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,2)\*scale(2), coeff(2,2)\*scale(2), coeff(3,2)\*scale(2), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,3)\*scale(3), coeff(2,3)\*scale(3), coeff(3,3)\*scale(3), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

xlabel('X');

ylabel('Y');

zlabel('Z');

title('Data Points, Error Sphere, Fitted Ellipsoid, and PCA Eigenvectors');

legend('Data Points', 'Error Sphere', 'Fitted Ellipsoid', 'PCA Vectors');

axis equal;

grid on;

hold off;

% Define the target point

target = [mean(ept2(:,1)), mean(ept2(:,2)), mean(ept2(:,3))];

X = epf2(:,1);

Y = epf2(:,2);

Z = epf2(:,3);

Xt = ept2(:,1);

Yt = ept2(:,2);

Zt = ept2(:,3);

% 1. Absolute Constant Error

distancesToTarget = sqrt((X - Xt).^2 + (Y - Yt).^2 + (Z - Zt).^2);

absoluteConstantError = mean(distancesToTarget);

X = (X-Xt)+target(1);

Y = (Y-Yt)+target(2);

Z = (Z-Zt)+target(3);

% 2. Constant Error

dataCenter = [mean(X), mean(Y), mean(Z)];

constantError = norm(dataCenter - target);

% 3. Variable Error

stdX = std(X);

stdY = std(Y);

stdZ = std(Z);

variableError = sqrt(stdX^2 + stdY^2 + stdZ^2);

% Perform PCA to get the eigenvectors and eigenvalues

[coeff, ~, ~] = pca([X, Y, Z]);

dataCenter = mean([X, Y, Z]);

% Plot data, sphere, and ellipsoid

figure;

hold on;

plot3(X, Y, Z, 'ro');

[sphereX, sphereY, sphereZ] = sphere;

radius = variableError;

sphereX = radius \* sphereX + dataCenter(1);

sphereY = radius \* sphereY + dataCenter(2);

sphereZ = radius \* sphereZ + dataCenter(3);

surf(sphereX, sphereY, sphereZ, 'FaceAlpha', 0.3, 'EdgeColor', 'none');

% Plot ellipsoid

scale = [stdX, stdY, stdZ]; % Scaling factors for the ellipsoid

[ellipsoidX, ellipsoidY, ellipsoidZ] = ellipsoid(0, 0, 0, scale(1), scale(2), scale(3), 50);

transformedEllipsoid = [ellipsoidX(:), ellipsoidY(:), ellipsoidZ(:)] \* coeff;

ellipsoidX = reshape(transformedEllipsoid(:,1), size(ellipsoidX)) + dataCenter(1);

ellipsoidY = reshape(transformedEllipsoid(:,2), size(ellipsoidY)) + dataCenter(2);

ellipsoidZ = reshape(transformedEllipsoid(:,3), size(ellipsoidZ)) + dataCenter(3);

surf(ellipsoidX, ellipsoidY, ellipsoidZ, 'FaceColor', 'green', 'FaceAlpha', 0.1);

% Plot eigenvectors as arrows starting from the center of the data

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,1)\*scale(1), coeff(2,1)\*scale(1), coeff(3,1)\*scale(1), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,2)\*scale(2), coeff(2,2)\*scale(2), coeff(3,2)\*scale(2), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,3)\*scale(3), coeff(2,3)\*scale(3), coeff(3,3)\*scale(3), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

xlabel('X');

ylabel('Y');

zlabel('Z');

title('Data Points, Error Sphere, Fitted Ellipsoid, and PCA Eigenvectors');

legend('Data Points', 'Error Sphere', 'Fitted Ellipsoid', 'PCA Vectors');

axis equal;

grid on;

hold off;

%% For A

close all

Xvalue = A(:,1);

Yvalue = A(:,2);

Zvalue = A(:,3);

Cd\_x = diff(A(:,1));

Cd\_y = diff(A(:,2));

Cd\_z = diff(A(:,3));

Cd\_t = diff(TimeA);

velC = Cd\_x./Cd\_t;

velCy = Cd\_y./Cd\_t;

velCz = Cd\_z./Cd\_t;

velC = velC';

velCy = velCy';

velCz = velCz';

velC = [velC(end),velC];

velCy = [velCy(end),velCy];

velCz = [velCz(end),velCz];

velC = velC';

velCy = velCy';

velCz = velCz';

figure

plot(TimeA,velC)

[A1, B1] = butter(2, 50/500, "low");

velC = filtfilt(A1, B1, velC);

turningPoints = [];

i = 1;

while i <= length(velC) - 1

% Check if current time is within desired range before evaluating velC

if TimeA(i) > 0.7 && TimeA(i) < 8.8

if (velC(i) <= 0 && velC(i + 1) > 0)||(velC(i) >= 0 && velC(i + 1) < 0) % Condition for turning point

turningPoints = [turningPoints; i + 1]; % Collect turning point

i = i + 40; %Search for turning Points in set interval

continue;

end

end

i = i + 1; % Otherwise, proceed to the next index

end

% Extract every other value

everyOther = Xvalue(turningPoints(1:2:end));

everyOther2 = Yvalue(turningPoints(1:2:end));

everyOther3 = Zvalue(turningPoints(1:2:end));

% Calculate the average

% average = mean(everyOther);

% average2 = mean(everyOther2);

% Extract every other value starting from the second element

everyOtherOpposite = flip(Xvalue(turningPoints(2:2:end)));

everyOtherOpposite2 = flip(Yvalue(turningPoints(2:2:end)));

everyOtherOpposite3 = flip(Zvalue(turningPoints(2:2:end)));

% Calculate the average

% averageOpposite = mean(everyOtherOpposite);

% averageOpposite2 = mean(everyOtherOpposite2);

% TY = [average, average2];

% TZ = [averageOpposite, averageOpposite2];

% turningPointsmean = plot(TY,TZ,'Color','g');

timeForvelC = TimeA(1:end); % Drop the first time value

% Plot Y-velocity over time

figure;

plot(timeForvelC, velC, 'LineWidth', 2);

xlabel('Time');

ylabel('Velocity in Y');

title('Y-Velocity Over Time');

hold on;

% Highlight turning points on the plot

scatter(timeForvelC(turningPoints), velC(turningPoints), 'r', 'filled');

turningPointsX = timeForvelC(turningPoints(1:end));

turningPointsX = diff(turningPointsX);

% labeling

legend('X-Velocity', 'Turning Points');

figure;

hold on;

i1 = 1;

addedLegend1 = false;

for i2 = 2:2:length(turningPoints)

h1 = plot3(A(turningPoints(i1):turningPoints(i2),1), ...

