Matlab Code:

%% For "Moderate" Condition

clc;

clear variables;

close all;

data1 = readtable("mid.csv");

data1(1:7,:)=[];

data1array = table2array(data1);

Frame = data1(:,1);

Time=data1array(:,2);

X1=data1array(:,3);

Y1=data1array(:,4);

Z1=data1array(:,5);

X2=data1array(:,6);

Y2=data1array(:,7);

Z2=data1array(:,8);

X3=data1array(:,9);

Y3=data1array(:,10);

Z3=data1array(:,11);

plot3(X1,Y1,Z1);hold on

plot3(X2,Y2,Z2);

plot3(X3,Y3,Z3);

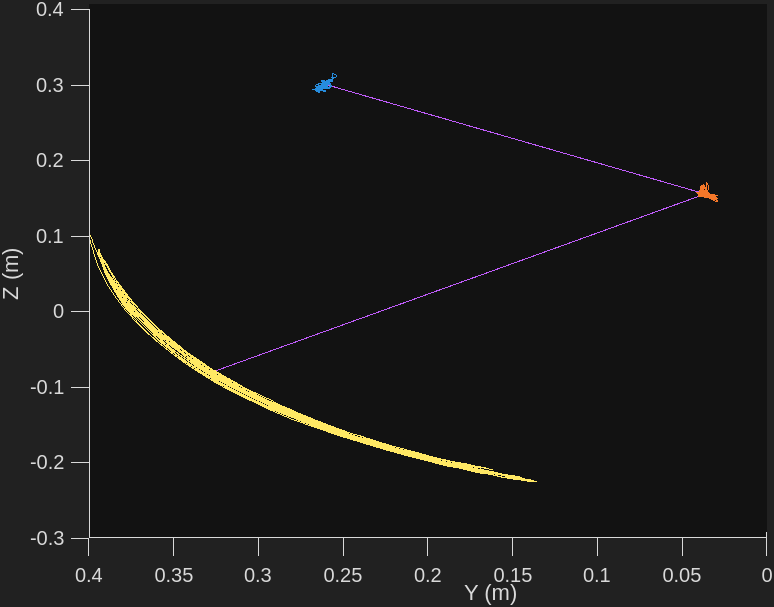
xlabel('X (m)');ylabel('Y (m)');zlabel('Z (m)');

line([X1(1),X2(1),X3(1)],[Y1(1),Y2(1),Y3(1)],[Z1(1),Z2(1),Z3(1)]);

hold off

% Jonathan's weight: 128 lb (58.0598 kg)

% Jonathan's forearm: 0.9289568 kgs



figure;

fframe=table2array(Frame);

numPoints = length(fframe);

angles = zeros(numPoints, 1);

%% angle of elbow over time

r = zeros(size(numPoints));

for i = 1:numPoints

stoe = [X3(i) - X2(i), Y3(i) - Y2(i), Z3(i) - Z2(i)];

etoh = [X2(i) - X1(i), Y2(i) - Y1(i), Z2(i) - Z1(i)];

he = norm(etoh);

r(i) = he;

dotProduct = dot(stoe, etoh) / (norm(stoe) \* norm(etoh));

dotProduct = max(min(dotProduct, 1), -1);

angles(i) = rad2deg(acos(dotProduct));

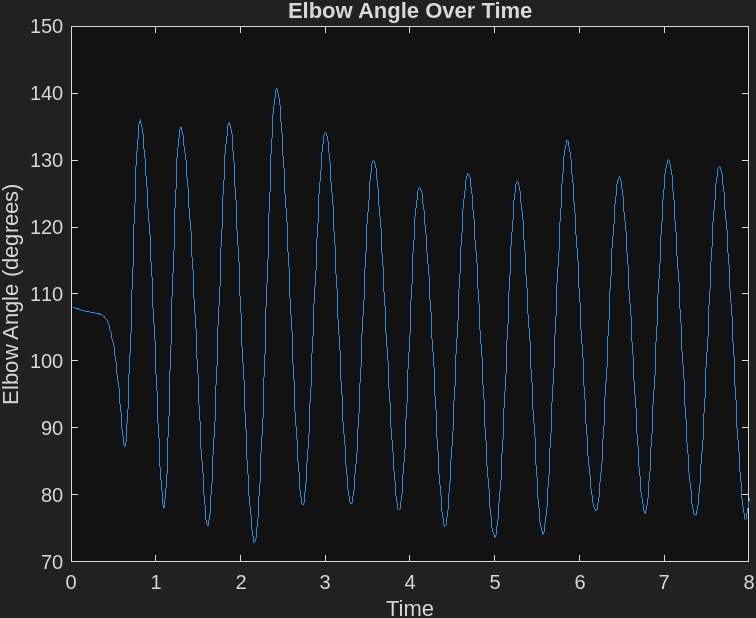
end

plot(Time, angles);

