MEAM 520

More Robot Dynamics

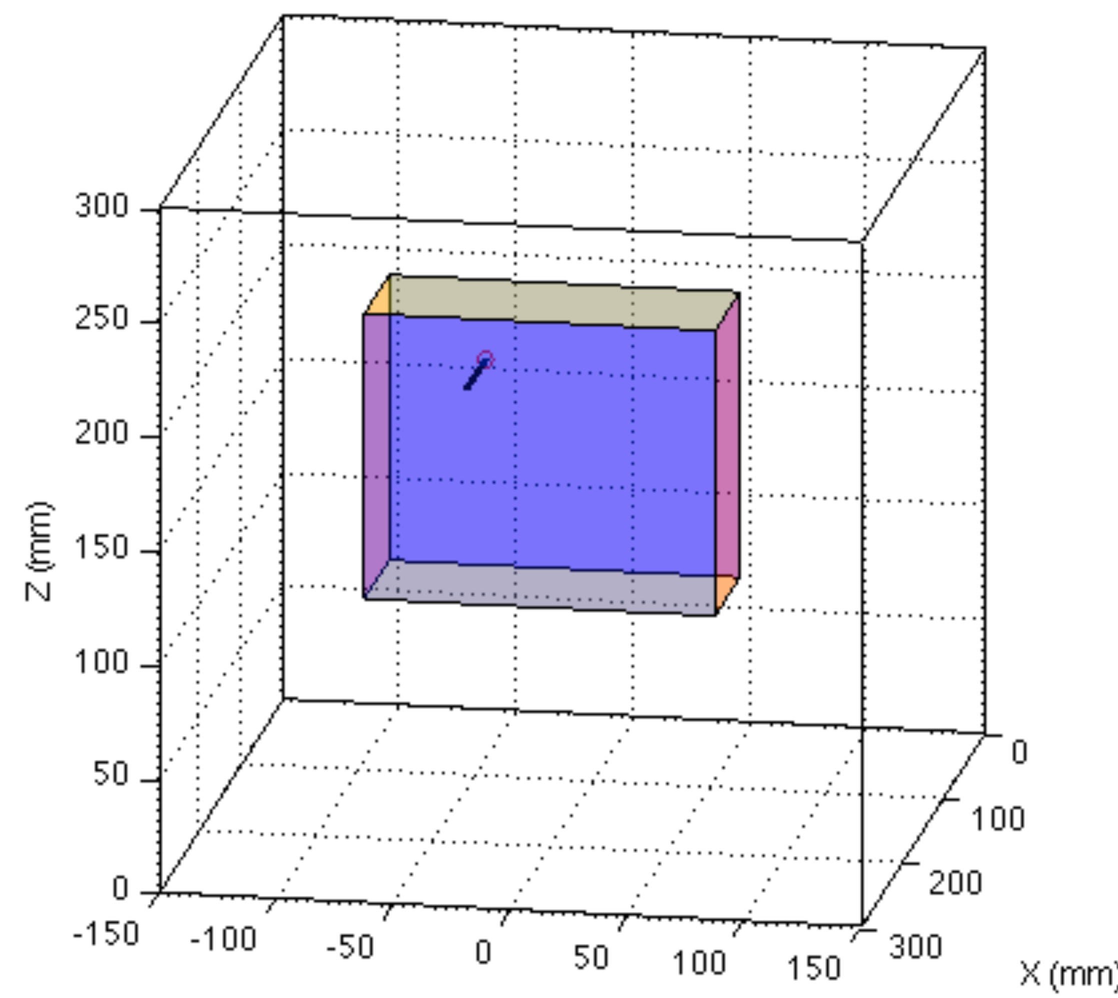
Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

Lecture 21: December 4, 2012

MEAM.Design : MEAM520-12C-P02-Rendering

Now that you have your team, it's time to get to work on project 2. This assignment is due by **5:00 p.m. on Tuesday, December 4**.

Start by downloading the starter code (v1). We are providing you with p-coded versions of all the functions described in the Phantom Guide. For example, calling `phantomStart(false);` starts the simulated phantom so you can work on your code on any computer. Instead of getting encoder readings from the real PHANToM, the system simulates the presence of a human user by reading a pre-recorded trajectory from the included `encsHistory.mat` file.

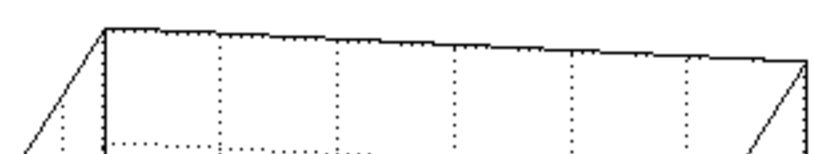


Demo: Haptic Box

Run `haptic_box_demo.m` and look at how it is written. This demonstration creates a virtual haptic box for the user to feel, as seen in the top illustration at right. The user is trapped inside the virtual box and feels a virtual spring force each time they contact a wall. The position of the PHANToM tip is shown as a red circle, the box is shown in transparent colors, and a scaled version of the force vector is shown as a thick black line.

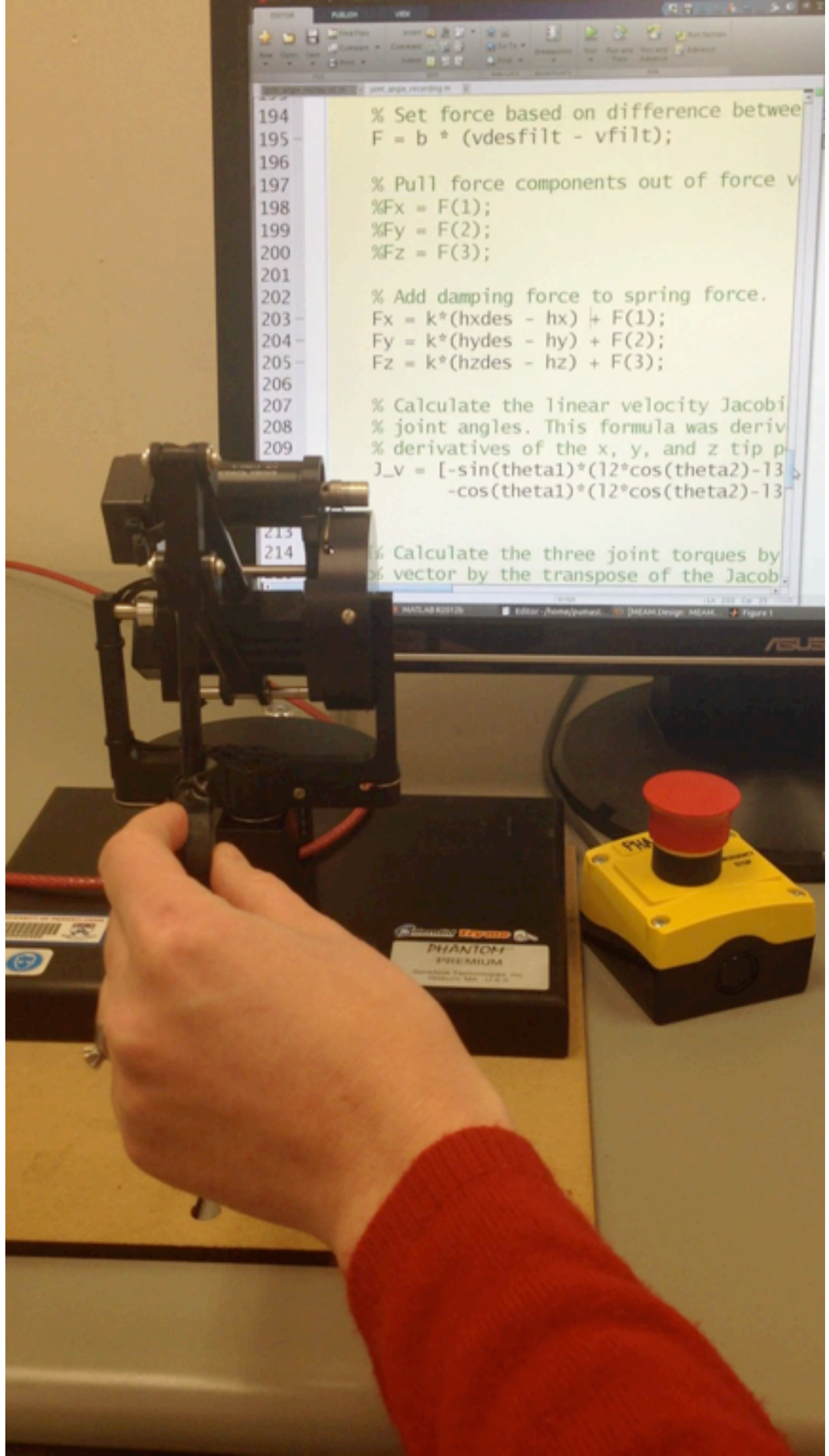
The system simulates the presence of a human user by default because you probably don't have a PHANToM connected to your computer. Look at how the forces Fx, Fy, and Fz are calculated from the positions hx, hy, and hz. This is the type of mapping you will need to create in this assignment.

Once you understand how the haptic box demo works, your team's task on this project is to complete the following two haptic rendering scenes created for the PHANToM.

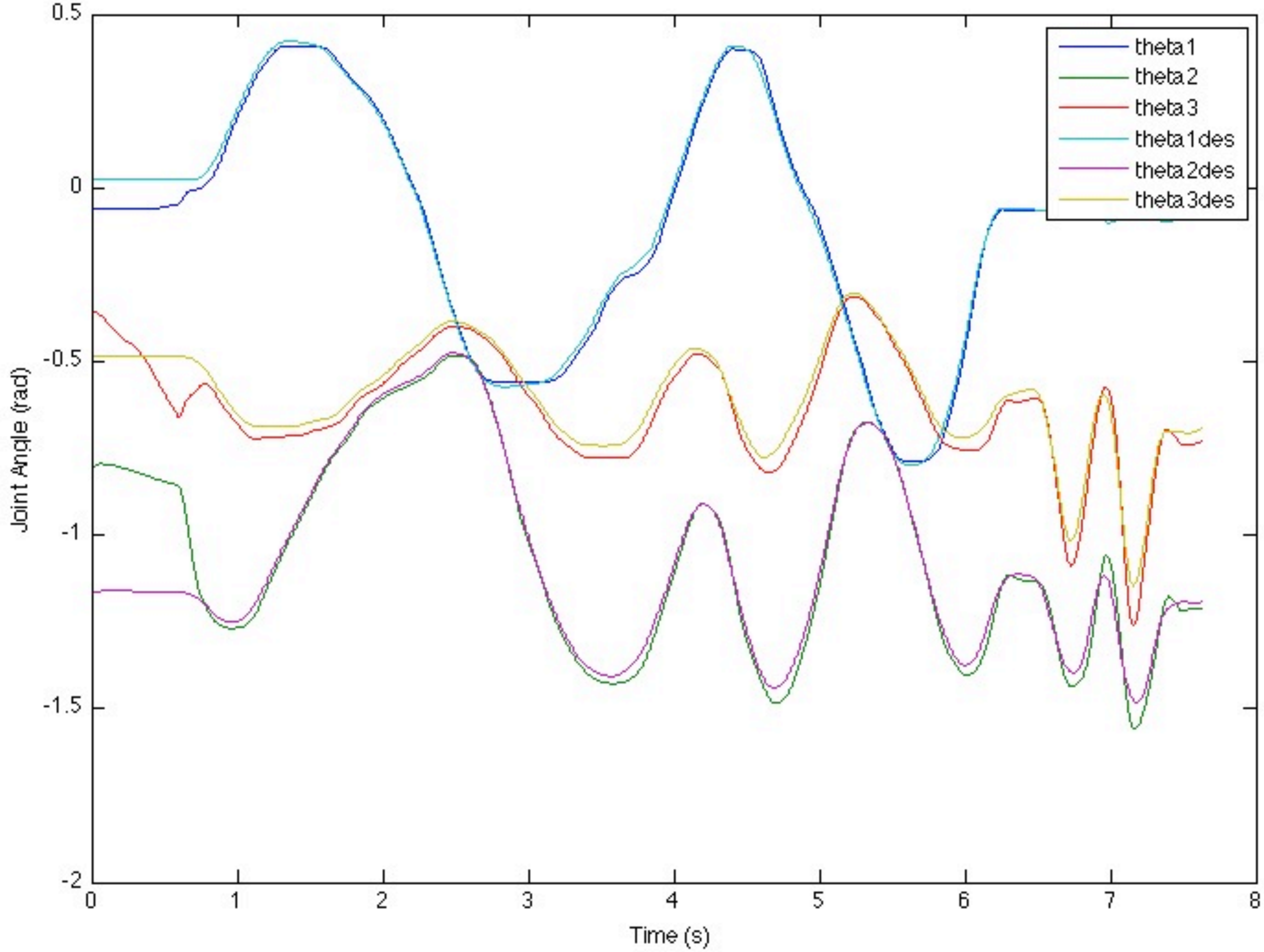


Task 1: Haptic Ball

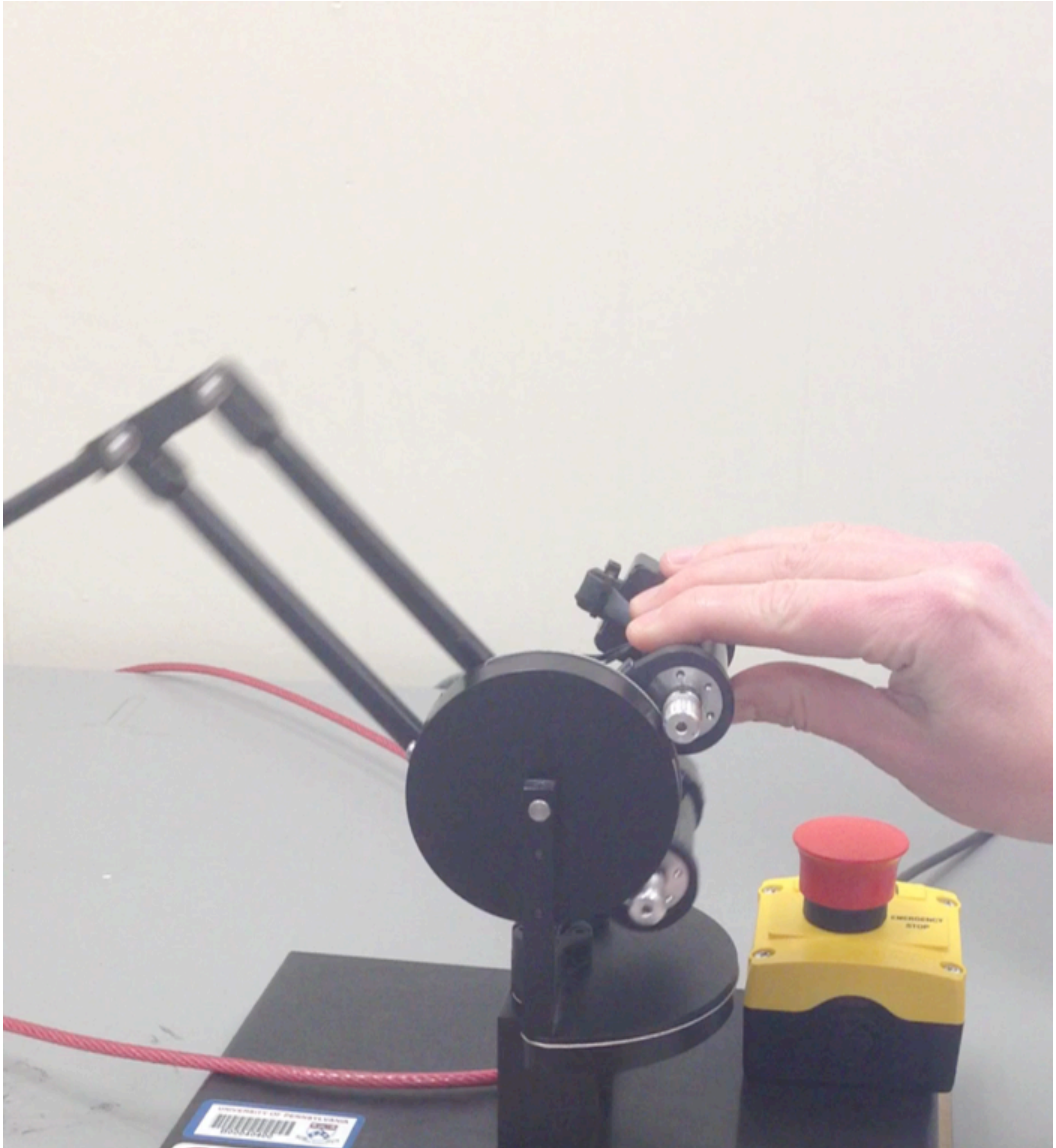
Complete the **haptic ball** scene that has been started for you in `haptic_ball_team50.m`. Change the filename to match replay loops final