A(turningPoints(i1):turningPoints(i2),2), ...

A(turningPoints(i1):turningPoints(i2),3), 'Color','r');

if ~addedLegend1

set(h1, 'DisplayName', 'Condition A');

addedLegend1 = true;

else

set(h1, 'HandleVisibility', 'off');

end

i1 = i1+2;

end

legend show;

hold off

endpointfinger = turningPoints(1:2:end);

for i = 1:length(endpointfinger)

epf(i,1) = C(endpointfinger(i),1);

epf(i,2) = C(endpointfinger(i),2);

epf(i,3) = C(endpointfinger(i),3);

end

%finds peaks

[Onepiece, locationx] = findpeaks(-velC, 'MinPeakHeight', 0.3, 'MinPeakDistance', 50);

figure;

hold on;

% Plot velocity components

plot(1:length(velC), velC, 'b', 'DisplayName', 'Velocity Component X');

% Plot peak locations

plot(locationx, velC(locationx), 'bo', 'MarkerSize', 8, 'DisplayName', 'Peaks X');

xlabel('Time');

ylabel('Absolute Velocity');

title('Absolute Velocity and Peaks in X, Y, and Z Components');

hold off;

for i = 1:length(locationx)

magnV(i) = sqrt(velC(i)^2+velCy(i)^2+velCz(i)^2);

end

meanPeakV = mean(magnV(1:end-1));

stdPeakV = std(magnV(1:end-1));

fprintf('Mean Peak Velocity is %d\n', meanPeakV)

fprintf('Standard Deviation is %d\n', stdPeakV)

%% Elbow Angle Vs Time and angular velocity

close all

dot = sum(V1A .\* V2A, 2);

magV1 = sqrt(sum(V1A.^2, 2));

magV2 = sqrt(sum(V2A.^2, 2));

angrads = acos(dot./(magV1.\*magV2));

elbow\_angle = angrads\*(180/pi);

delbow\_ang = diff(angrads);

angularvel = delbow\_ang./Cd\_t;

angularvel = angularvel';

angularvel = [angularvel(end),angularvel];

angularvel = angularvel';

bicepdata = load("Data 2\SahmetBicep1A.mat");

tricepdata = load("Data 2\SahmetTricep1A.mat");

biceps = bicepdata.ch2data;

triceps = tricepdata.ch1data;

biceps = [biceps(end),biceps];

triceps = [triceps(end),triceps];

biceps = biceps(1:10:end);

triceps = triceps(1:10:end);

figure

plot(TimeA, elbow\_angle, 'b-');

title("Elbow Joint Angle vs Time");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

EMGTime = linspace(1,10,1001);

p = 1;

for i = 1:2:length(turningPoints)-1

figure

subplot(2,2,p)

hold on

plot(TimeA(turningPoints(i):turningPoints(i+1)), elbow\_angle(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Angle vs Time");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

hold off

p = p+1;

subplot(2,2,p)

hold on

plot(TimeA(turningPoints(i):turningPoints(i+1)), angularvel(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Anglular Velocity vs Time");

xlabel('Time (s)');

ylabel('Angular Velocity (rads/s)');

hold off

p = p+1;

if turningPoints(i) <= 1000

subplot(2,2,p)

hold on

plot(EMGTime(turningPoints(i):turningPoints(i+1)), biceps(turningPoints(i):turningPoints(i+1)), 'b-');

plot(EMGTime(turningPoints(i):turningPoints(i+1)), triceps(turningPoints(i):turningPoints(i+1)), 'r-');

title("EMGdata vs Time");

xlabel('Time (s)');

ylabel('Voltage');

legend('Biceps','Triceps')

hold off

else

continue

end

p = 1;

end

figure

hold on

yyaxis left

plot(TimeA,angularvel, 'b-')

ylabel('Angular Velocity (rad/s)')

yyaxis right

plot(EMGTime,biceps, 'r-')

plot(EMGTime(1:length(triceps)),triceps, 'k-')

title('EMG Data & Angular Velocity vs Time')

ylabel('Voltage')

xlabel('Time (s)')

legend('Angular Velocity', 'Biceps Data', 'Triceps Data')

%% For B

close all

Xvalue = B(:,1);

Yvalue = B(:,2);

Zvalue = B(:,3);

Bd\_x = diff(B(:,1));

Bd\_y = diff(B(:,2));

Bd\_z = diff(B(:,3));

Bd\_t = diff(TimeB);

velB = Bd\_x./Bd\_t;

velBy = Bd\_y./Bd\_t;

velBz = Bd\_z./Bd\_t;

velB = velB';

velBy = velBy';

velBz = velBz';

velB = [velB(end),velB];

velBy = [velBy(end),velBy];

velBz = [velBz(end),velBz];

velB = velB';

velBy = velBy';

velBz = velBz';

figure

plot(TimeB,velB)

[A1, B1] = butter(2, 50/500, "low");

velB = filtfilt(A1, B1, velB);

turningPoints = [];

i = 1;

while i <= length(velB) - 1

% Check if current time is within desired range before evaluating velC

if TimeB(i) > 0.6 && TimeB(i) < 9.5

if (velB(i) <= 0 && velB(i + 1) > 0)||(velB(i) >= 0 && velB(i + 1) < 0) % Condition for turning point

turningPoints = [turningPoints; i + 1]; % Collect turning point

i = i + 40; %Search for turning Points in set interval

continue;

end

end

i = i + 1; % Otherwise, proceed to the next index

end

% Extract every other value

everyOther = Xvalue(turningPoints(1:2:end));

everyOther2 = Yvalue(turningPoints(1:2:end));

everyOther3 = Zvalue(turningPoints(1:2:end));