xlabel('Time');

ylabel('Elbow Angle (degrees)');

title('Elbow Angle Over Time');



%% angular velocity of elbow

figure;

Evel=diff(angles);

Evel=[Evel;Evel(end)];

plot(Time,Evel);

xlabel('Time');

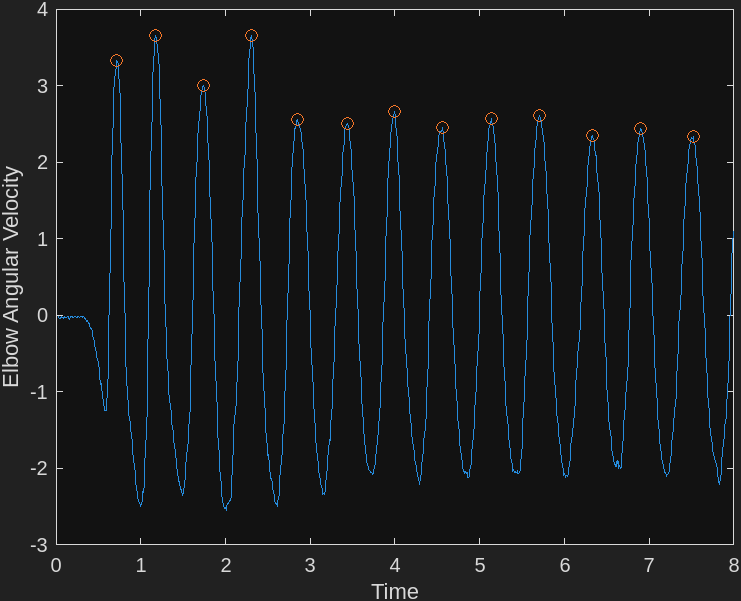
ylabel('Elbow Angular Velocity');

hold on

% finding peaks

[peakX, locX] = findpeaks(Evel, Time,'MinPeakDistance',.2,'MinPeakHeight',1);

scatter(locX,peakX);



%% Angular Acceleration over time

figure;

Eaccel=diff(Evel);

Eaccel=[Eaccel;Eaccel(end)];

plot(Time,Eaccel);

xlabel('Time');

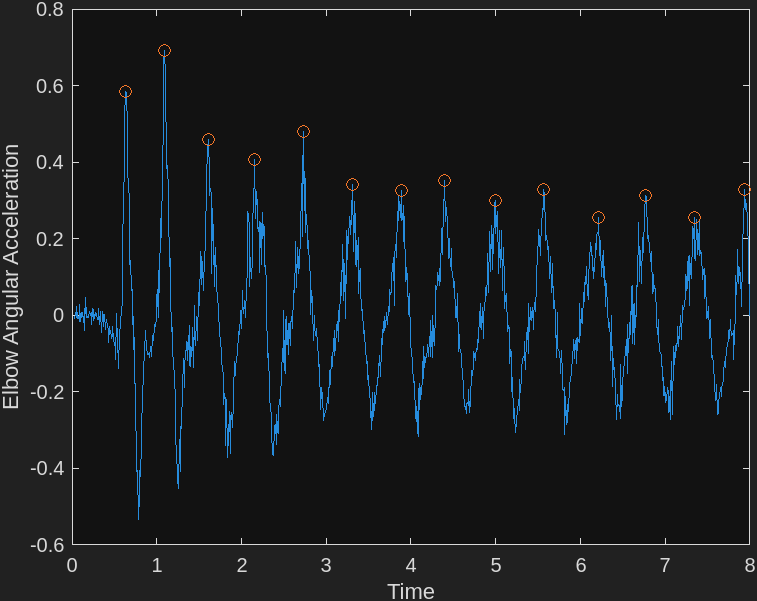
ylabel('Elbow Angular Acceleration');

hold on

% finding peaks

[peakX, locX] = findpeaks(Eaccel, Time,'MinPeakDistance',.4,'MinPeakHeight',0.2);

scatter(locX,peakX);