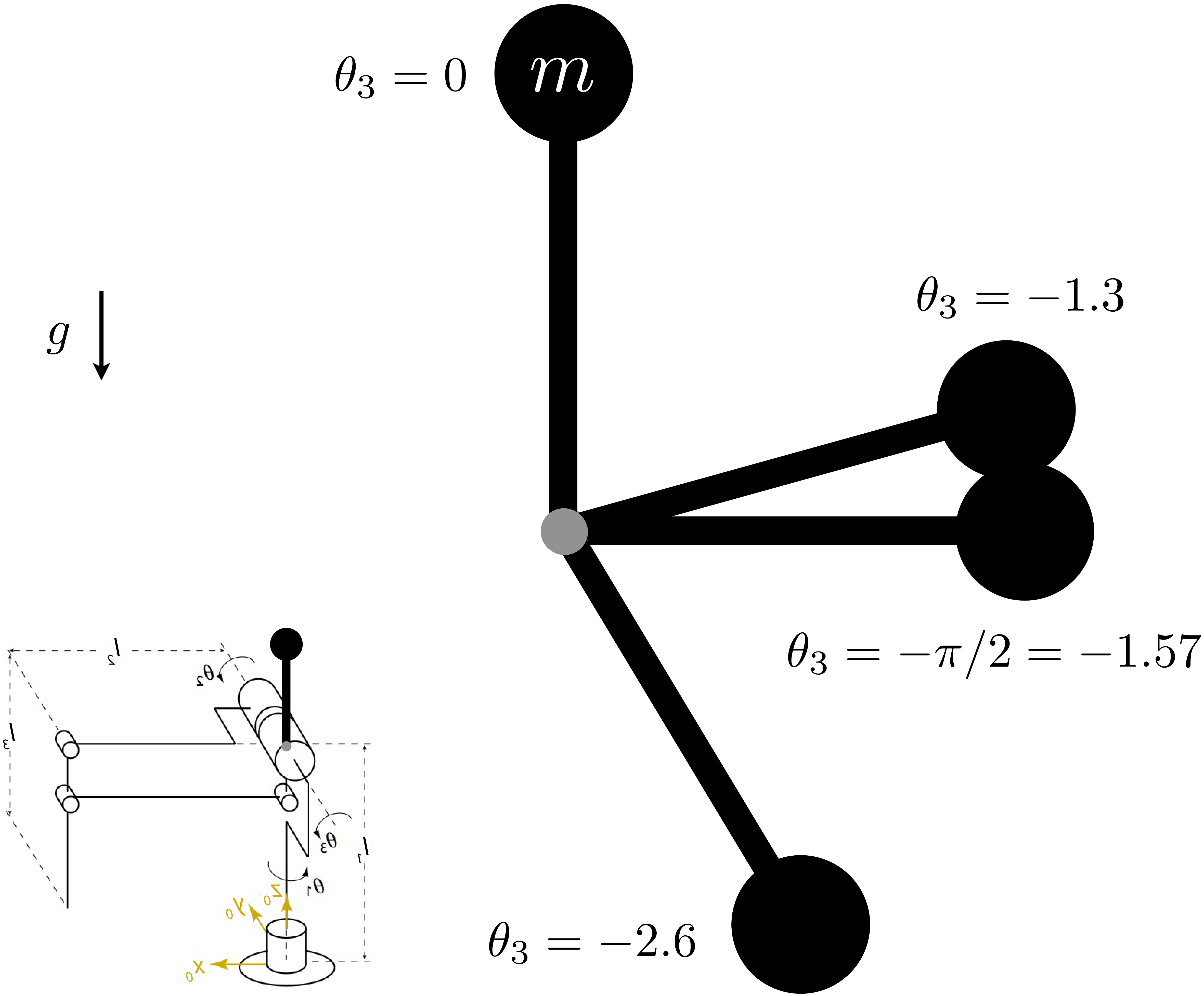
Very nice, but not perfect!



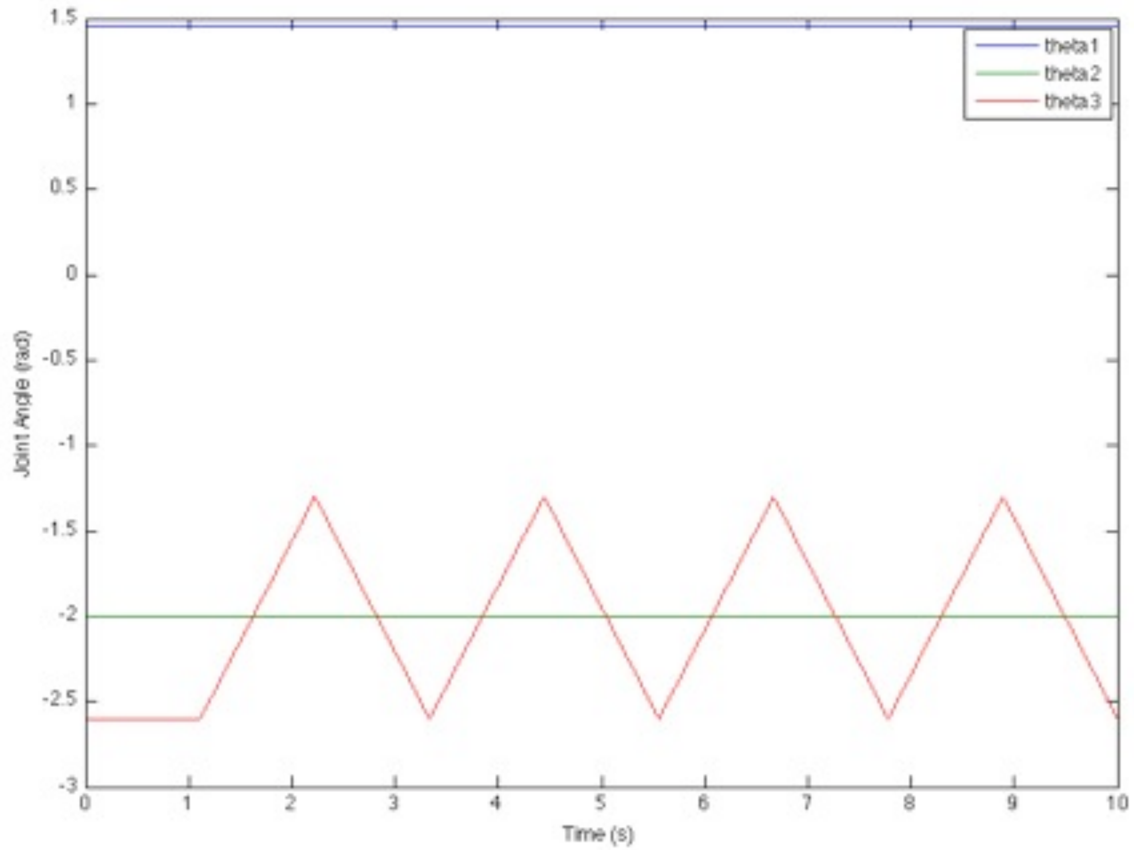
Add gravity compensation!

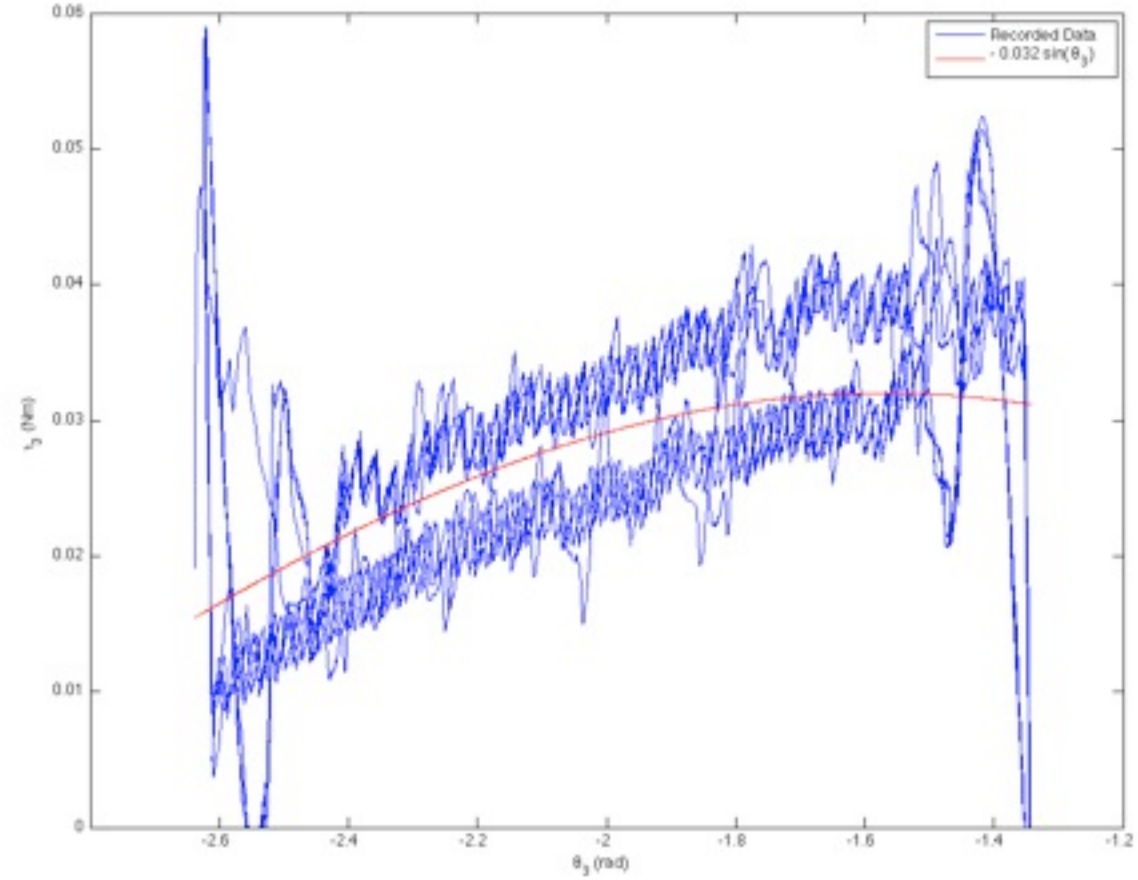


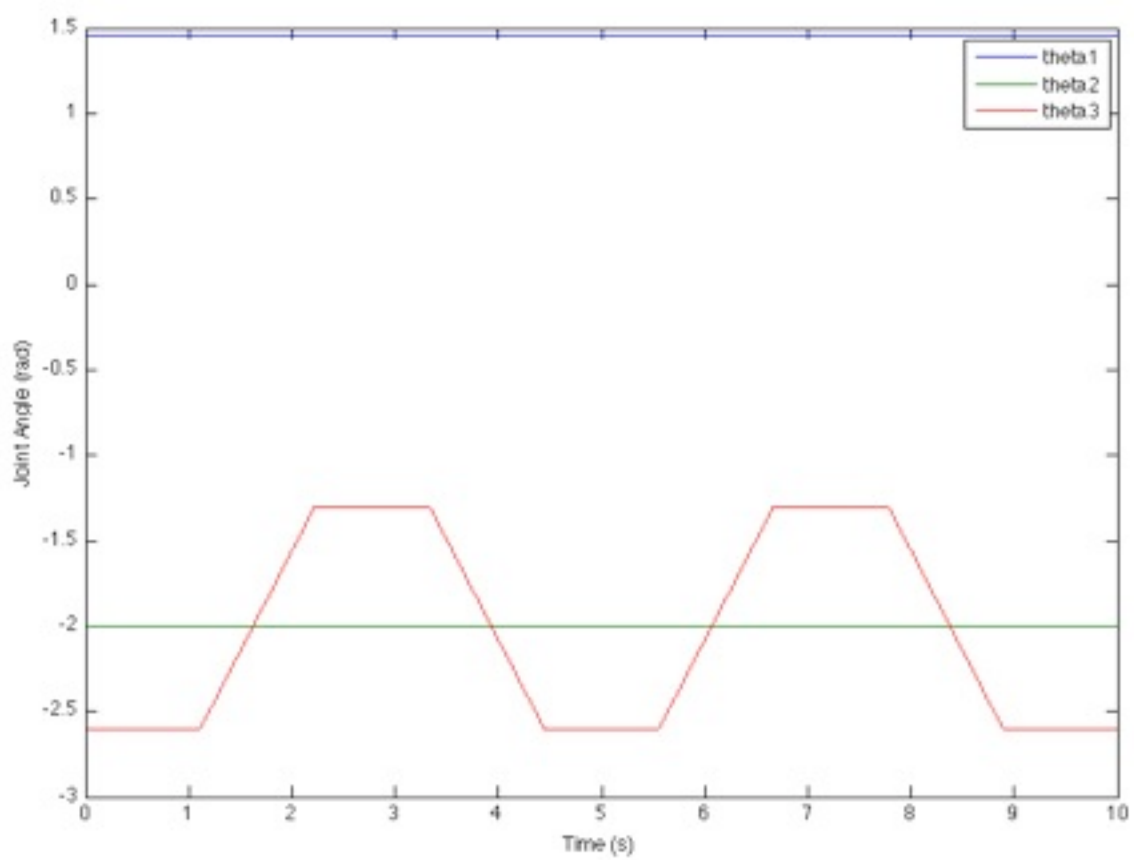
Move the robot slowly through a trajectory and record the torque needed to hold up the weight of the robot.