% Calculate the average

% average = mean(everyOther);

% average2 = mean(everyOther2);

% Extract every other value starting from the second element

everyOtherOpposite = flip(Xvalue(turningPoints(2:2:end)));

everyOtherOpposite2 = flip(Yvalue(turningPoints(2:2:end)));

everyOtherOpposite3 = flip(Zvalue(turningPoints(2:2:end)));

% Calculate the average

% averageOpposite = mean(everyOtherOpposite);

% averageOpposite2 = mean(everyOtherOpposite2);

% TY = [average, average2];

% TZ = [averageOpposite, averageOpposite2];

% turningPointsmean = plot(TY,TZ,'Color','g');

timeForvelB = TimeB(1:end); % Drop the first time value

% Plot Y-velocity over time

figure;

plot(timeForvelB, velB, 'LineWidth', 2);

xlabel('Time');

ylabel('Velocity in Y');

title('Y-Velocity Over Time');

hold on;

% Highlight turning points on the plot

scatter(timeForvelB(turningPoints), velB(turningPoints), 'r', 'filled');

turningPointsX = timeForvelB(turningPoints(1:end));

turningPointsX = diff(turningPointsX);

% labeling

legend('X-Velocity', 'Turning Points');

figure;

hold on;

i1 = 1;

addedLegend1 = false;

for i2 = 2:2:length(turningPoints)

h1 = plot3(B(turningPoints(i1):turningPoints(i2),1), ...

B(turningPoints(i1):turningPoints(i2),2), ...

B(turningPoints(i1):turningPoints(i2),3), 'Color','r');

if ~addedLegend1

set(h1, 'DisplayName', 'Condition A');

addedLegend1 = true;

else

set(h1, 'HandleVisibility', 'off');

end

i1 = i1+2;

end

legend show;

hold off

endpointfinger = turningPoints(1:2:end);

for i = 1:length(endpointfinger)

epf(i,1) = B(endpointfinger(i),1);

epf(i,2) = B(endpointfinger(i),2);

epf(i,3) = B(endpointfinger(i),3);

end

%finds peaks

[Onepiece, locationx] = findpeaks(-velB, 'MinPeakHeight', 0.3, 'MinPeakDistance', 50);

figure;

hold on;

% Plot velocity components

plot(1:length(velB), velB, 'b', 'DisplayName', 'Velocity Component X');

% Plot peak locations

plot(locationx, velB(locationx), 'bo', 'MarkerSize', 8, 'DisplayName', 'Peaks X');

xlabel('Time');

ylabel('Absolute Velocity');

title('Absolute Velocity and Peaks in X, Y, and Z Components');

hold off;

for i = 1:length(locationx)

magnV(i) = sqrt(velB(i)^2+velBy(i)^2+velBz(i)^2);

end

meanPeakV = mean(magnV(1:end-1));

stdPeakV = std(magnV(1:end-1));

fprintf('Mean Peak Velocity is %d\n', meanPeakV)

fprintf('Standard Deviation is %d\n', stdPeakV)

%% Elbow Angle Vs Time and angular velocity

close all

dot = sum(V1B .\* V2B, 2);

magV1 = sqrt(sum(V1B.^2, 2));

magV2 = sqrt(sum(V2B.^2, 2));

angrads = acos(dot./(magV1.\*magV2));

elbow\_angle = angrads\*(180/pi);

delbow\_ang = diff(angrads);

angularvel = delbow\_ang./Bd\_t;

angularvel = angularvel';

angularvel = [angularvel(end),angularvel];

angularvel = angularvel';

bicepdata = load("Data 2/SahmetBicep1B.mat");

tricepdata = load("Data 2/SahmetTricep1B.mat");

biceps = bicepdata.ch2data;

triceps = tricepdata.ch1data;

biceps = [biceps(end),biceps];

triceps = [triceps(end),triceps];

biceps = biceps(1:10:end);

triceps = triceps(1:10:end);

figure

plot(TimeB, elbow\_angle, 'b-');

title("Elbow Joint Angle vs Time");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

EMGTime = linspace(1,10,1001);

p = 1;

for i = 1:2:length(turningPoints)-1

figure

subplot(2,2,p)

hold on

plot(TimeB(turningPoints(i):turningPoints(i+1)), elbow\_angle(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Angle vs Time");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

hold off

p = p+1;

subplot(2,2,p)

hold on

plot(TimeB(turningPoints(i):turningPoints(i+1)), angularvel(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Anglular Velocity vs Time");

xlabel('Time (s)');

ylabel('Angular Velocity (rads/s)');

hold off

p = p+1;

if turningPoints(i+1) <= 1000

subplot(2,2,p)

hold on

plot(EMGTime(turningPoints(i):turningPoints(i+1)), biceps(turningPoints(i):turningPoints(i+1)), 'b-');

plot(EMGTime(turningPoints(i):turningPoints(i+1)), triceps(turningPoints(i):turningPoints(i+1)), 'r-');

title("EMGdata vs Time");

xlabel('Time (s)');

ylabel('Voltage');

legend('Biceps','Triceps')

hold off

else

continue

end

p = 1;

end

figure

hold on

yyaxis left

plot(TimeB,angularvel, 'b-')

ylabel('Angular Velocity (rad/s)')

yyaxis right

plot(EMGTime,biceps, 'r-')

plot(EMGTime(1:length(triceps)),triceps, 'k-')

title('EMG Data & Angular Velocity vs Time')

ylabel('Voltage')

xlabel('Time (s)')

legend('Angular Velocity', 'Biceps Data', 'Triceps Data')

%% For D1

close all

Xvalue = D1(:,1);

Yvalue = D1(:,2);

Zvalue = D1(:,3);

Cd\_x = diff(D1(:,1));

Cd\_y = diff(D1(:,2));

Cd\_z = diff(D1(:,3));