%%

s = size(angles);

displacement = zeros;

i = 2;

displacement(1) = 0;

for i = 2:s(1)

dist = angles(i) - angles(i-1);

displacement(i) = displacement(i-1) + abs(dist);

end

%% angular velocity vs angular displacement of elbow

figure

plot(angles, Evel)

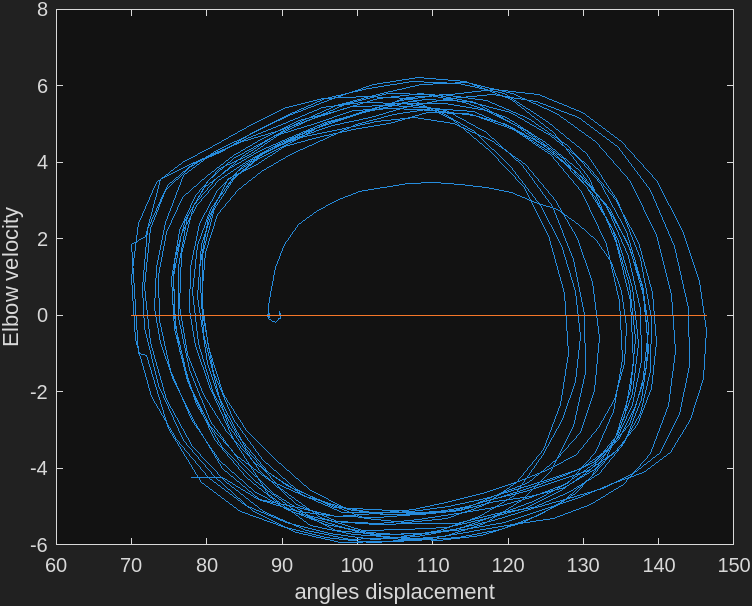
xlabel('angles displacement');

ylabel('Elbow velocity');

hold on

y = 0.\*angles;

plot(angles,y)



%% Finding inertia

m = 58.0598\*0.022;

L = 0.28;

RoG = r.\*0.827;

inertia = m.\*(RoG.^2);

%% Moderate Torque T=Iα

midTorque = inertia.\*Eaccel';

midrangeTorque = max(midTorque)-min(midTorque);

midmeanTorque = mean(midTorque);

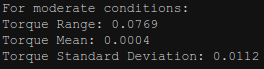
midstdTorque = std(midTorque);

disp('For moderate conditions:')

fprintf('Torque Range: %.4f\n', midrangeTorque);

fprintf('Torque Mean: %.4f\n', midmeanTorque);

fprintf('Torque Standard Deviation: %.4f\n', midstdTorque);



%% Moderate Torque Peaks

midPeakTorque = findpeaks(midTorque);

midrangePeakTorque = max(midPeakTorque)-min(midPeakTorque);

midmeanPeakTorque = mean(midPeakTorque);

midstdPeakTorque = std(midPeakTorque);

fprintf('Peak Torques Range: %.4f\n', midrangePeakTorque);

fprintf('Peak Torques Mean: %.4f\n', midmeanPeakTorque);

fprintf('Peak Torques Standard Deviation: %.4f\n', midstdPeakTorque);



%% Same process but under "As Fast as Possible" Conditions

clear variables

data1 = readtable("nyoooooooooom.csv");

data1(1:7,:)=[];

data1array = table2array(data1);

Frame = data1(:,1);

Time=data1array(:,2);

X1=data1array(:,3);

Y1=data1array(:,4);

Z1=data1array(:,5);

X2=data1array(:,6);

Y2=data1array(:,7);

Z2=data1array(:,8);

X3=data1array(:,9);

Y3=data1array(:,10);

Z3=data1array(:,11);

figure

plot3(X1,Y1,Z1);hold on

plot3(X2,Y2,Z2);

plot3(X3,Y3,Z3);