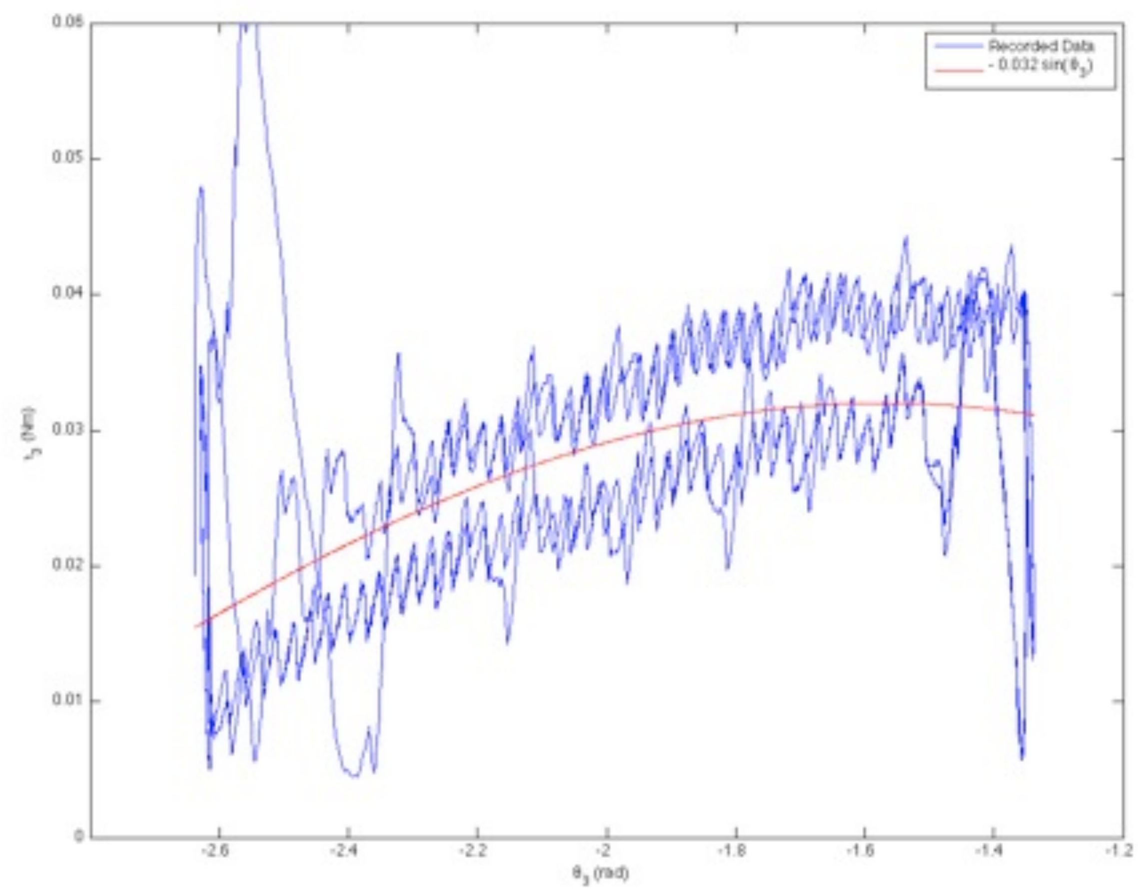


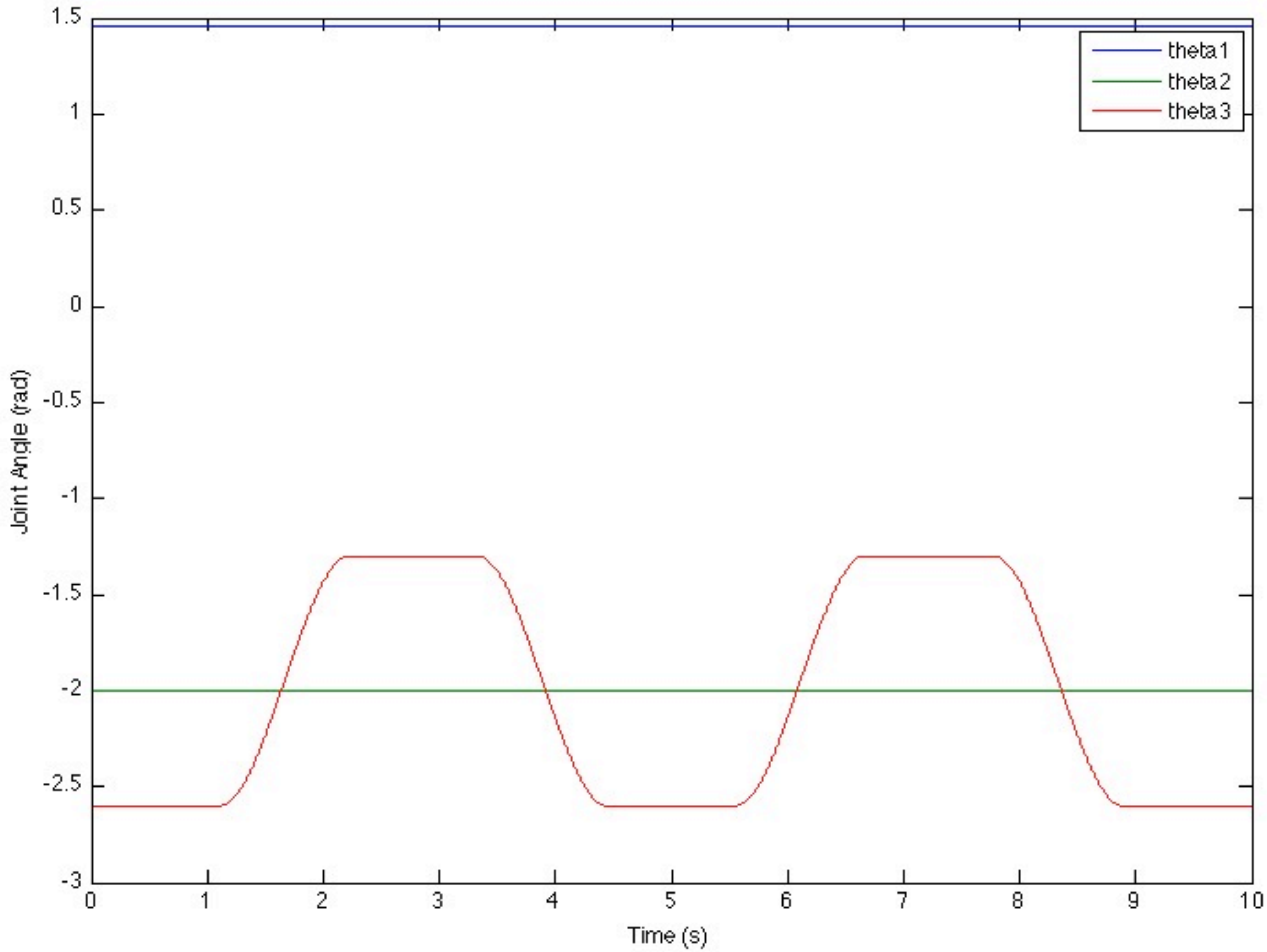
$\theta_3 = 0$
m
g
$\theta_3 = -1.3$
$\theta_3 = -\pi/2 = -1.57$
$\theta_3 = -2.6$


theta1
theta2
theta3
Joint Angle (rad)
Time (s)


Recorded Data
- 0.032 sin(θ_3)
τ_3 (Nm)
θ_3 (rad)

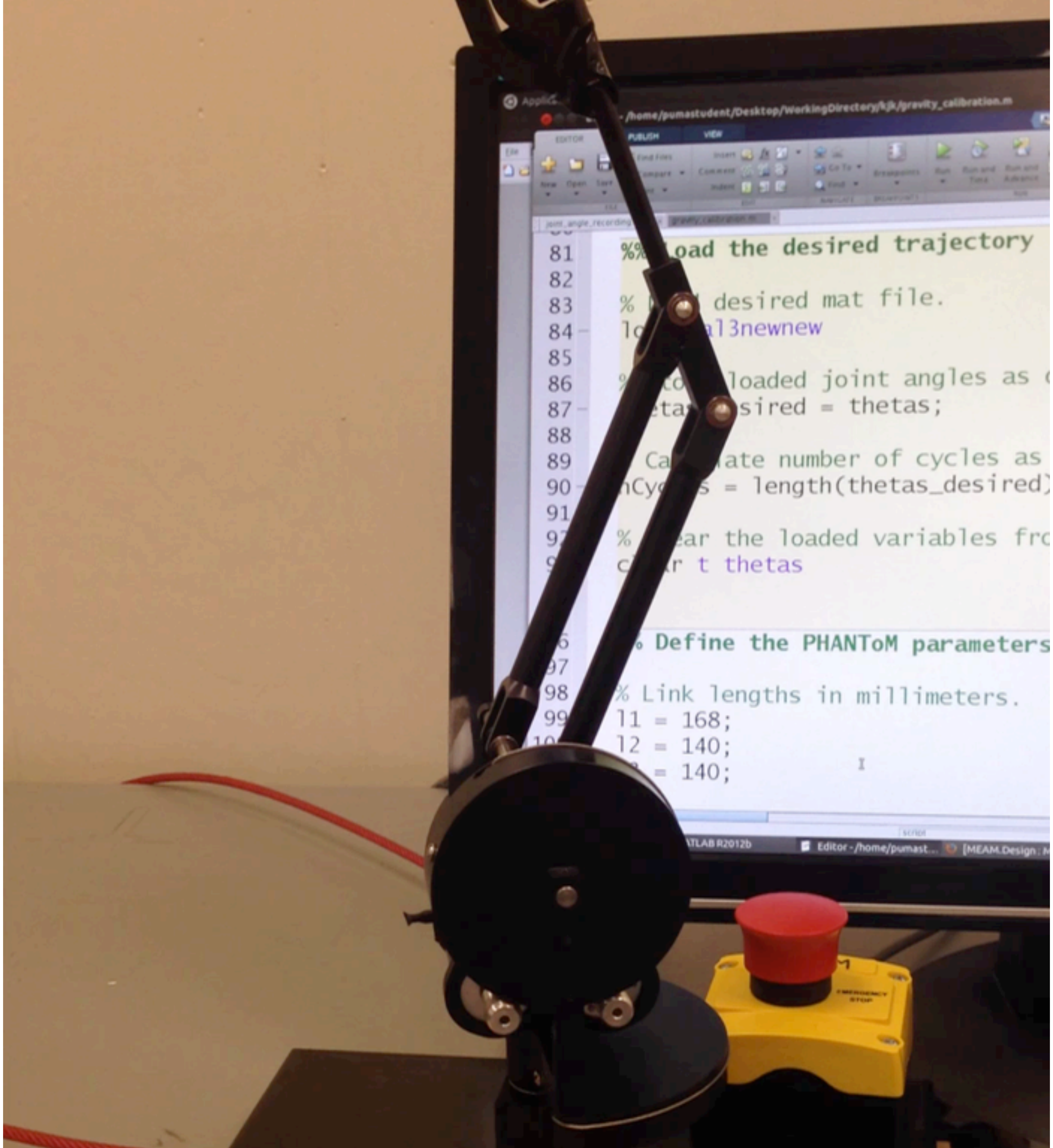

theta1
theta2
theta3
Joint Angle (rad)
Time (s)


Recorded Data
- 0.032 sin(θ_3)
τ_3 (Nm)
θ_3 (rad)

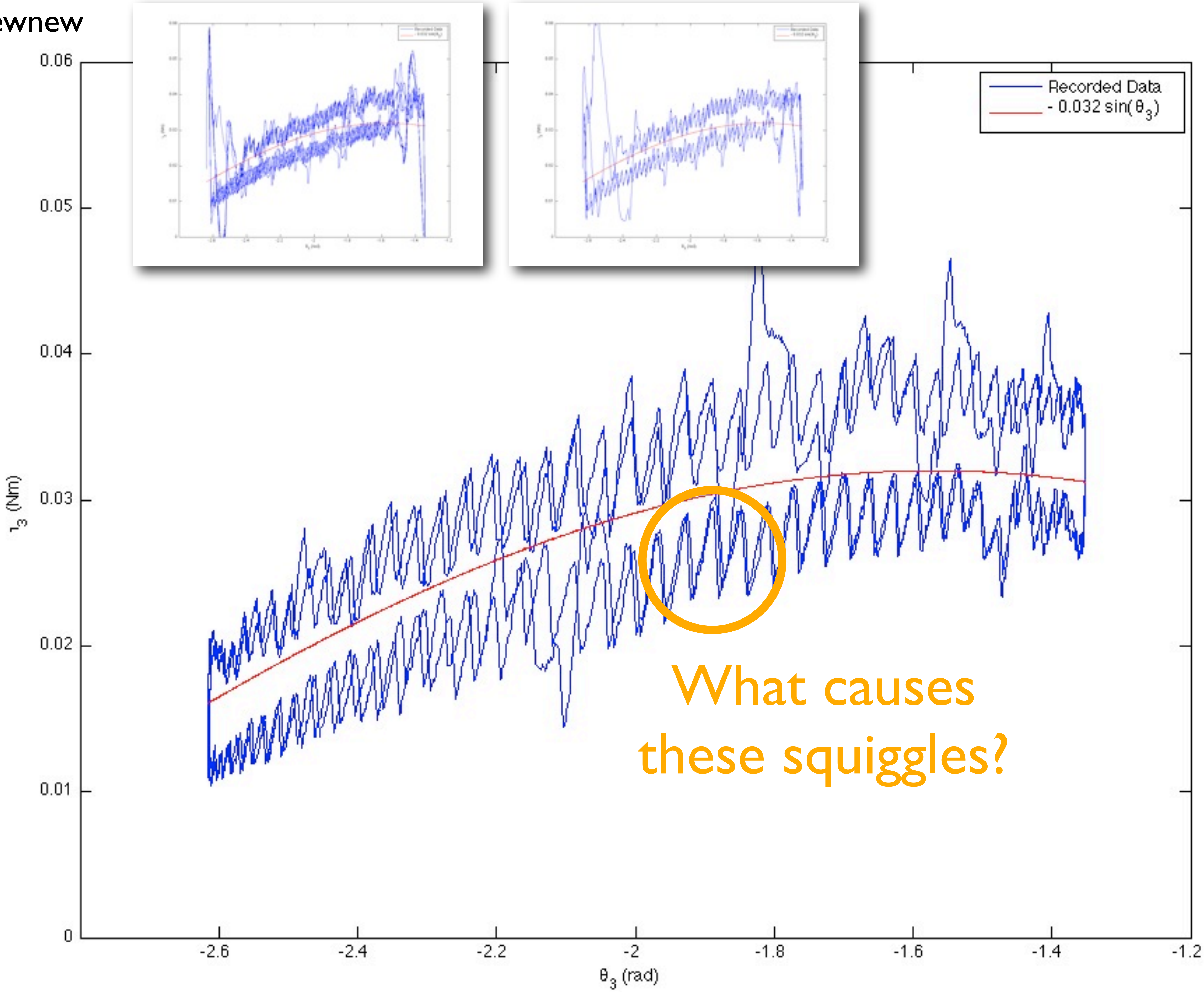


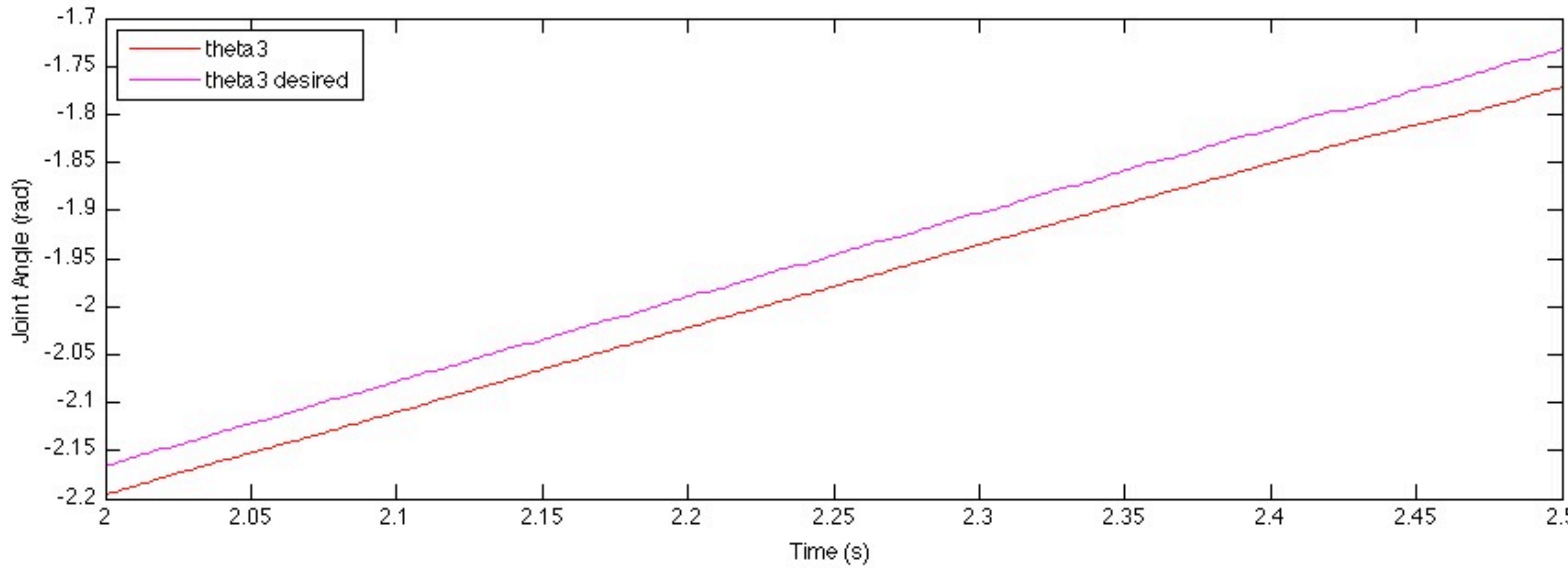
theta1
theta2
theta3
Joint Angle (rad)
Time (s)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
1
2
3
4
5
6
7
8
9
10