Cd\_t = diff(TimeD1);

velC = Cd\_x./Cd\_t;

velCy = Cd\_y./Cd\_t;

velCz = Cd\_z./Cd\_t;

velC = velC';

velCy = velCy';

velCz = velCz';

velC = [velC(end),velC];

velCy = [velCy(end),velCy];

velCz = [velCz(end),velCz];

velC = velC';

velCy = velCy';

velCz = velCz';

figure

plot(TimeD1,velC)

[A, B1] = butter(2, 50/500, "low");

velC = filtfilt(A, B1, velC);

turningPoints = [];

i = 1;

while i <= length(velC) - 1

% Check if current time is within desired range before evaluating velC

if TimeD1(i) > 5.416 && TimeD1(i) < 68.3

if abs((velC(i)-velC(i+10))) >= 0.05||abs((velC(i)-velC(i-10))) >= 0.05

if (velC(i) <= 0.01 && velC(i + 1) > -0.01)||(velC(i) >= -0.01 && velC(i + 1) < 0.01) % Condition for turning point

turningPoints = [turningPoints; i + 1]; % Collect turning point

i = i + 50; %Search for turning Points in set interval

continue;

end

end

end

i = i + 1; % Otherwise, proceed to the next index

end

% Extract every other value

everyOther = Xvalue(turningPoints(1:2:end));

everyOther2 = Yvalue(turningPoints(1:2:end));

everyOther3 = Zvalue(turningPoints(1:2:end));

% Calculate the average

% average = mean(everyOther);

% average2 = mean(everyOther2);

% Extract every other value starting from the second element

everyOtherOpposite = flip(Xvalue(turningPoints(2:2:end)));

everyOtherOpposite2 = flip(Yvalue(turningPoints(2:2:end)));

everyOtherOpposite3 = flip(Zvalue(turningPoints(2:2:end)));

% Calculate the average

% averageOpposite = mean(everyOtherOpposite);

% averageOpposite2 = mean(everyOtherOpposite2);

% TY = [average, average2];

% TZ = [averageOpposite, averageOpposite2];

% turningPointsmean = plot(TY,TZ,'Color','g');

timeForvelC = TimeD1(1:end); % Drop the first time value

% Plot Y-velocity over time

figure(3);

plot(timeForvelC, velC, 'LineWidth', 2);

xlabel('Time');

ylabel('Velocity in Y');

title('Y-Velocity Over Time');

hold on;

% Highlight turning points on the plot

figure(3);

scatter(timeForvelC(turningPoints), velC(turningPoints), 'r', 'filled');

turningPointsX = timeForvelC(turningPoints(1:end));

turningPointsX = diff(turningPointsX);

% labeling

legend('X-Velocity', 'Turning Points');

figure;

hold on;

i1 = 1;

addedLegend1 = false;

for i2 = 2:3:length(turningPoints)

h1 = plot3(D1(turningPoints(i1):turningPoints(i2),1), ...

D1(turningPoints(i1):turningPoints(i2),2), ...

D1(turningPoints(i1):turningPoints(i2),3), 'Color','r');

if ~addedLegend1

set(h1, 'DisplayName', 'Target 1');

addedLegend1 = true;

else

set(h1, 'HandleVisibility', 'off');

end

i1 = i1+3;

end

legend show;

hold off

endpointfinger = turningPoints(2:3:end);

for i = 1:length(endpointfinger)

epf(i,1) = D1(endpointfinger(i),1);

epf(i,2) = D1(endpointfinger(i),2);

epf(i,3) = D1(endpointfinger(i),3);

end

%finds peaks

[Onepiece, locationx] = findpeaks(-velC, 'MinPeakHeight', 0.3, 'MinPeakDistance', 50);

figure;

hold on;

% Plot velocity components

plot(1:length(velC), velC, 'b', 'DisplayName', 'Velocity Component X');

% Plot peak locations

plot(locationx, velC(locationx), 'bo', 'MarkerSize', 8, 'DisplayName', 'Peaks X');

xlabel('Time');

ylabel('Absolute Velocity');

title('Absolute Velocity and Peaks in X, Y, and Z Components');

hold off;

for i = 1:length(locationx)

magnV(i) = sqrt(velC(i)^2+velCy(i)^2+velCz(i)^2);

end

meanPeakV = mean(magnV);

stdPeakV = std(magnV);

fprintf('Mean Peak Velocity is %d\n', meanPeakV)

fprintf('Standard Deviation is %d\n', stdPeakV)

%% Elbow Angle Vs Time and angular velocity

close all

dot = sum(V1D1 .\* V2D1, 2);

magV1 = sqrt(sum(V1D1.^2, 2));

magV2 = sqrt(sum(V2D1.^2, 2));

angrads = acos(dot./(magV1.\*magV2));

elbow\_angle = angrads\*(180/pi);

delbow\_ang = diff(angrads);

angularvel = delbow\_ang./Cd\_t;

angularvel = angularvel';

angularvel = [angularvel(end),angularvel];

angularvel = angularvel';

bicepdata = load("Data 2\SahmetBicep2DTarget2.mat");

tricepdata = load("Data 2\SahmetTricep2DTarget2.mat");

biceps = bicepdata.ch2data;

triceps = tricepdata.ch1data;

biceps = [biceps(end),biceps];

triceps = [triceps(end),triceps];

biceps = biceps(1:10:end);

triceps = triceps(1:10:end);

figure

plot(TimeD1, elbow\_angle, 'b-');

title("Elbow Joint Angle vs Time");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