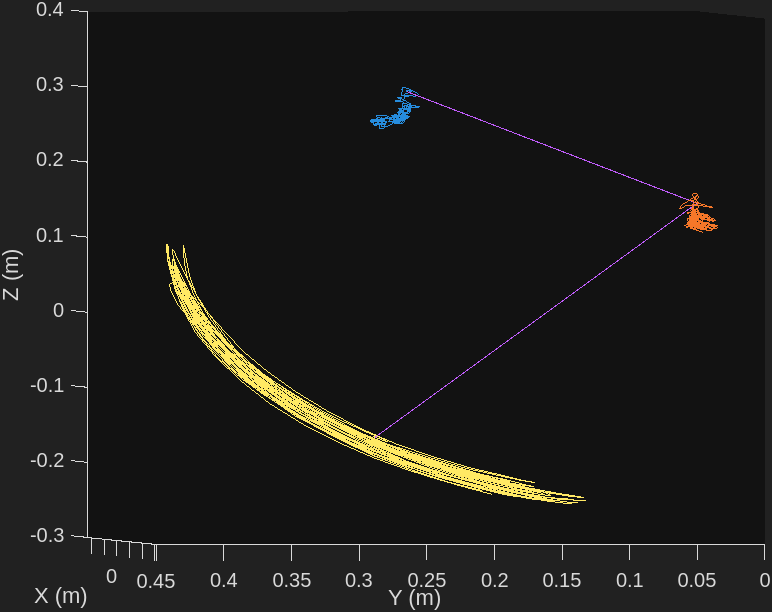
xlabel('X (m)');ylabel('Y (m)');zlabel('Z (m)');

line([X1(1),X2(1),X3(1)],[Y1(1),Y2(1),Y3(1)],[Z1(1),Z2(1),Z3(1)]);

hold off

% Jonathan's weight: 128 lb (58.0598 kg)

% Jonathan's forearm: 0.9289568 kgs



figure;

fframe=table2array(Frame);

numPoints = length(fframe);

angles = zeros(numPoints, 1);

%% angle of elbow over time

r = zeros(size(numPoints));

for i = 1:numPoints

stoe = [X3(i) - X2(i), Y3(i) - Y2(i), Z3(i) - Z2(i)];

etoh = [X2(i) - X1(i), Y2(i) - Y1(i), Z2(i) - Z1(i)];

he = norm(etoh);

r(i) = he;

dotProduct = dot(stoe, etoh) / (norm(stoe) \* norm(etoh));

dotProduct = max(min(dotProduct, 1), -1);

angles(i) = rad2deg(acos(dotProduct));

end

angles(268) = 71; %fixing the sensor bugging out?

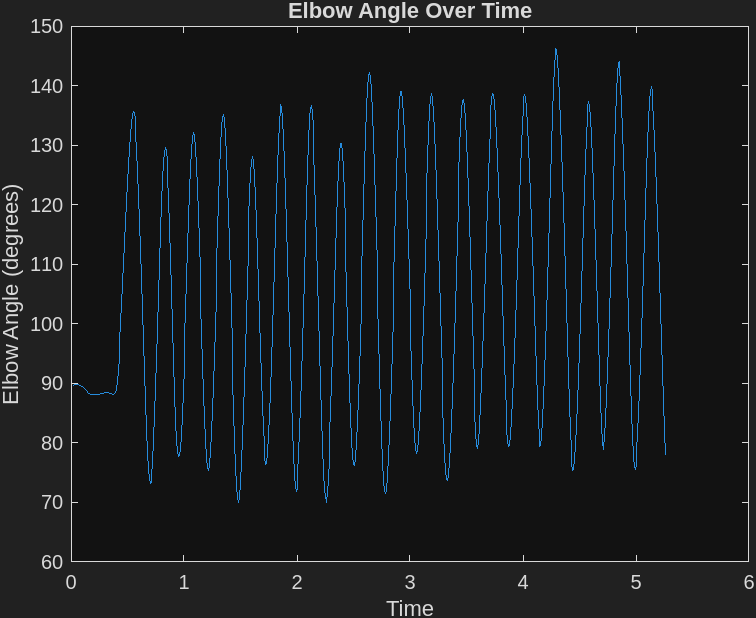
angles(269) = 70; %fixing the sensor bugging out?