cal3newnew

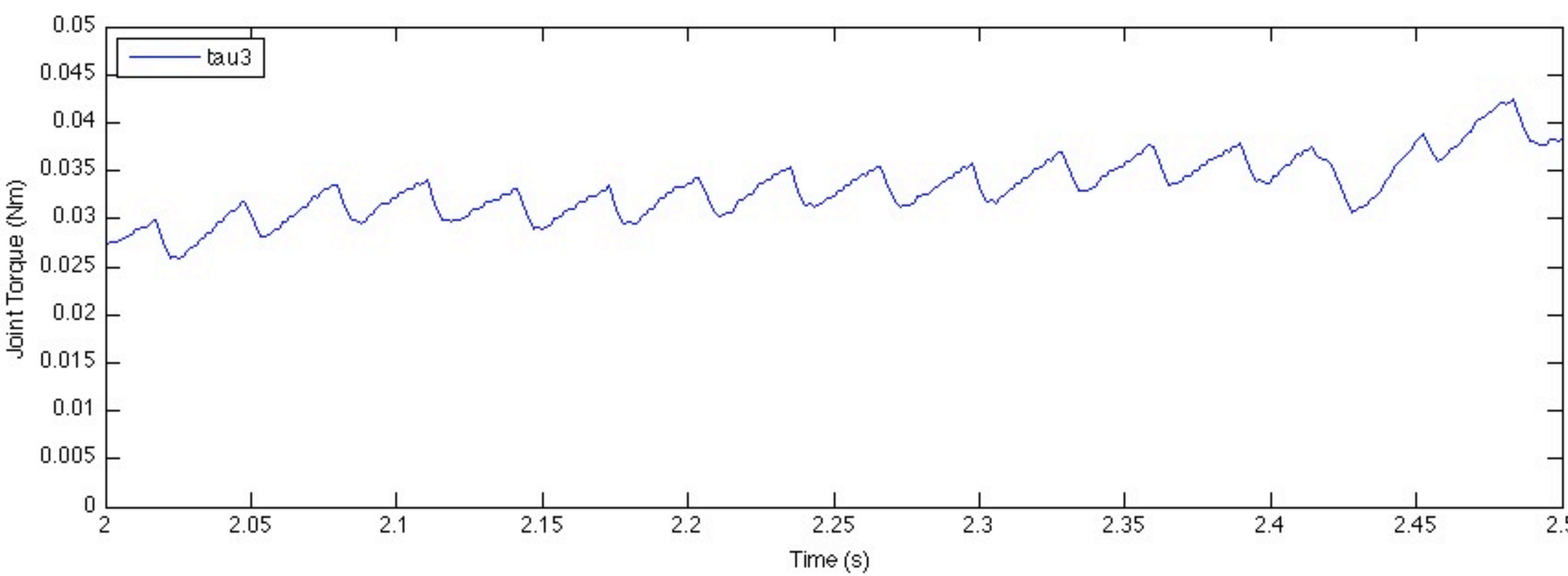

%% Load the desired trajectory
% desired mat file.
al3newnew
loaded joint angles as
sired = thetas;
ate number of cycles as
= length(thetas_desired)
ear the loaded variables fro
r t thetas
Define the PHANToM parameters
% Link lengths in millimeters.
l1 = 168;
l2 = 140;
= 140;

cal3newnew

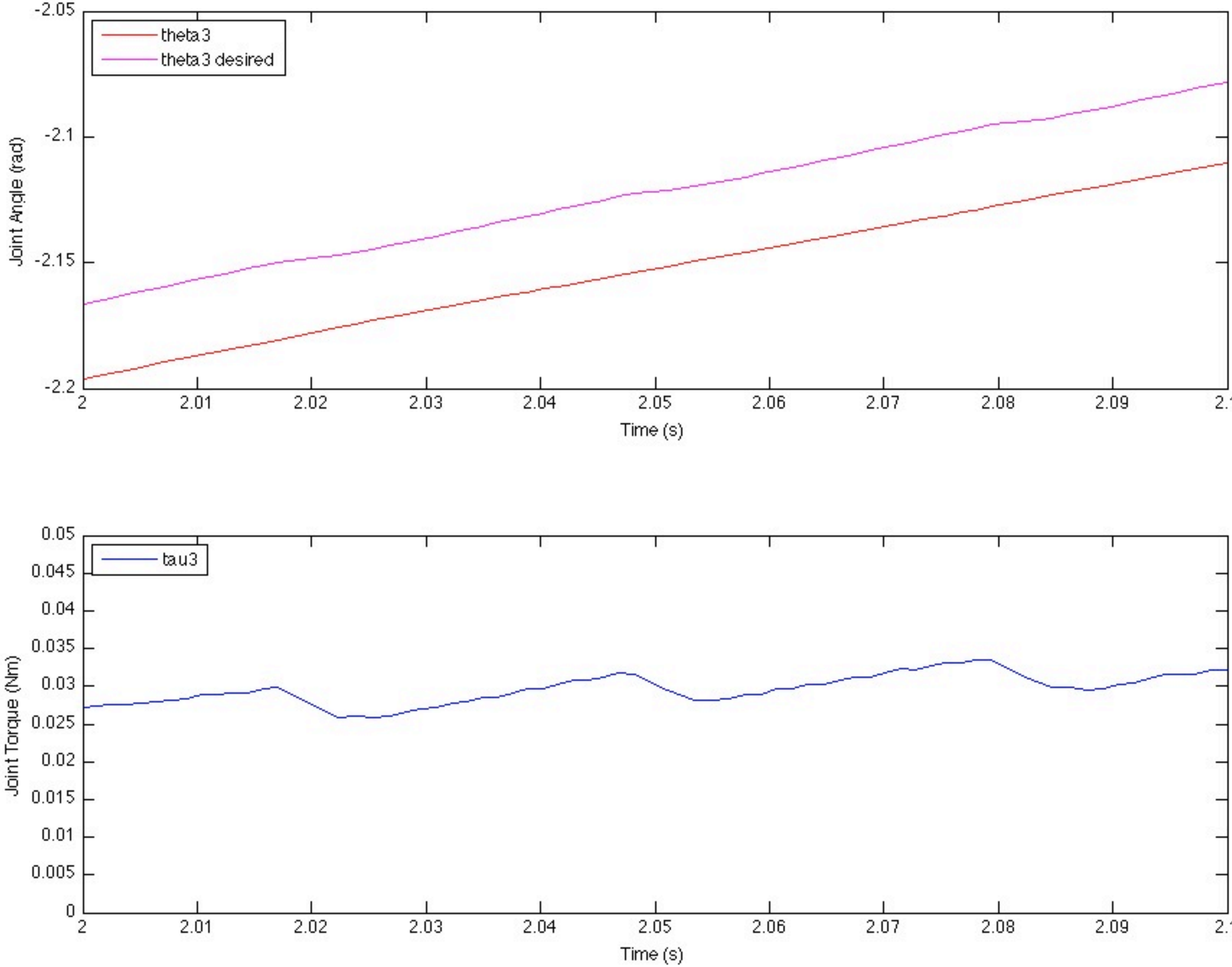

Recorded Data
- 0.032 sin(θ3)
0.06
0.05
0.04
0.03
0.02
0.01
0
τ3 (Nm)
-2.6
-2.4
-2.2
-2
-1.8
-1.6
-1.4
-1.2
θ3 (rad)
What causes these squiggles?



theta3
theta3 desired
Joint Angle (rad)
Time (s)



tau3
Joint Torque (Nm)
Time (s)



theta3
theta3 desired
Joint Angle (rad)
-2.05
-2.1
-2.15
-2.2
2
2.01
2.02
2.03
2.04
2.05
2.06
2.07
2.08
2.09
2.1
Time (s)
tau3
Joint Torque (Nm)
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
2
2.01
2.02
2.03
2.04
2.05
2.06
2.07
2.08
2.09
2.1
Time (s)

What is going on?

MEAM 520

https://piazza.com/class#fall2012/meam520/189

piazza | MEAM 520 | Q & A | Course Page | Manage Class | Katherine J....

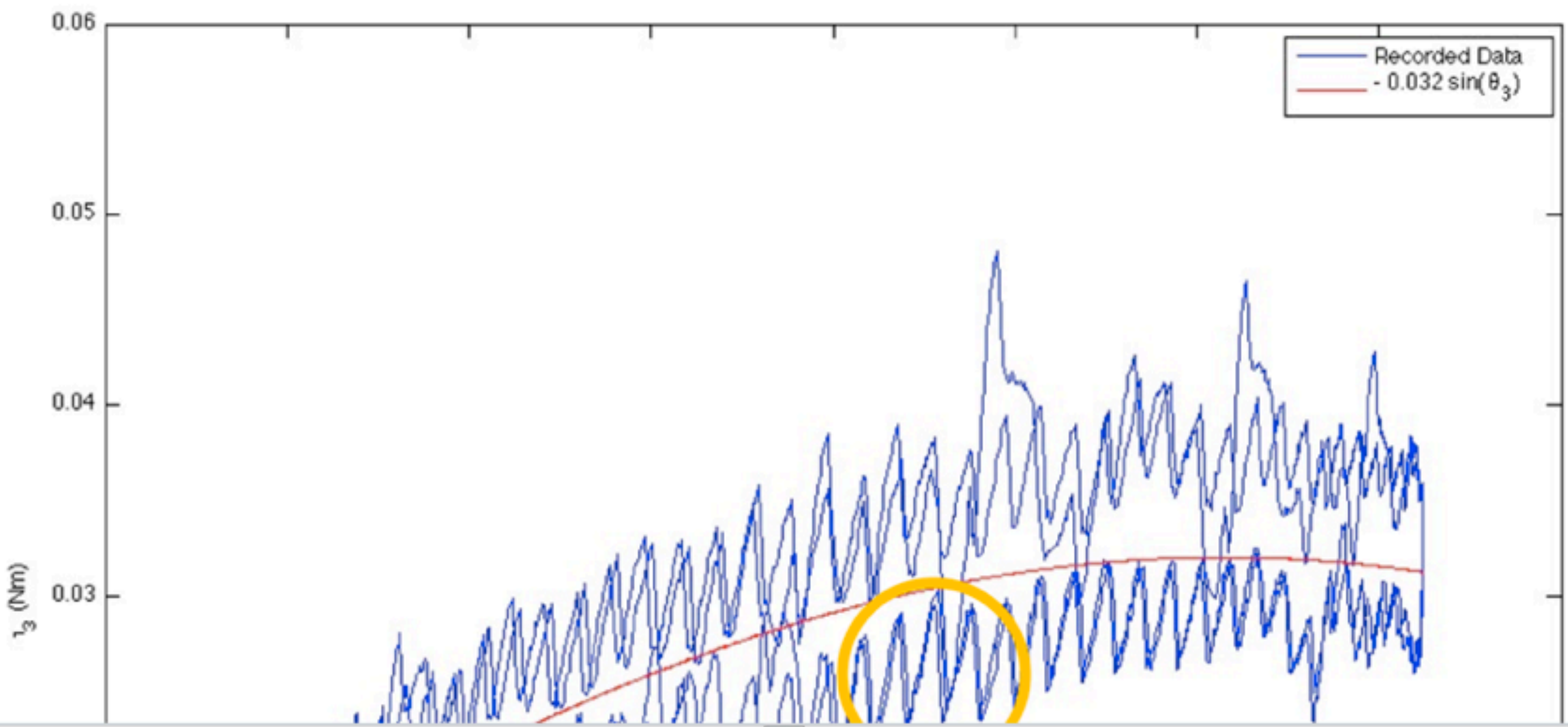
Note History:

note

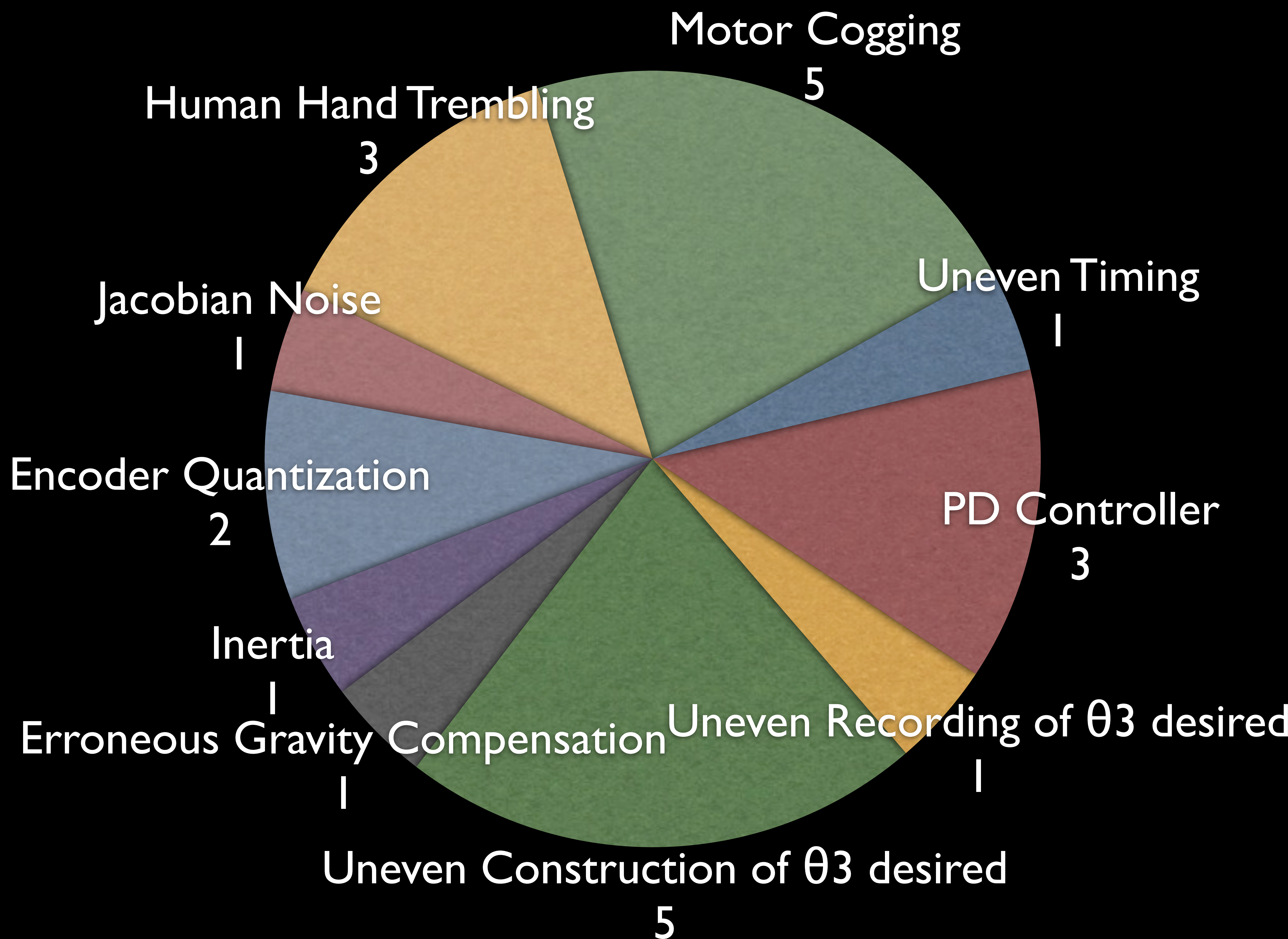
58 views

EXTRA CREDIT: What causes the squiggles?