EMGTime = linspace(1,80,8001);

p = 1;

for i = 1:3:length(turningPoints)-1

figure

subplot(2,2,p)

hold on

plot(TimeD1(turningPoints(i):turningPoints(i+1)), elbow\_angle(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Angle vs Time (Target 1)");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

hold off

p = p+1;

subplot(2,2,p)

hold on

plot(TimeD1(turningPoints(i):turningPoints(i+1)), angularvel(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Anglular Velocity vs Time (Target 1)");

xlabel('Time (s)');

ylabel('Angular Velocity (rads/s)');

hold off

p = p+1;

if turningPoints(i+1) <= 8000

subplot(2,2,p)

hold on

plot(EMGTime(turningPoints(i):turningPoints(i+1)), biceps(turningPoints(i):turningPoints(i+1)), 'b-');

plot(EMGTime(turningPoints(i):turningPoints(i+1)), triceps(turningPoints(i):turningPoints(i+1)), 'r-');

title("EMGdata vs Time (Target 1)");

xlabel('Time (s)');

ylabel('Voltage');

legend('Biceps','Triceps')

hold off

else

continue

end

p = 1;

end

figure

hold on

yyaxis left

plot(TimeD1,angularvel, 'b-')

ylabel('Angular Velocity (rad/s)')

yyaxis right

plot(EMGTime,biceps, 'r-')

plot(EMGTime(1:length(triceps)),triceps, 'k-')

title('EMG Data & Angular Velocity vs Time')

ylabel('Voltage')

xlabel('Time (s)')

legend('Angular Velocity', 'Biceps Data', 'Triceps Data')

%%

close all

XTvalue = DT1(:,1);

YTvalue = DT1(:,2);

ZTvalue = DT1(:,3);

Cd\_x = diff(DT1(:,1));

Cd\_t = diff(TimeD1);

velC = Cd\_x./Cd\_t;

velC = velC';

velC = [velC(end),velC];

velC = velC';

figure

plot(TimeD1,velC)

[A, B1] = butter(4, 1/60, "low");

velC = filtfilt(A, B1, velC);

turningPoints = [];

turningPoints = [];

i = 1;

while i <= length(velC) - 1

% Check if current time is within desired range before evaluating velC

if TimeD1(i) > 0.1 && TimeD1(i) < 66

if abs((velC(i)-velC(i+10))) >= 0.002||abs((velC(i)-velC(i-10))) >= 0.002

if (velC(i) <= 0 && velC(i + 1) > 0)||(velC(i) >= 0 && velC(i + 1) < 0) % Condition for turning point

turningPoints = [turningPoints; i + 1]; % Collect turning point

i = i + 50; %Search for turning Points in set interval

continue;

end

end

end

i = i + 1; % Otherwise, proceed to the next index

end

% Extract every other value

everyOther = Xvalue(turningPoints(1:2:end));

everyOther2 = Yvalue(turningPoints(1:2:end));

everyOther3 = Zvalue(turningPoints(1:2:end));

% Calculate the average

% average = mean(everyOther);

% average2 = mean(everyOther2);

% Extract every other value starting from the second element

everyOtherOpposite = flip(Xvalue(turningPoints(2:2:end)));

everyOtherOpposite2 = flip(Yvalue(turningPoints(2:2:end)));

everyOtherOpposite3 = flip(Zvalue(turningPoints(2:2:end)));

% Calculate the average

% averageOpposite = mean(everyOtherOpposite);

% averageOpposite2 = mean(everyOtherOpposite2);

% TY = [average, average2];

% TZ = [averageOpposite, averageOpposite2];

% turningPointsmean = plot(TY,TZ,'Color','g');

timeForvelC = TimeD1(1:end); % Drop the first time value

% Plot Y-velocity over time

figure(3);

plot(timeForvelC, velC, 'LineWidth', 2);

xlabel('Time');

ylabel('Velocity in Y');

title('Y-Velocity Over Time');

hold on;

% Highlight turning points on the plot

figure(3);

scatter(timeForvelC(turningPoints), velC(turningPoints), 'r', 'filled');

turningPointsX = timeForvelC(turningPoints(1:end));

turningPointsX = diff(turningPointsX);

% labeling

legend('X-Velocity', 'Turning Points');

figure;

hold on;

i1 = 2;

addedLegend1 = false;

for i2 = 3:3:length(turningPoints)

h1 = plot3(DT1(turningPoints(i1):turningPoints(i2),1), ...

DT1(turningPoints(i1):turningPoints(i2),2), ...

DT1(turningPoints(i1):turningPoints(i2),3), 'Color','r');

if ~addedLegend1

set(h1, 'DisplayName', 'Target 1');

addedLegend1 = true;

else

set(h1, 'HandleVisibility', 'off');

end

i1 = i1+3;

end

legend show;

hold off

endpointtarget = turningPoints(3:3:end);

for i = 1:length(endpointtarget)

ept(i,1) = DT1(endpointtarget(i),1);

ept(i,2) = DT1(endpointtarget(i),2);

ept(i,3) = DT1(endpointtarget(i),3);

end

figure

hold on

scatter3(epf(1:end,1), epf(1:end,2), epf(1:end,3))

scatter3(ept(1:end,1), ept(1:end,2), ept(1:end,3))

title('Scatter Plot of Targets and Movement Reversals');

legend("Finger Endpoints", "Target Endpoints")

grid on

hold off

for i = 1:size(epf,1)

disx(i) = epf(i,1)-ept(i,1);

disy(i) = epf(i,2)-ept(i,2);

disz(i) = epf(i,3)-ept(i,3);

dis(i) = sqrt(disx(i)^2+disy(i)^2+disz(i)^2);

end

dismean = mean(dis);

disstd = std(dis);

fprintf('Mean distance is %d\n', dismean)

fprintf('Standard Deviation is %d\n', disstd)

p = 1;

for i = 1:size(epf,1)

epf1(p,1) = epf(i,1);

epf1(p,2) = epf(i,2);

epf1(p,3) = epf(i,3);

p = p+1;

end

p = 1;

for i = 1:size(ept,1)

ept1(p,1) = ept(i,1);

ept1(p,2) = ept(i,2);

ept1(p,3) = ept(i,3);

p = p+1;

end

%%

close all

% Define the target point

target = [mean(ept1(:,1)), mean(ept1(:,2)), mean(ept1(:,3))];

X = (epf1(:,1));

Y = epf1(:,2);

Z = epf1(:,3);

Xt = ept1(:,1);

Yt = ept1(:,2);

Zt = ept1(:,3);

% 1. Absolute Constant Error

distancesToTarget = sqrt((X - Xt).^2 + (Y - Yt).^2 + (Z - Zt).^2);

absoluteConstantError = mean(distancesToTarget);