plot(Time, angles);

xlabel('Time');

ylabel('Elbow Angle (degrees)');

title('Elbow Angle Over Time');



%% angular velocity of elbow

figure;

Evel=diff(angles);

Evel=[Evel;Evel(end)];

plot(Time,Evel);

xlabel('Time');

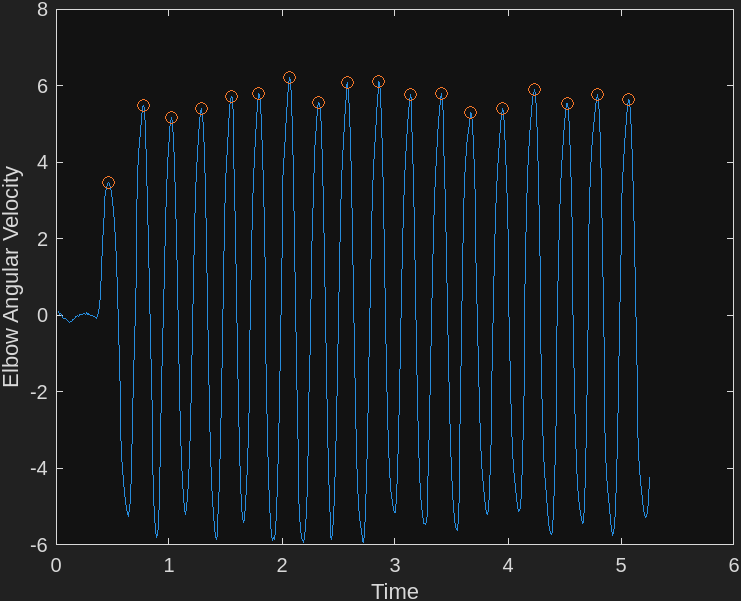
ylabel('Elbow Angular Velocity');

hold on

% finding peaks

[peakX, locX] = findpeaks(Evel, Time,'MinPeakDistance',.2,'MinPeakHeight',1);

scatter(locX,peakX);



%% Angular Acceleration over time

figure;

Eaccel=diff(Evel);

Eaccel=[Eaccel;Eaccel(end)];

plot(Time,Eaccel);

xlabel('Time');

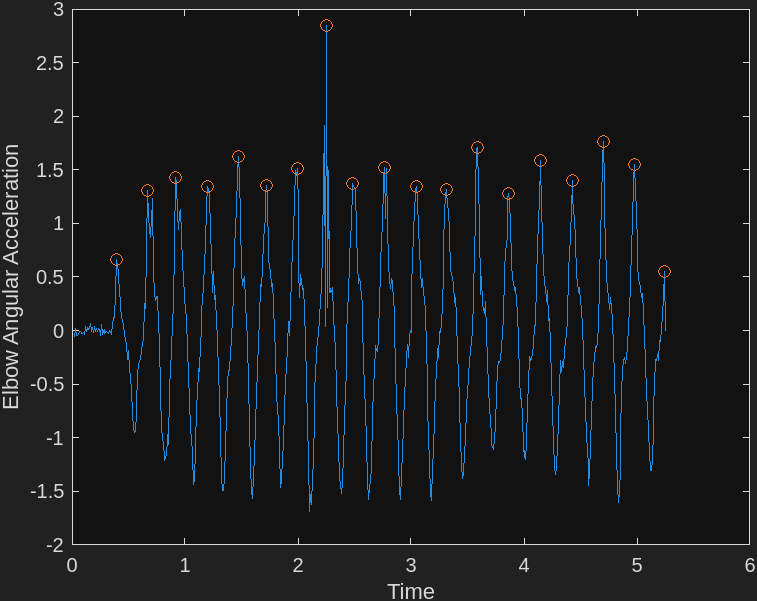
ylabel('Elbow Angular Acceleration');

hold on

% finding peaks

[peakX, locX] = findpeaks(Eaccel, Time,'MinPeakDistance',.2,'MinPeakHeight',0.2);

scatter(locX,peakX);



%%

s = size(angles);

displacement = zeros;

displacement(1) = 0;

for i = 2:s(1)

dist = angles(i) - angles(i-1);

displacement(i) = displacement(i-1) + abs(dist);

end

%% angular velocity vs angular displacement of elbow

figure

plot(angles, Evel)

xlabel('angles displacement');