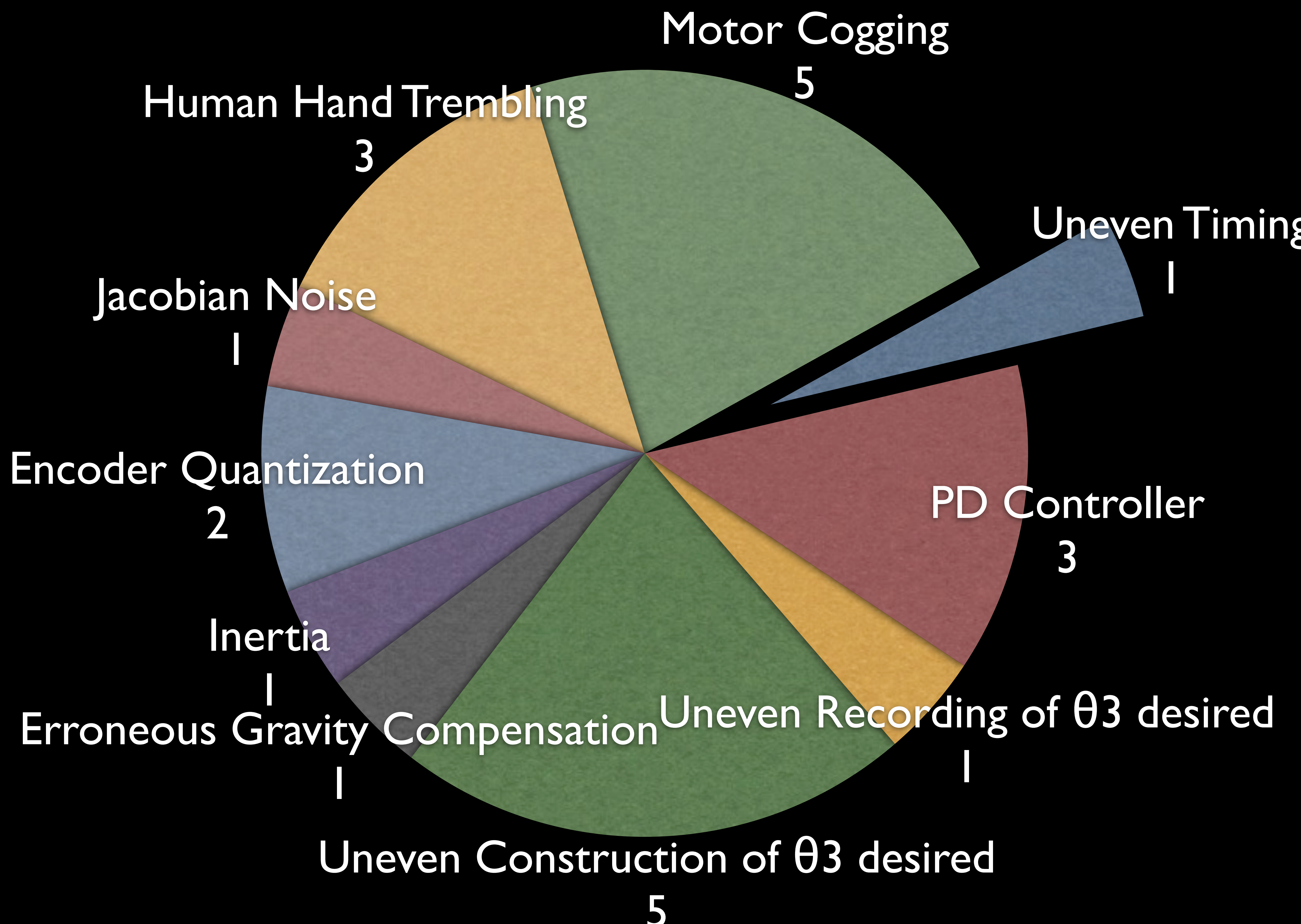
Today's lecture covered methods for adding gravity compensation to a robot. The data we obtained using the cubic trajectory looked much better than what we got with linear motion, but there were irritating squiggles. You can see them in the attached image and in the slides posted here:
http://medesign.seas.upenn.edu/uploads/Courses/robotics20dynamics.pdf

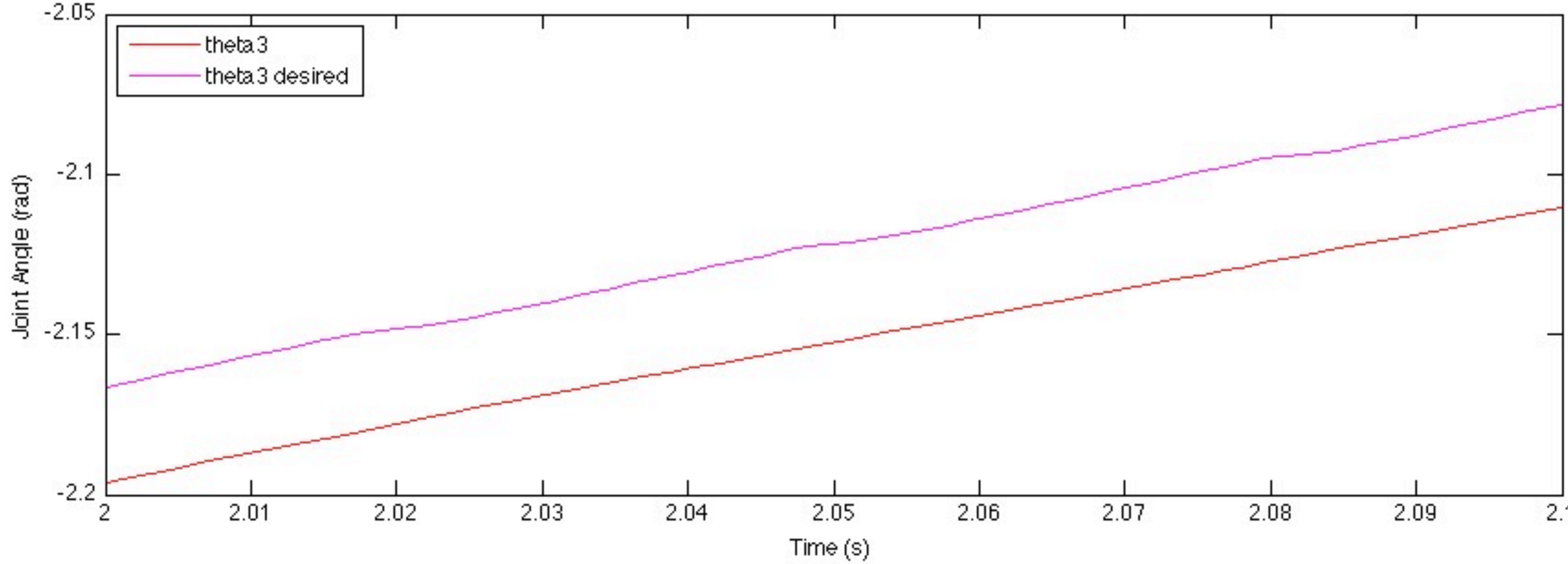


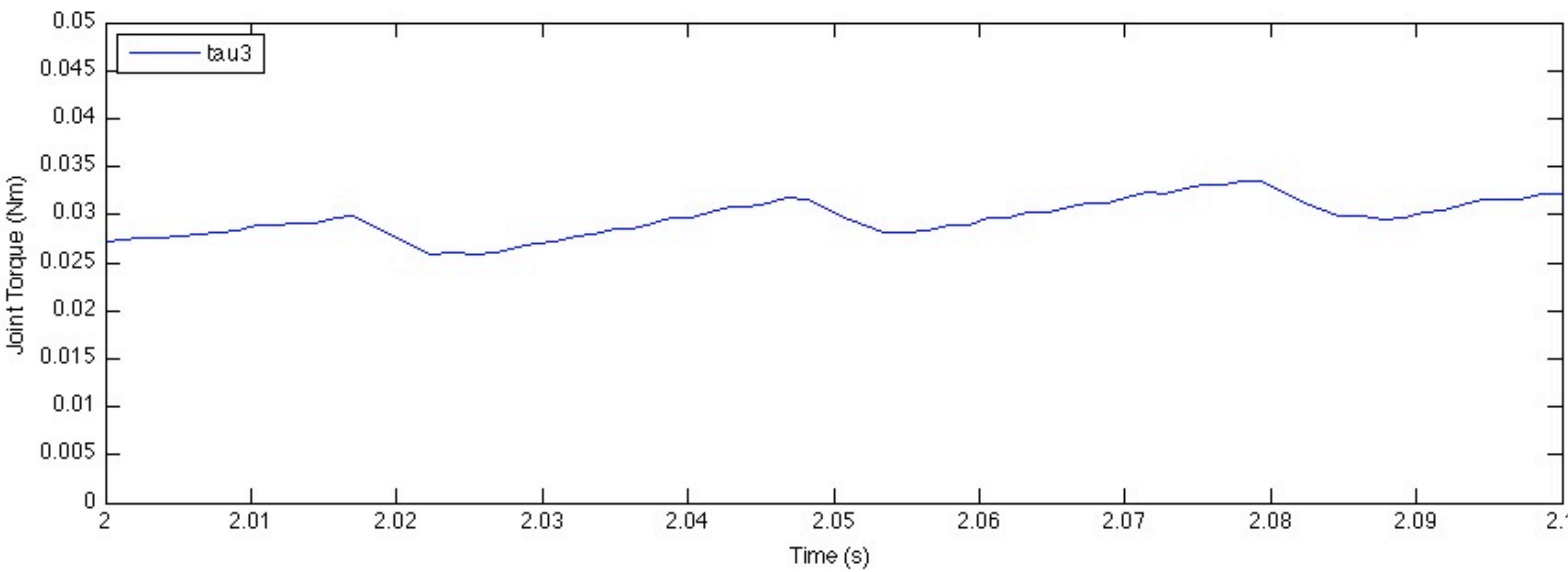
Squiggle Guesses

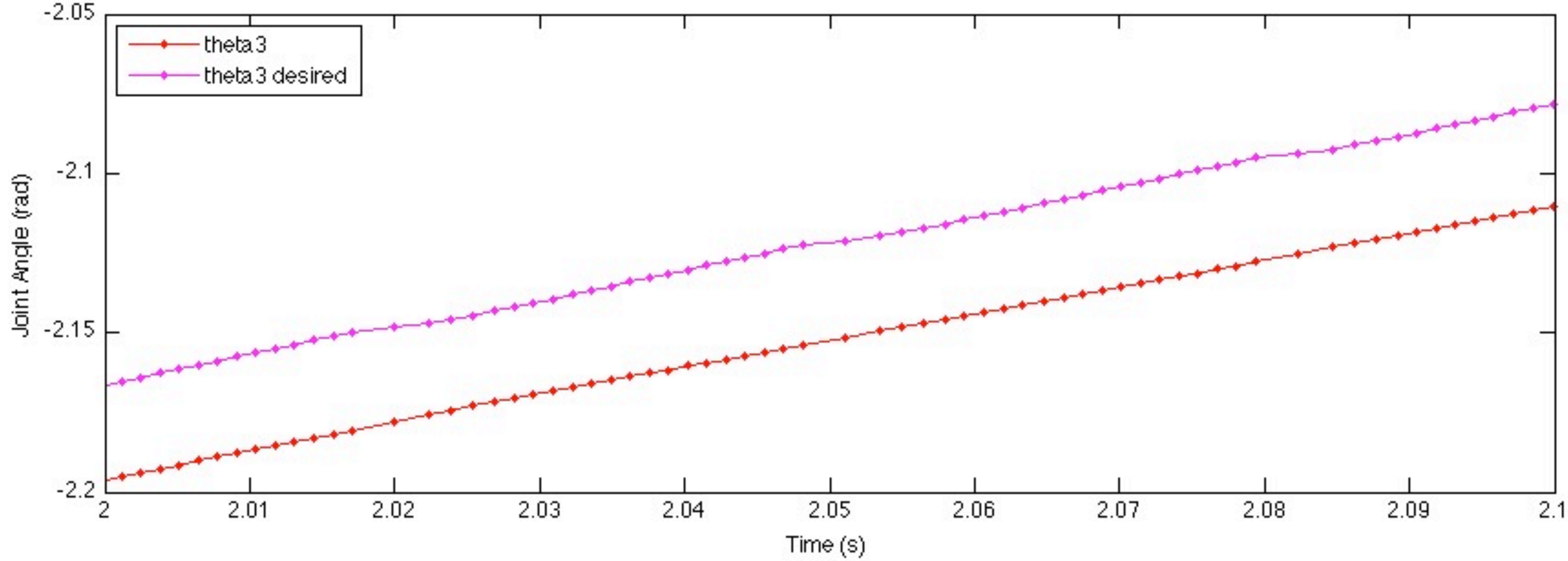


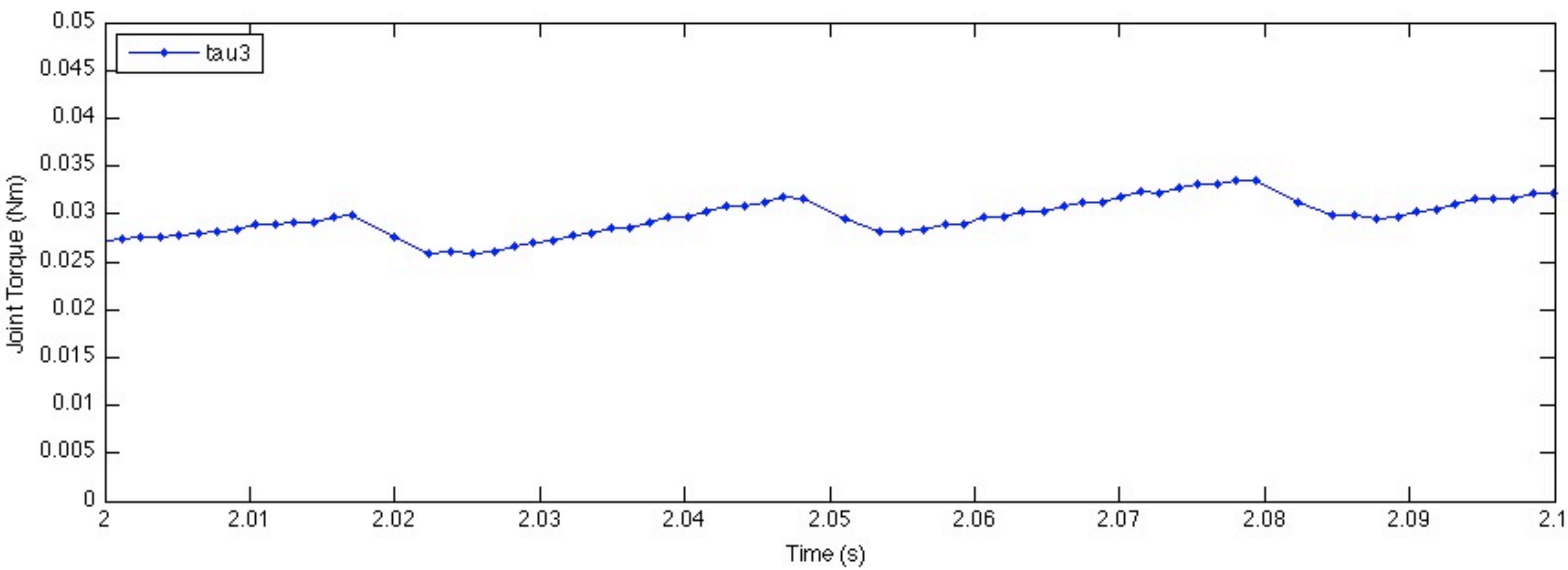
Squiggle Guesses

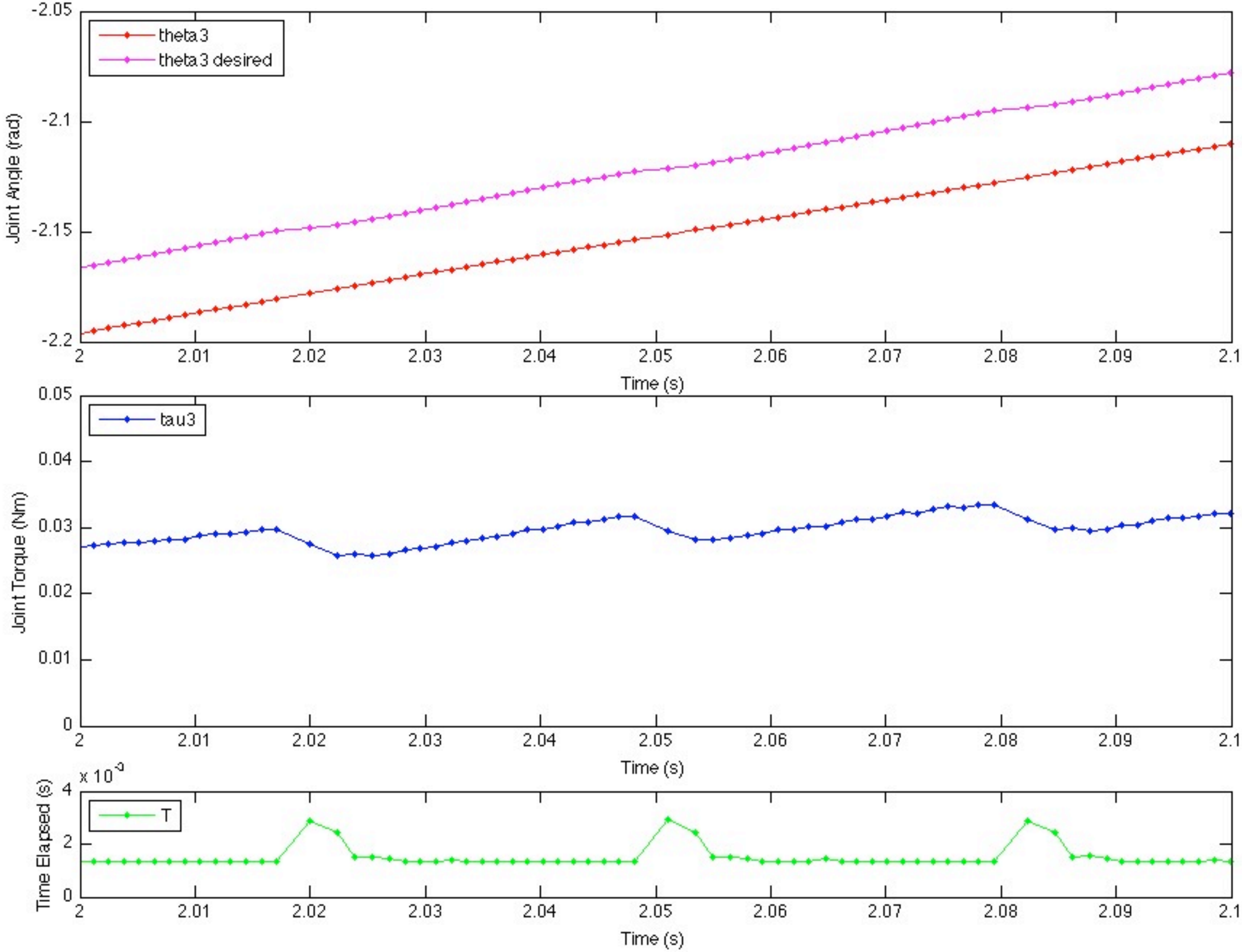



theta3
theta3 desired
Joint Angle (rad)
-2.05
-2.1
-2.15
-2.2
2
2.01
2.02
2.03
2.04
2.05
2.06
2.07
2.08
2.09
2.1
Time (s)