X = (X-Xt)+target(1);

Y = (Y-Yt)+target(2);

Z = (Z-Zt)+target(3);

% 2. Constant Error

dataCenter = [mean(X), mean(Y), mean(Z)];

constantError = norm(dataCenter - target);

% 3. Variable Error

stdX = std(X);

stdY = std(Y);

stdZ = std(Z);

variableError = sqrt(stdX^2 + stdY^2 + stdZ^2);

% Perform PCA to get the eigenvectors and eigenvalues

[coeff, ~, ~] = pca([X, Y, Z]);

dataCenter = mean([X, Y, Z]);

% Plot data, sphere, and ellipsoid

figure;

hold on;

plot3(X, Y, Z, 'ro');

[sphereX, sphereY, sphereZ] = sphere;

radius = variableError;

sphereX = radius \* sphereX + dataCenter(1);

sphereY = radius \* sphereY + dataCenter(2);

sphereZ = radius \* sphereZ + dataCenter(3);

surf(sphereX, sphereY, sphereZ, 'FaceAlpha', 0.3, 'EdgeColor', 'none');

% Plot ellipsoid

scale = [stdX, stdY, stdZ]; % Scaling factors for the ellipsoid

[ellipsoidX, ellipsoidY, ellipsoidZ] = ellipsoid(0, 0, 0, scale(1), scale(2), scale(3), 50);

transformedEllipsoid = [ellipsoidX(:), ellipsoidY(:), ellipsoidZ(:)] \* coeff;

ellipsoidX = reshape(transformedEllipsoid(:,1), size(ellipsoidX)) + dataCenter(1);

ellipsoidY = reshape(transformedEllipsoid(:,2), size(ellipsoidY)) + dataCenter(2);

ellipsoidZ = reshape(transformedEllipsoid(:,3), size(ellipsoidZ)) + dataCenter(3);

surf(ellipsoidX, ellipsoidY, ellipsoidZ, 'FaceColor', 'green', 'FaceAlpha', 0.1);

% Plot eigenvectors as arrows starting from the center of the data

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,1)\*scale(1), coeff(2,1)\*scale(1), coeff(3,1)\*scale(1), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,2)\*scale(2), coeff(2,2)\*scale(2), coeff(3,2)\*scale(2), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,3)\*scale(3), coeff(2,3)\*scale(3), coeff(3,3)\*scale(3), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

xlabel('X');

ylabel('Y');

zlabel('Z');

title('Data Points, Error Sphere, Fitted Ellipsoid, and PCA Eigenvectors');

legend('Data Points', 'Error Sphere', 'Fitted Ellipsoid', 'PCA Vectors');

axis equal;

grid on;

hold off;

%% For D2

close all

Xvalue = D2(:,1);

Yvalue = D2(:,2);

Zvalue = D2(:,3);

Cd\_x = diff(D2(:,1));

Cd\_y = diff(D2(:,2));

Cd\_z = diff(D2(:,3));

Cd\_t = diff(TimeD2);

velC = Cd\_x./Cd\_t;

velCy = Cd\_y./Cd\_t;

velCz = Cd\_z./Cd\_t;

velC = velC';

velCy = velCy';

velCz = velCz';

velC = [velC(end),velC];

velCy = [velCy(end),velCy];

velCz = [velCz(end),velCz];

velC = velC';

velCy = velCy';

velCz = velCz';

figure

plot(TimeD2,velC)

[A, B1] = butter(2, 5/60, "low");

velC = filtfilt(A, B1, velC);

turningPoints = [];

i = 1;

while i <= length(velC) - 1

% Check if current time is within desired range before evaluating velC

if TimeD2(i) > 3.9 && TimeD2(i) < 90.54

if abs((velC(i)-velC(i+10))) >= 0.05||abs((velC(i)-velC(i-10))) >= 0.05

if (velC(i) <= 0.01 && velC(i + 1) > -0.01)||(velC(i) >= -0.01 && velC(i + 1) < 0.01) % Condition for turning point

turningPoints = [turningPoints; i + 1]; % Collect turning point

i = i + 50; %Search for turning Points in set interval

continue;

end

end

end

i = i + 1; % Otherwise, proceed to the next index

end

% Extract every other value

everyOther = Xvalue(turningPoints(1:2:end));

everyOther2 = Yvalue(turningPoints(1:2:end));

everyOther3 = Zvalue(turningPoints(1:2:end));

% Calculate the average

% average = mean(everyOther);

% average2 = mean(everyOther2);

% Extract every other value starting from the second element

everyOtherOpposite = flip(Xvalue(turningPoints(2:2:end)));

everyOtherOpposite2 = flip(Yvalue(turningPoints(2:2:end)));

everyOtherOpposite3 = flip(Zvalue(turningPoints(2:2:end)));

% Calculate the average

% averageOpposite = mean(everyOtherOpposite);

% averageOpposite2 = mean(everyOtherOpposite2);

% TY = [average, average2];

% TZ = [averageOpposite, averageOpposite2];

% turningPointsmean = plot(TY,TZ,'Color','g');

timeForvelC = TimeD2(1:end); % Drop the first time value

% Plot Y-velocity over time

figure(3);

plot(timeForvelC, velC, 'LineWidth', 2);

xlabel('Time');

ylabel('Velocity in Y');

title('Y-Velocity Over Time');

hold on;

% Highlight turning points on the plot

figure(3);

scatter(timeForvelC(turningPoints), velC(turningPoints), 'r', 'filled');

turningPointsX = timeForvelC(turningPoints(1:end));

turningPointsX = diff(turningPointsX);

% labeling

legend('X-Velocity', 'Turning Points');

figure;

hold on;

i1 = 1;

addedLegend1 = false;

for i2 = 2:3:length(turningPoints)

h1 = plot3(D2(turningPoints(i1):turningPoints(i2),1), ...

D2(turningPoints(i1):turningPoints(i2),2), ...