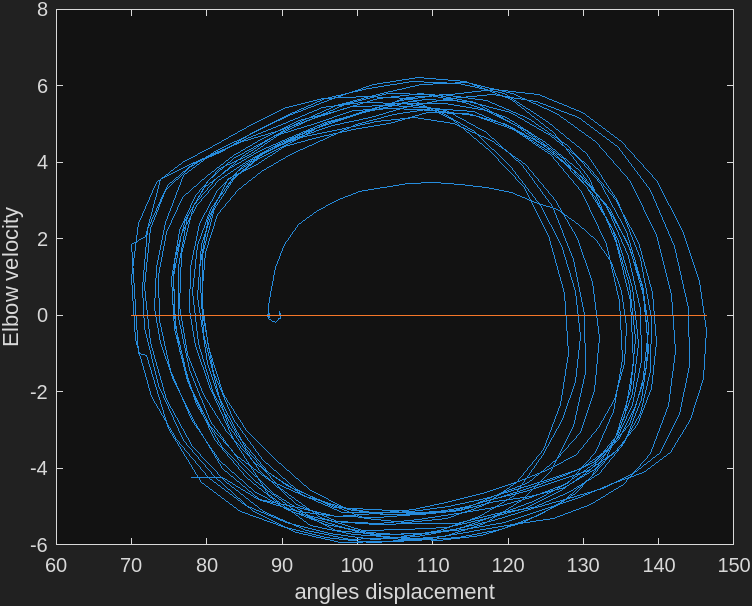
ylabel('Elbow velocity');

hold on

y = 0.\*angles;

plot(angles,y)

%finding peaks



%% Finding inertia

m = 58.0598\*0.022;

L = 0.28;

RoG = r.\*0.827;

inertia = m.\*(RoG.^2);

%% "As Fast as Possible" Torque T=Iα

fastTorque = inertia.\*Eaccel';

fastrangeTorque = max(fastTorque)-min(fastTorque);

fastmeanTorque = mean(fastTorque);

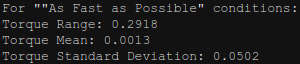
faststdTorque = std(fastTorque);

disp('For ""As Fast as Possible" conditions:')

fprintf('Torque Range: %.4f\n', fastrangeTorque);

fprintf('Torque Mean: %.4f\n', fastmeanTorque);

fprintf('Torque Standard Deviation: %.4f\n', faststdTorque);



%% "As Fast as Possible" Torque Peaks

fastPeakTorque = findpeaks(fastTorque);

fastrangePeakTorque = max(fastPeakTorque)-min(fastPeakTorque);

fastmeanPeakTorque = mean(fastPeakTorque);

faststdPeakTorque = std(fastPeakTorque);

fprintf('Peak Torques Range: %.4f\n', fastrangePeakTorque);

fprintf('Peak Torques Mean: %.4f\n', fastmeanPeakTorque);

fprintf('Peak Torques Standard Deviation: %.4f\n', faststdPeakTorque);



This lab involved getting used to using Optitrack and the kind of data analysis required by measuring the flexion and extension of the forearm by bending the elbow joint. Optitrack output the values of the x, y, and z positions for each of the motion trackers on the hand, elbow, and shoulder of the test subject to a CSV file. From there it was possible to load the CSV into matlab and analyze the data. Specifically, we were able to determine the angular displacement, angular velocity, angular acceleration, of the elbow. After externally measuring the radius of the elbow, and determining the mass of the forearm using a table, it was also possible to calculate the inertia and torque. The biggest difficulty in the lab involved plotting the peaks in the elbow’s angular velocity. This was because for our data, there was a segment where the sensors seemingly bugged out and registered an angle of 0 degrees. This meant that the displacement array didn’t always increase, which lead to finding the peaks not working initially. After locating the offending values and manually setting them to values near the expected values, the peaks were successfully located. In terms of technical difficulties, it was difficult to get the initial calibration of the motion tracking camera. One issue we encountered was the cameras appearing below the x-y plane. After several minutes of troubleshooting, it was discovered that flipping over the calibration balls would flip the orientation of the cameras.