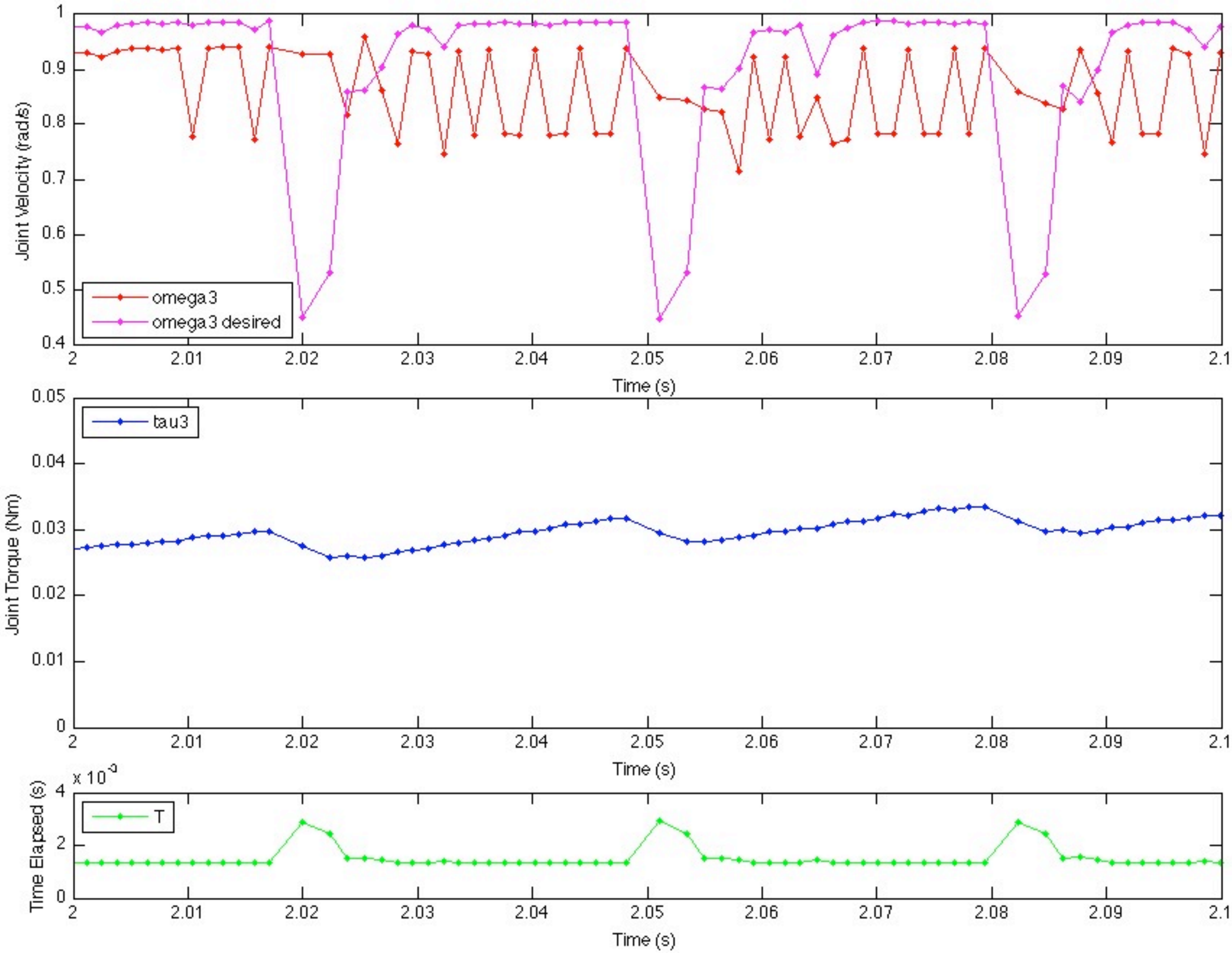

tau3
Joint Torque (Nm)
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
2
2.01
2.02
2.03
2.04
2.05
2.06
2.07
2.08
2.09
2.1
Time (s)


theta3
theta3 desired
Joint Angle (rad)
Time (s)
-2.05
-2.1
-2.15
-2.2
2
2.01
2.02
2.03
2.04
2.05
2.06
2.07
2.08
2.09
2.1

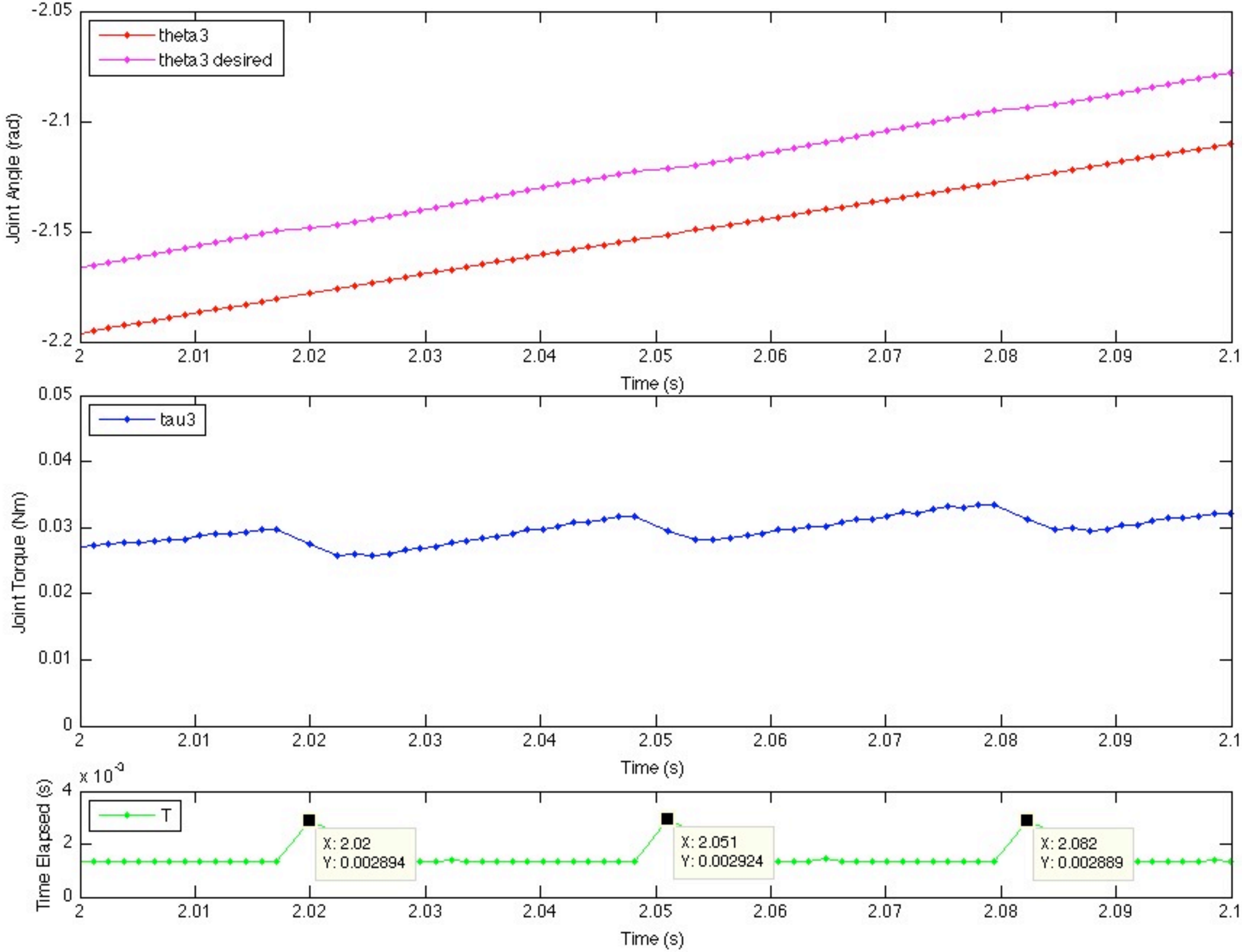

tau3
Joint Torque (Nm)
Time (s)
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
2
2.01
2.02
2.03
2.04
2.05
2.06
2.07
2.08
2.09
2.1



theta3
theta3 desired
Joint Angle (rad)
Time (s)
tau3
Joint Torque (Nm)
Time (s)
x 10^-3
T
Time Elapsed (s)
Time (s)



Joint Velocity (rad/s)
omega3
omega3 desired
Time (s)
Joint Torque (Nm)
tau3
Time (s)
x 10^-3
Time Elapsed (s)
T
Time (s)



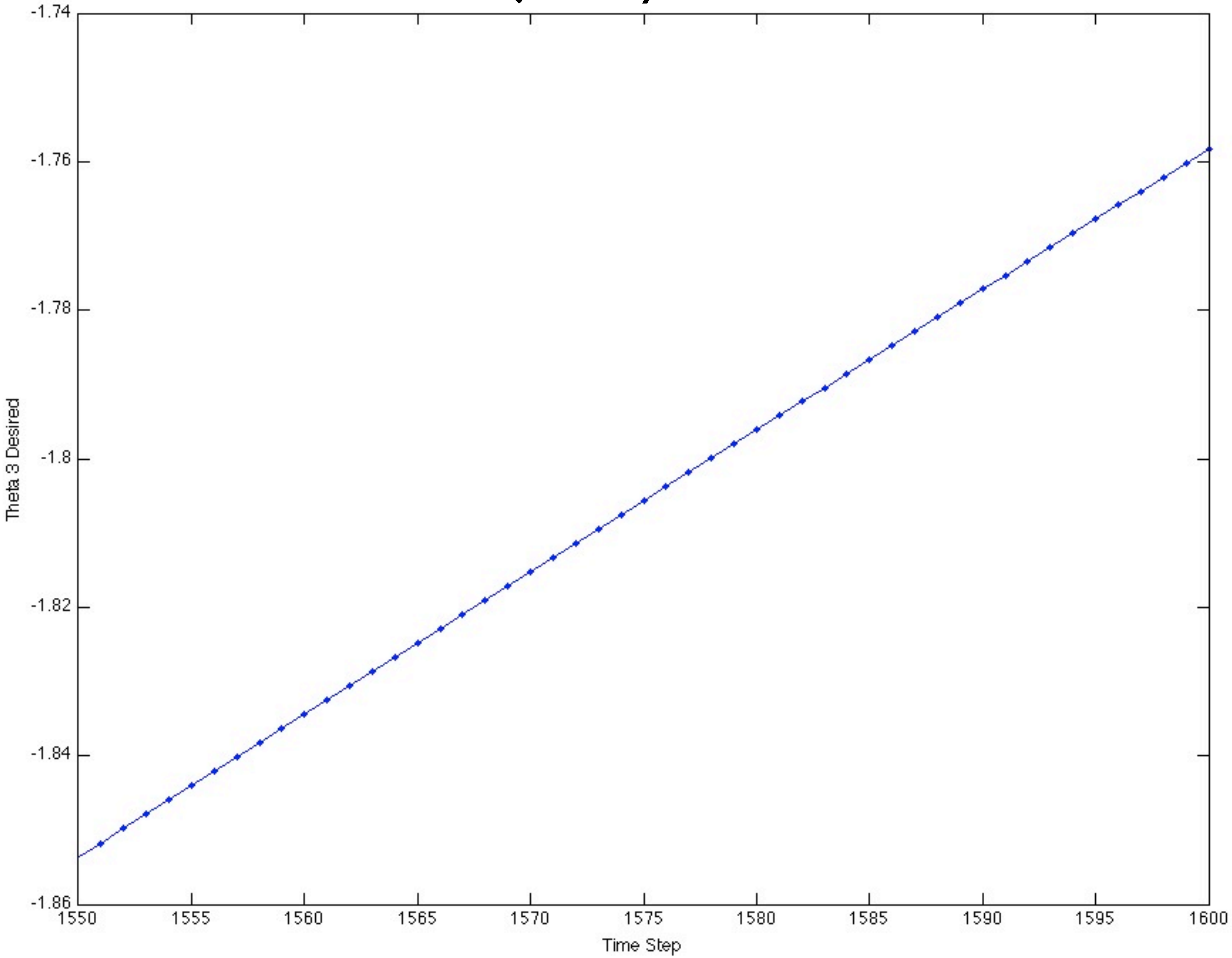
About 0.03 seconds apart... Why?