D2(turningPoints(i1):turningPoints(i2),3), 'Color','r');

if ~addedLegend1

set(h1, 'DisplayName', 'Target 1');

addedLegend1 = true;

else

set(h1, 'HandleVisibility', 'off');

end

i1 = i1+3;

end

legend show;

hold off

endpointfinger = turningPoints(2:3:end);

for i = 1:length(endpointfinger)

epf(i,1) = D2(endpointfinger(i),1);

epf(i,2) = D2(endpointfinger(i),2);

epf(i,3) = D2(endpointfinger(i),3);

end

%finds peaks

[Onepiece, locationx] = findpeaks(-velC, 'MinPeakHeight', 0.3, 'MinPeakDistance', 50);

figure;

hold on;

% Plot velocity components

plot(1:length(velC), velC, 'b', 'DisplayName', 'Velocity Component X');

% Plot peak locations

plot(locationx, velC(locationx), 'bo', 'MarkerSize', 8, 'DisplayName', 'Peaks X');

xlabel('Time');

ylabel('Absolute Velocity');

title('Absolute Velocity and Peaks in X, Y, and Z Components');

hold off;

for i = 1:length(locationx)

magnV(i) = sqrt(velC(i)^2+velCy(i)^2+velCz(i)^2);

end

meanPeakV = mean(magnV);

stdPeakV = std(magnV);

fprintf('Mean Peak Velocity is %d\n', meanPeakV)

fprintf('Standard Deviation is %d\n', stdPeakV)

%% Elbow Angle Vs Time and angular velocity

close all

dot = sum(V1D2 .\* V2D2, 2);

magV1 = sqrt(sum(V1D2.^2, 2));

magV2 = sqrt(sum(V2D2.^2, 2));

angrads = acos(dot./(magV1.\*magV2));

elbow\_angle = angrads\*(180/pi);

delbow\_ang = diff(angrads);

angularvel = delbow\_ang./Cd\_t;

angularvel = angularvel';

angularvel = [angularvel(end),angularvel];

angularvel = angularvel';

bicepdata = load("Data 2\SahmetBicep2DTarget1.mat");

tricepdata = load("Data 2\SahmetTicep2DTarget1.mat");

biceps = bicepdata.ch2data;

triceps = tricepdata.ch1data;

biceps = [biceps(end),biceps];

triceps = [triceps(end),triceps];

biceps = biceps(1:10:end);

triceps = triceps(1:10:end);

figure

plot(TimeD2, elbow\_angle, 'b-');

title("Elbow Joint Angle vs Time");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

EMGTime = linspace(1,80,8001);

p = 1;

for i = 1:3:length(turningPoints)-1

figure

subplot(2,2,p)

hold on

plot(TimeD2(turningPoints(i):turningPoints(i+1)), elbow\_angle(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Angle vs Time (Target 2)");

xlabel('Time (s)');

ylabel('Elbow Angle (degrees)');

hold off

p = p+1;

subplot(2,2,p)

hold on

plot(TimeD2(turningPoints(i):turningPoints(i+1)), angularvel(turningPoints(i):turningPoints(i+1)), 'b-');

title("Elbow Joint Anglular Velocity vs Time (Target 2)");

xlabel('Time (s)');

ylabel('Angular Velocity (rads/s)');

hold off

p = p+1;

if turningPoints(i+1) <= 8000

subplot(2,2,p)

hold on

plot(EMGTime(turningPoints(i):turningPoints(i+1)), biceps(turningPoints(i):turningPoints(i+1)), 'b-');

plot(EMGTime(turningPoints(i):turningPoints(i+1)), triceps(turningPoints(i):turningPoints(i+1)), 'r-');

title("EMGdata vs Time (Target 2)");

xlabel('Time (s)');

ylabel('Voltage');

legend('Biceps','Triceps')

hold off

else

p = 1;

continue

end

p = 1;

end

figure

hold on

yyaxis left

plot(TimeD2,angularvel, 'b-')

ylabel('Angular Velocity (rad/s)')

yyaxis right

plot(EMGTime,biceps, 'r-')

plot(EMGTime(1:length(triceps)),triceps, 'k-')

title('EMG Data & Angular Velocity vs Time')

ylabel('Voltage')

xlabel('Time (s)')

legend('Angular Velocity', 'Biceps Data', 'Triceps Data')

%%

close all

turningPoints = [];

ept = [];

ept1 = [];

XTvalue = DT2(:,1);

YTvalue = DT2(:,2);

ZTvalue = DT2(:,3);

Cd\_x = diff(DT2(:,1));

Cd\_t = diff(TimeD2);

velC = Cd\_x./Cd\_t;

velC = velC';

velC = [velC(end),velC];

velC = velC';

figure

plot(TimeD2,velC)

[A, B1] = butter(2, 0.8/60, "low");

velC = filtfilt(A, B1, velC);

turningPoints(1) = 1;

i = 2;

while i <= length(velC) - 1

% Check if current time is within desired range before evaluating velC

if TimeD2(i) > 0.33 && TimeD2(i) < 93.85

if abs((velC(i)-velC(i+10))) >= 0.0005||abs((velC(i)-velC(i-10))) >= 0.0005

if (velC(i) <= 0 && velC(i + 1) > 0)||(velC(i) >= 0 && velC(i + 1) < 0) % Condition for turning point

turningPoints = [turningPoints; i + 1]; % Collect turning point

i = i + 100; %Search for turning Points in set interval

continue;

end

end

end

i = i + 1; % Otherwise, proceed to the next index

end

% Extract every other value

everyOther = Xvalue(turningPoints(1:2:end));

everyOther2 = Yvalue(turningPoints(1:2:end));

everyOther3 = Zvalue(turningPoints(1:2:end));

% Calculate the average

% average = mean(everyOther);

% average2 = mean(everyOther2);

% Extract every other value starting from the second element

everyOtherOpposite = flip(Xvalue(turningPoints(2:2:end)));

everyOtherOpposite2 = flip(Yvalue(turningPoints(2:2:end)));

everyOtherOpposite3 = flip(Zvalue(turningPoints(2:2:end)));

% Calculate the average

% averageOpposite = mean(everyOtherOpposite);

% averageOpposite2 = mean(everyOtherOpposite2);

% TY = [average, average2];

% TZ = [averageOpposite, averageOpposite2];

% turningPointsmean = plot(TY,TZ,'Color','g');

timeForvelC = TimeD2(1:end); % Drop the first time value

% Plot Y-velocity over time

figure(3);

plot(timeForvelC, velC, 'LineWidth', 2);

xlabel('Time');

ylabel('Velocity in Y');

title('Y-Velocity Over Time');

hold on;

% Highlight turning points on the plot

figure(3);

scatter(timeForvelC(turningPoints), velC(turningPoints), 'r', 'filled');

turningPointsX = timeForvelC(turningPoints(1:end));

turningPointsX = diff(turningPointsX);

% labeling

legend('X-Velocity', 'Turning Points');

figure;

hold on;

i1 = 2;

addedLegend1 = false;

for i2 = 3:3:length(turningPoints)

h1 = plot3(DT2(turningPoints(i1):turningPoints(i2),1), ...

DT2(turningPoints(i1):turningPoints(i2),2), ...

DT2(turningPoints(i1):turningPoints(i2),3), 'Color','r');

if ~addedLegend1

set(h1, 'DisplayName', 'Target 1');

addedLegend1 = true;

else

set(h1, 'HandleVisibility', 'off');

end

i1 = i1+3;

end

legend show;

hold off

endpointtarget = turningPoints(3:3:end);

for i = 1:length(endpointtarget)

ept(i,1) = DT2(endpointtarget(i),1);

ept(i,2) = DT2(endpointtarget(i),2);

ept(i,3) = DT2(endpointtarget(i),3);

end

figure

hold on

scatter3(epf(1:end,1), epf(1:end,2), epf(1:end,3))

scatter3(ept(1:end,1), ept(1:end,2), ept(1:end,3))

title('Scatter Plot of Targets and Movement Reversals');

legend("Finger Endpoints", "Target Endpoints")

grid on

hold off

for i = 1:size(epf,1)

disx(i) = epf(i,1)-ept(i,1);

disy(i) = epf(i,2)-ept(i,2);

disz(i) = epf(i,3)-ept(i,3);

dis(i) = sqrt(disx(i)^2+disy(i)^2+disz(i)^2);

end

dismean = mean(dis);

disstd = std(dis);

fprintf('Mean distance is %d\n', dismean)

fprintf('Standard Deviation is %d\n', disstd)

p = 1;

for i = 1:size(epf,1)

epf1(p,1) = epf(i,1);

epf1(p,2) = epf(i,2);

epf1(p,3) = epf(i,3);

p = p+1;

end

p = 1;

for i = 1:size(ept,1)

ept1(p,1) = ept(i,1);

ept1(p,2) = ept(i,2);

ept1(p,3) = ept(i,3);

p = p+1;

end

%%

close all

% Define the target point

target = [mean(ept1(:,1)), mean(ept1(:,2)), mean(ept1(:,3))];

X = (epf1(:,1));

Y = epf1(:,2);

Z = epf1(:,3);

Xt = ept1(:,1);

Yt = ept1(:,2);

Zt = ept1(:,3);

% 1. Absolute Constant Error

distancesToTarget = sqrt((X - Xt).^2 + (Y - Yt).^2 + (Z - Zt).^2);

absoluteConstantError = mean(distancesToTarget);

X = (X-Xt)+target(1);

Y = (Y-Yt)+target(2);

Z = (Z-Zt)+target(3);

% 2. Constant Error

dataCenter = [mean(X), mean(Y), mean(Z)];

constantError = norm(dataCenter - target);

% 3. Variable Error

stdX = std(X);

stdY = std(Y);

stdZ = std(Z);

variableError = sqrt(stdX^2 + stdY^2 + stdZ^2);

% Perform PCA to get the eigenvectors and eigenvalues

[coeff, ~, ~] = pca([X, Y, Z]);

dataCenter = mean([X, Y, Z]);

% Plot data, sphere, and ellipsoid

figure;

hold on;

plot3(X, Y, Z, 'ro');

[sphereX, sphereY, sphereZ] = sphere;

radius = variableError;

sphereX = radius \* sphereX + dataCenter(1);

sphereY = radius \* sphereY + dataCenter(2);

sphereZ = radius \* sphereZ + dataCenter(3);

surf(sphereX, sphereY, sphereZ, 'FaceAlpha', 0.3, 'EdgeColor', 'none');

% Plot ellipsoid

scale = [stdX, stdY, stdZ]; % Scaling factors for the ellipsoid

[ellipsoidX, ellipsoidY, ellipsoidZ] = ellipsoid(0, 0, 0, scale(1), scale(2), scale(3), 50);

transformedEllipsoid = [ellipsoidX(:), ellipsoidY(:), ellipsoidZ(:)] \* coeff;

ellipsoidX = reshape(transformedEllipsoid(:,1), size(ellipsoidX)) + dataCenter(1);

ellipsoidY = reshape(transformedEllipsoid(:,2), size(ellipsoidY)) + dataCenter(2);

ellipsoidZ = reshape(transformedEllipsoid(:,3), size(ellipsoidZ)) + dataCenter(3);

surf(ellipsoidX, ellipsoidY, ellipsoidZ, 'FaceColor', 'green', 'FaceAlpha', 0.1);

% Plot eigenvectors as arrows starting from the center of the data

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,1)\*scale(1), coeff(2,1)\*scale(1), coeff(3,1)\*scale(1), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,2)\*scale(2), coeff(2,2)\*scale(2), coeff(3,2)\*scale(2), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

quiver3(dataCenter(1), dataCenter(2), dataCenter(3), coeff(1,3)\*scale(3), coeff(2,3)\*scale(3), coeff(3,3)\*scale(3), 'k', 'LineWidth', 2, 'MaxHeadSize', 2);

xlabel('X');

ylabel('Y');

zlabel('Z');

title('Data Points, Error Sphere, Fitted Ellipsoid, and PCA Eigenvectors');

legend('Data Points', 'Error Sphere', 'Fitted Ellipsoid', 'PCA Vectors');

axis equal;

grid on;

hold off;