Editor – /Users/kuchenbe/Desktop/kjk/gravity_calibration.m

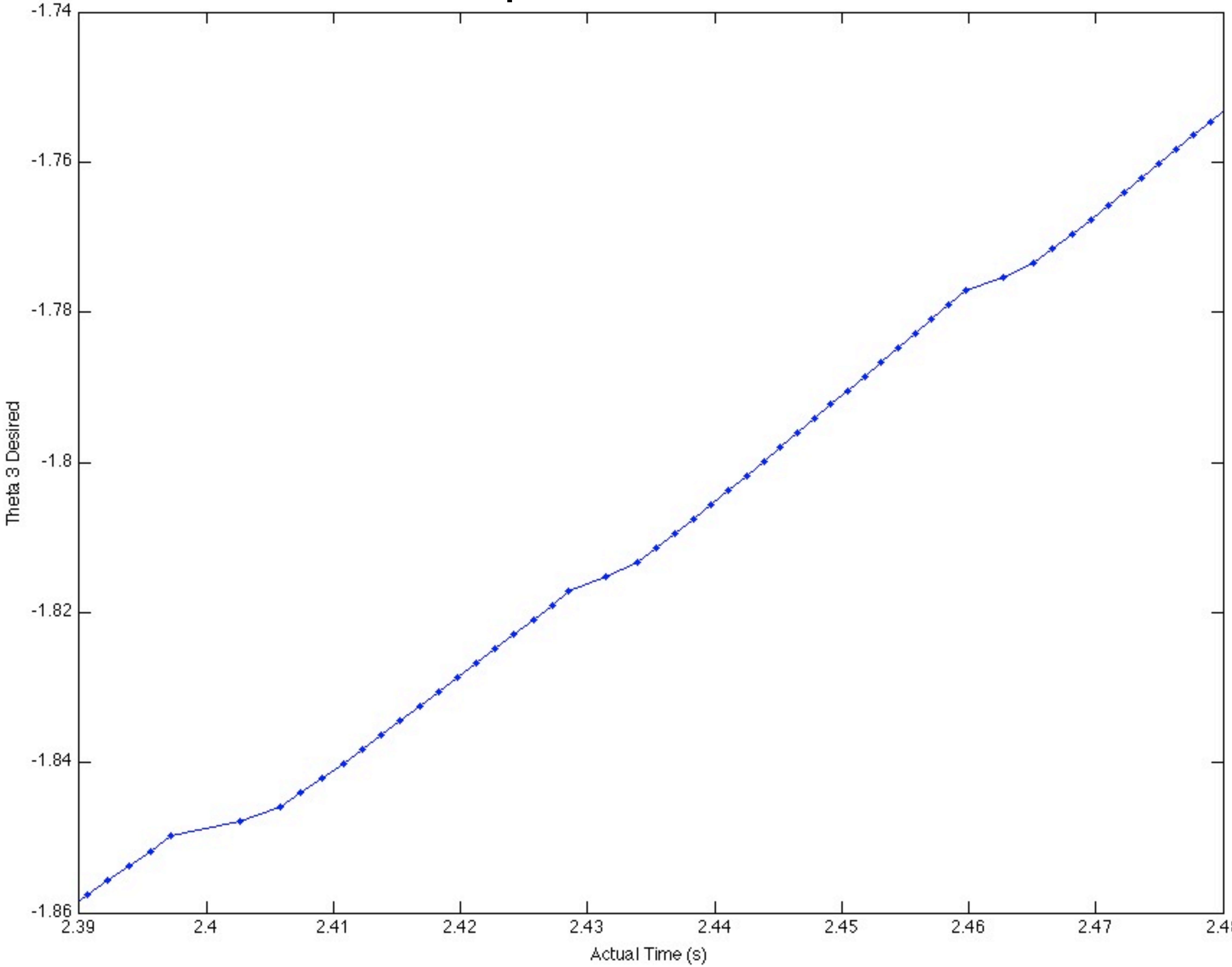
```matlab
    % If you're getting a lot of warnings for asking for too high of j
    % torques, you can turn that warning off by uncommenting a line ne
    % the top of this script.
    phantomJointTorques(tau123(1), tau123(2), tau123(3));

    % Check how much time has elapsed since we last updated the graphi
    if (t(i) - lastGraphicsTime > 0.03)
        % Enough time has passed.

        % Update the graph by setting the data for the PHANToM's dot t
        % position of the haptic device.
        set(hPhantomDot, 'xdata', hx, 'ydata', hy, 'zdata', hz)

        % Update the graph by setting the data for the desired dot to
        % desired position of the haptic device.
        set(hDesiredDot, 'xdata', hxdes, 'ydata', hydes, 'zdata', hzde

        % Update the graph by setting the data for the force line to s
        % scaled version of the commanded force.
        set(hForceLine,'xdata',[hx hx+Fx*fScale],'ydata',[hy hy+Fy*fSc

        % Store this time for future comparisons.
        lastGraphicsTime = t(i);
    end

    % Pause for one millisecond to keep things moving at a reasonable
    pause(.002);
end
```

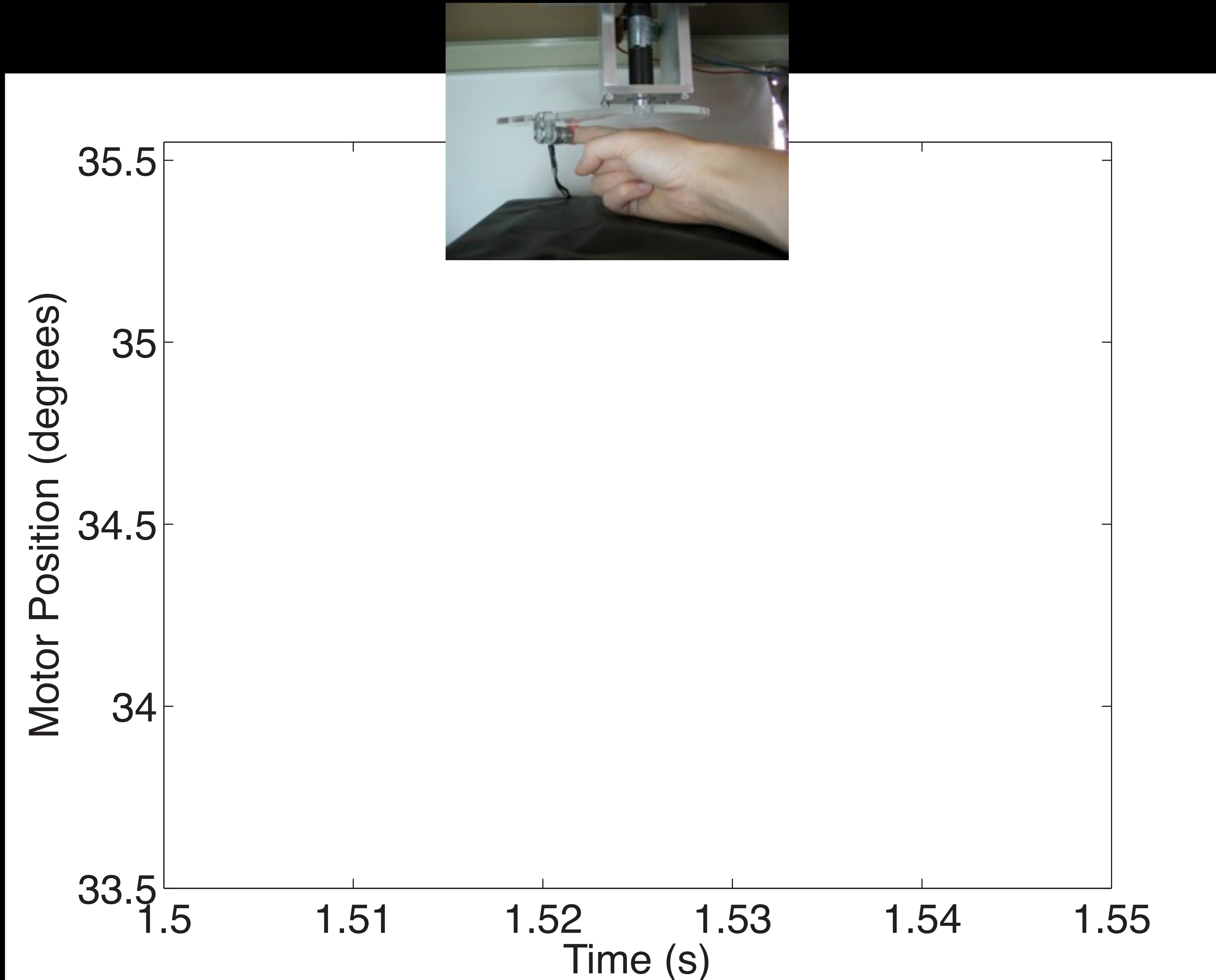
Updating the graphics takes time, delaying the start of the next servo loop.

Pre-calculated Trajectory for Theta 3 Desired

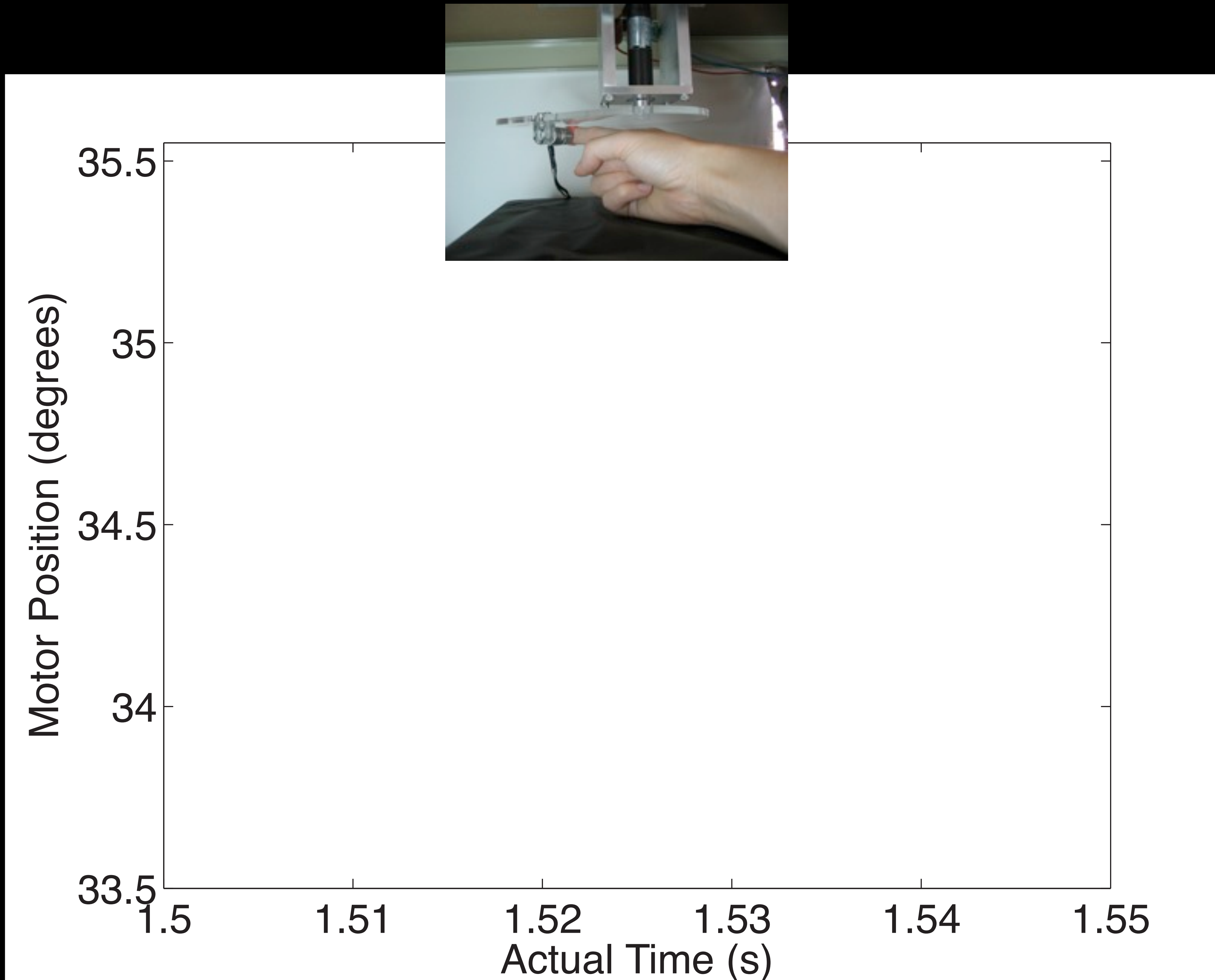


Actual Output of Theta 3 Desired





Motor Position (degrees)
35.5
35
34.5
34
33.5
1.5
1.51
1.52
1.53
1.54
1.55
Time (s)


Motor Position (degrees)
35.5
35
34.5
34
33.5
1.5
1.51
1.52
1.53
1.54
1.55
Actual Time (s)

Editor – /Users/kuchenbe/Desktop/kjk/gravity_calibration.m

```matlab
        % If you're getting a lot of warnings for asking for too high of 
        % torques, you can turn that warning off by uncommenting a line ne
        % the top of this script.
        phantomJointTorques(tau123(1), tau123(2), tau123(3));

        % Check how much time has elapsed since we last updated the graphi
        if (t(i) - lastGraphicsTime > 0.03)
            % Enough time has passed.

            % Update the graph by setting the data for the PHANToM's dot t
            % position of the haptic device.
            set(hPhantomDot, 'xdata', hx, 'ydata', hy, 'zdata', hz)

            % Update the graph by setting the data for the desired dot to
            % desired position of the haptic device.
            set(hDesiredDot, 'xdata', hxdes, 'ydata', hydes, 'zdata', hzde

            % Update the graph by setting the data for the force line to s
            % scaled version of the commanded force.
            set(hForceLine,'xdata',[hx hx+Fx*fScale],'ydata',[hy hy+Fy*fSc

            % Store this time for future comparisons.
            lastGraphicsTime = t(i);
        end

        % Pause for one millisecond to keep things moving at a reasonable 
        pause(.002);
    end
```

How can we stabilize the timing?

plot_gravity.m | gravity_calibration.m

script | Ln 249 Col 8

Editor - /Users/kuchenbe/Desktop/kjk/gravity_calibration.m

```matlab
    % If you're getting a lot of warnings for asking for too high of 
    % torques, you can turn that warning off by uncommenting a line ne
    % the top of this script.
    phantomJointTorques(tau123(1), tau123(2), tau123(3));

    if (false)
    % Check how much time has elapsed since we last updated the graphi
    if (t(i) - lastGraphicsTime > 0.03)
        % Enough time has passed.

        % Update the graph by setting the data for the PHANToM's dot t
        % position of the haptic device.
        set(hPhantomDot, 'xdata', hx, 'ydata', hy, 'zdata', hz)

        % Update the graph by setting the data for the desired dot to 
        % desired position of the haptic device.
        set(hDesiredDot, 'xdata', hxdes, 'ydata', hydes, 'zdata', hzde

        % Update the graph by setting the data for the force line to s
        % scaled version of the commanded force.
        set(hForceLine,'xdata',[hx hx+Fx*fScale],'ydata',[hy hy+Fy*fSc

        % Store this time for future comparisons.
        lastGraphicsTime = t(i);
    end
    end

    % Pause for one millisecond to keep things moving at a reasonable
```

Don't display graphics!

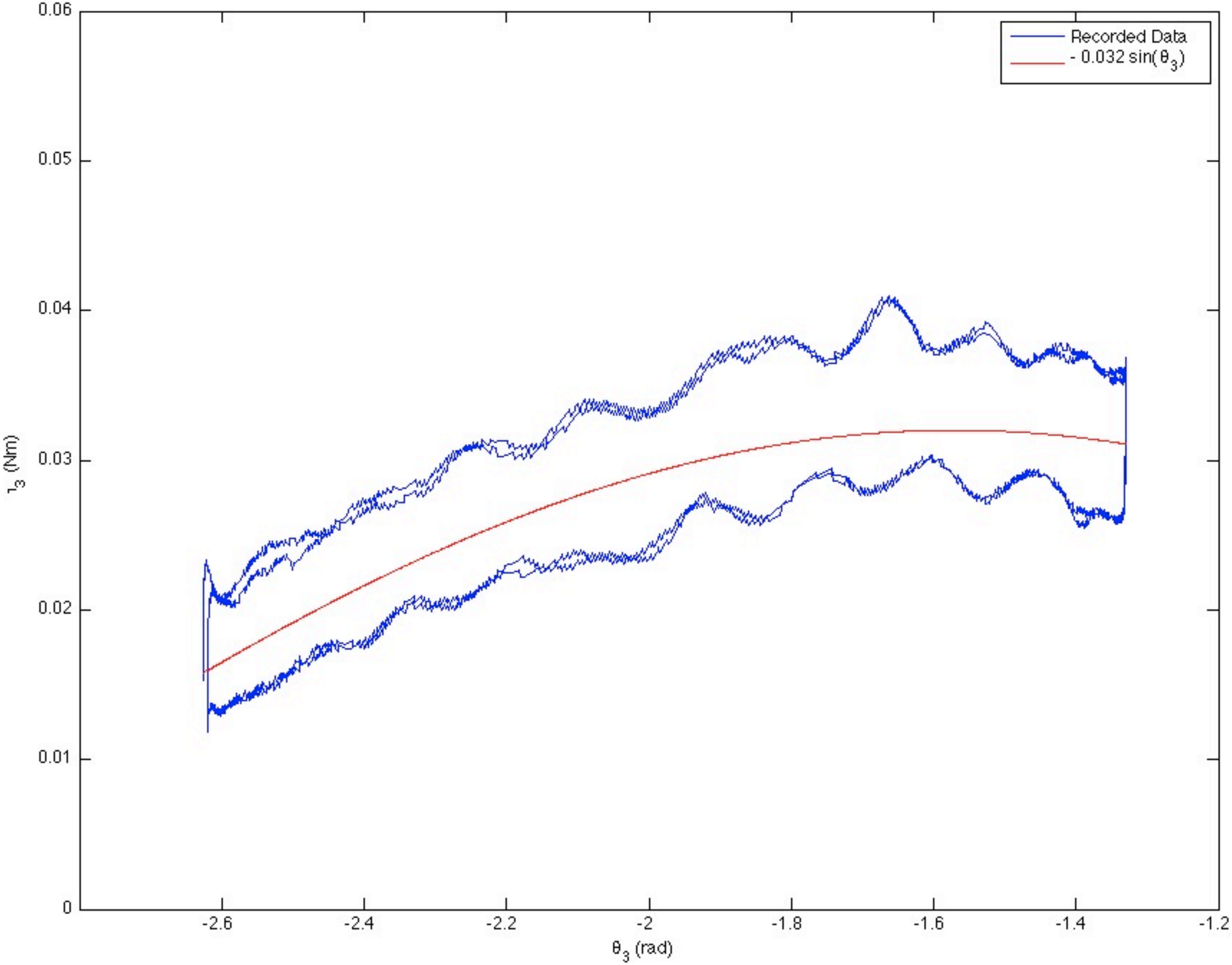
Even better would be to measure time and calculate desired position accordingly...

plot_gravity.m | gravity_calibration.m

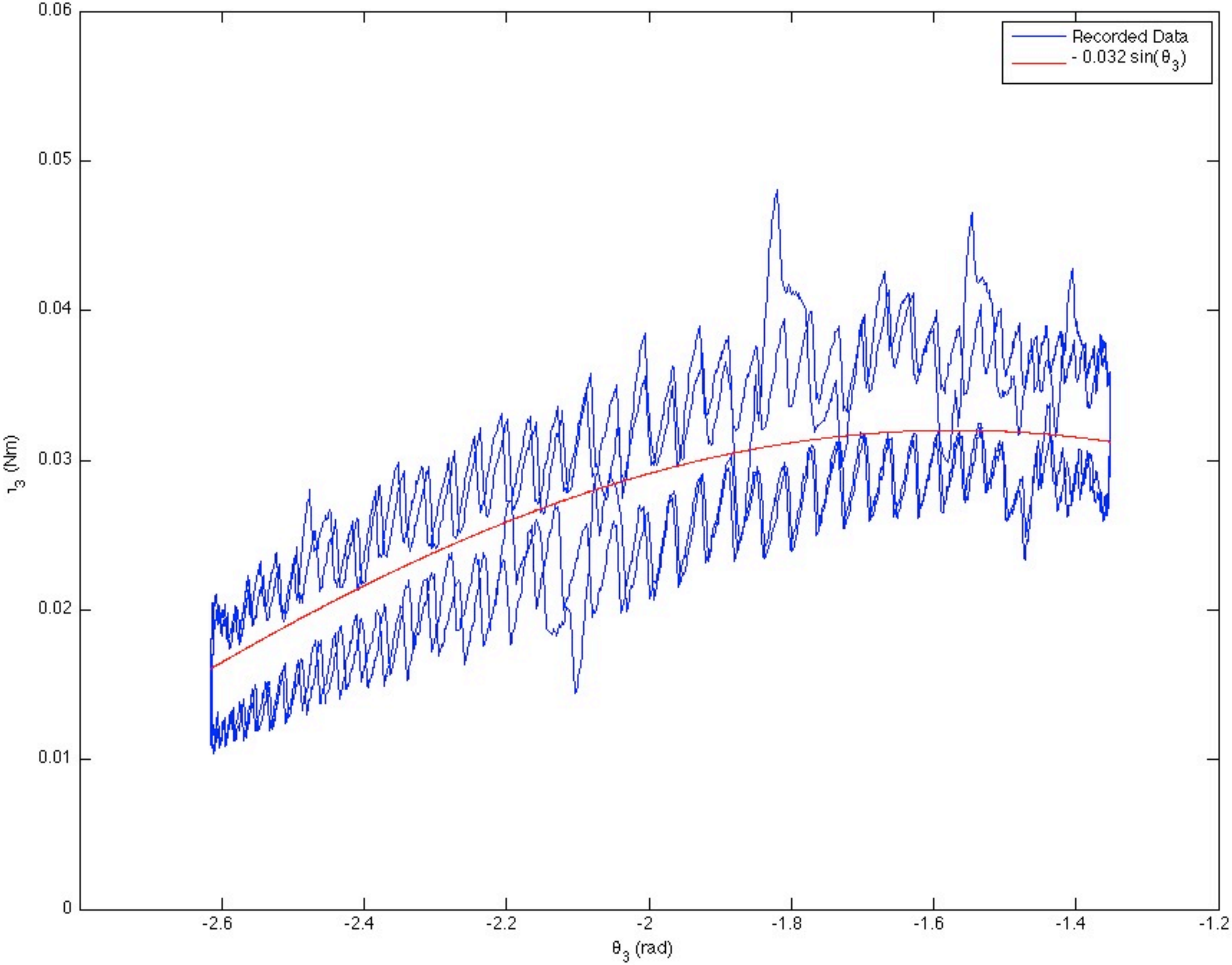
3 usages of "lastGraphicsTime" found | script | Ln 233 Col 40

cal3newnewb

cal3newnewb

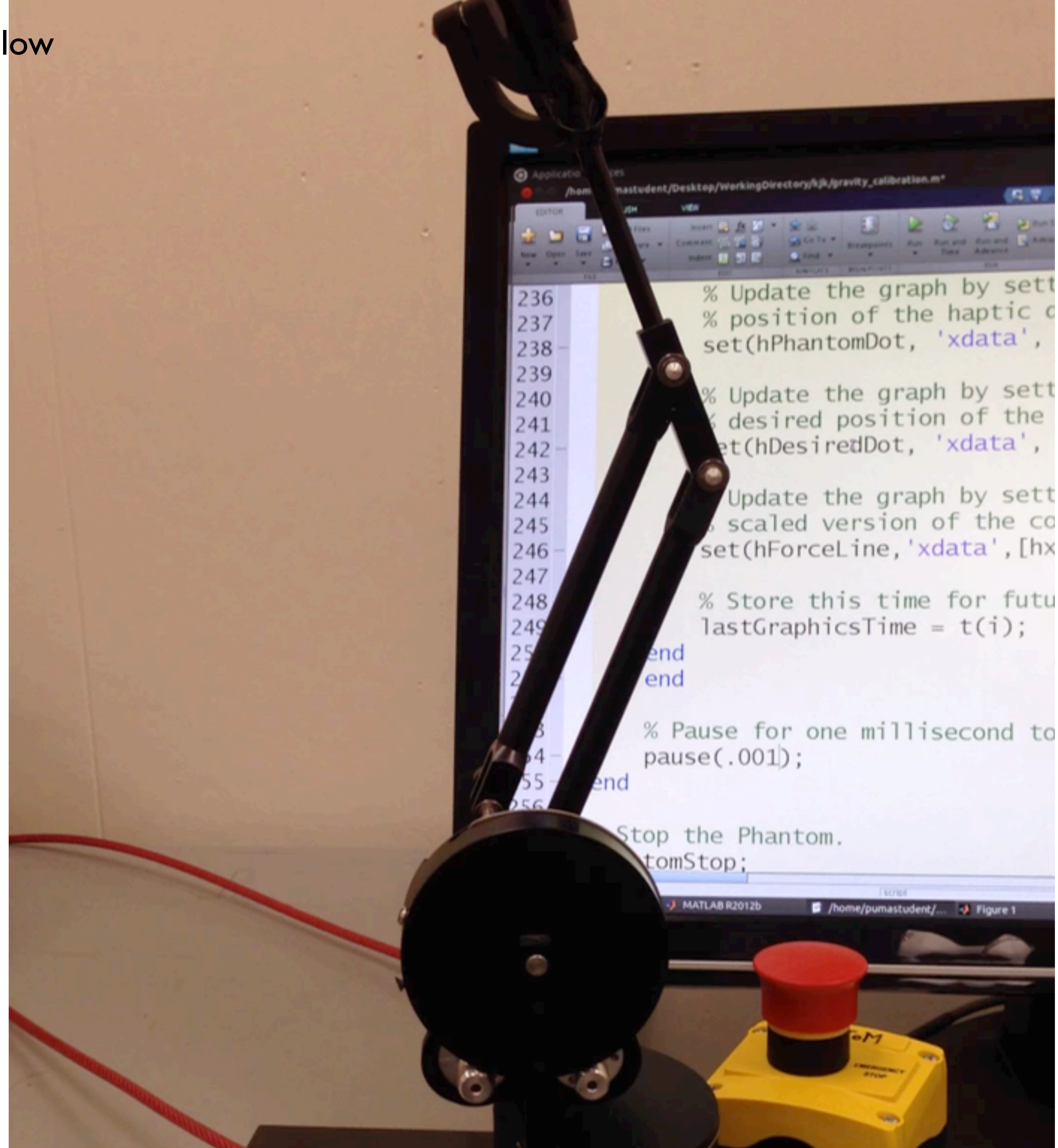


cal3newnew

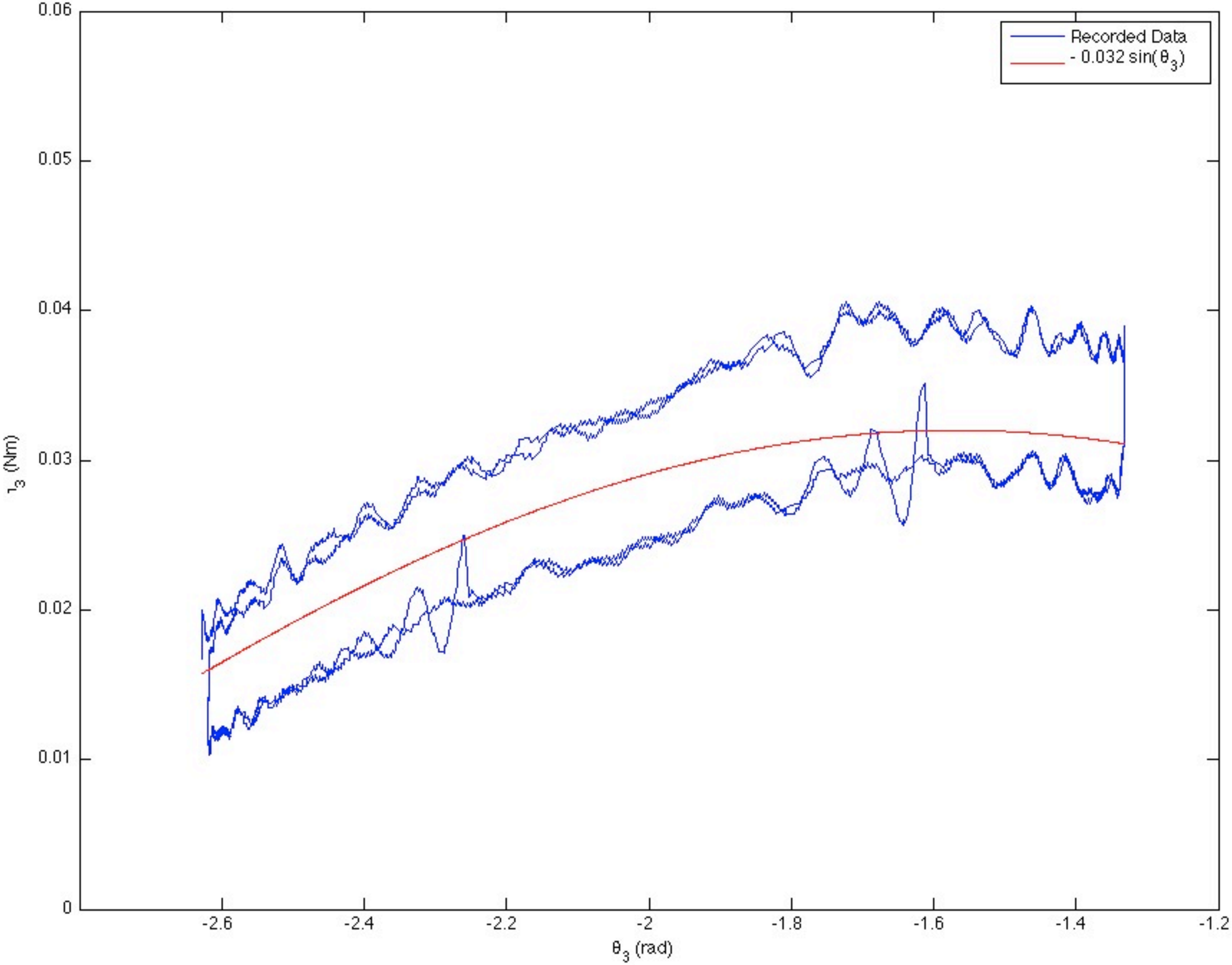


Recorded Data
- 0.032 sin(θ3)
τ3 (Nm)
θ3 (rad)
0.06
0.05
0.04
0.03
0.02
0.01
0
-2.6
-2.4
-2.2
-2
-1.8
-1.6
-1.4
-1.2

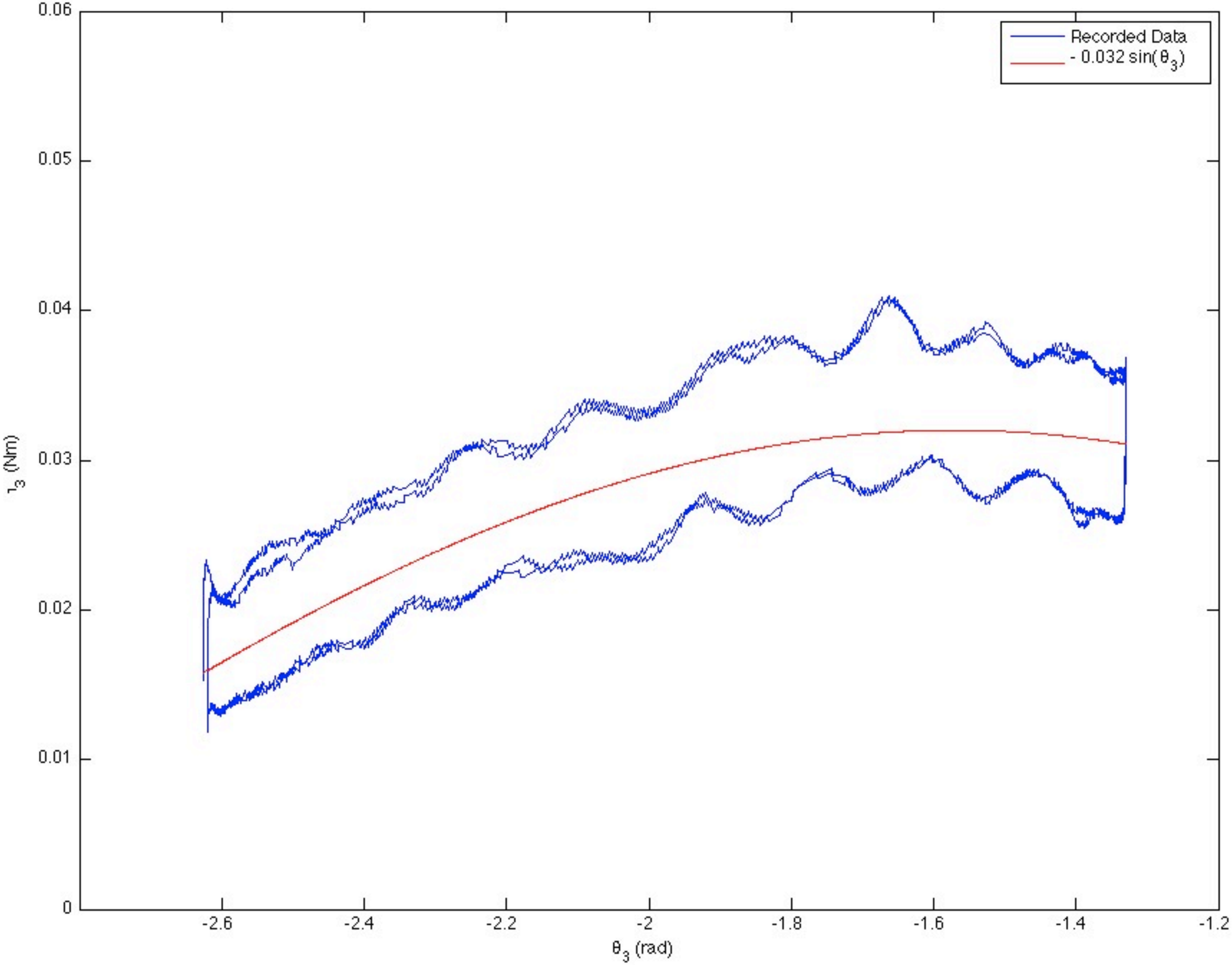
cal3newnewbslow



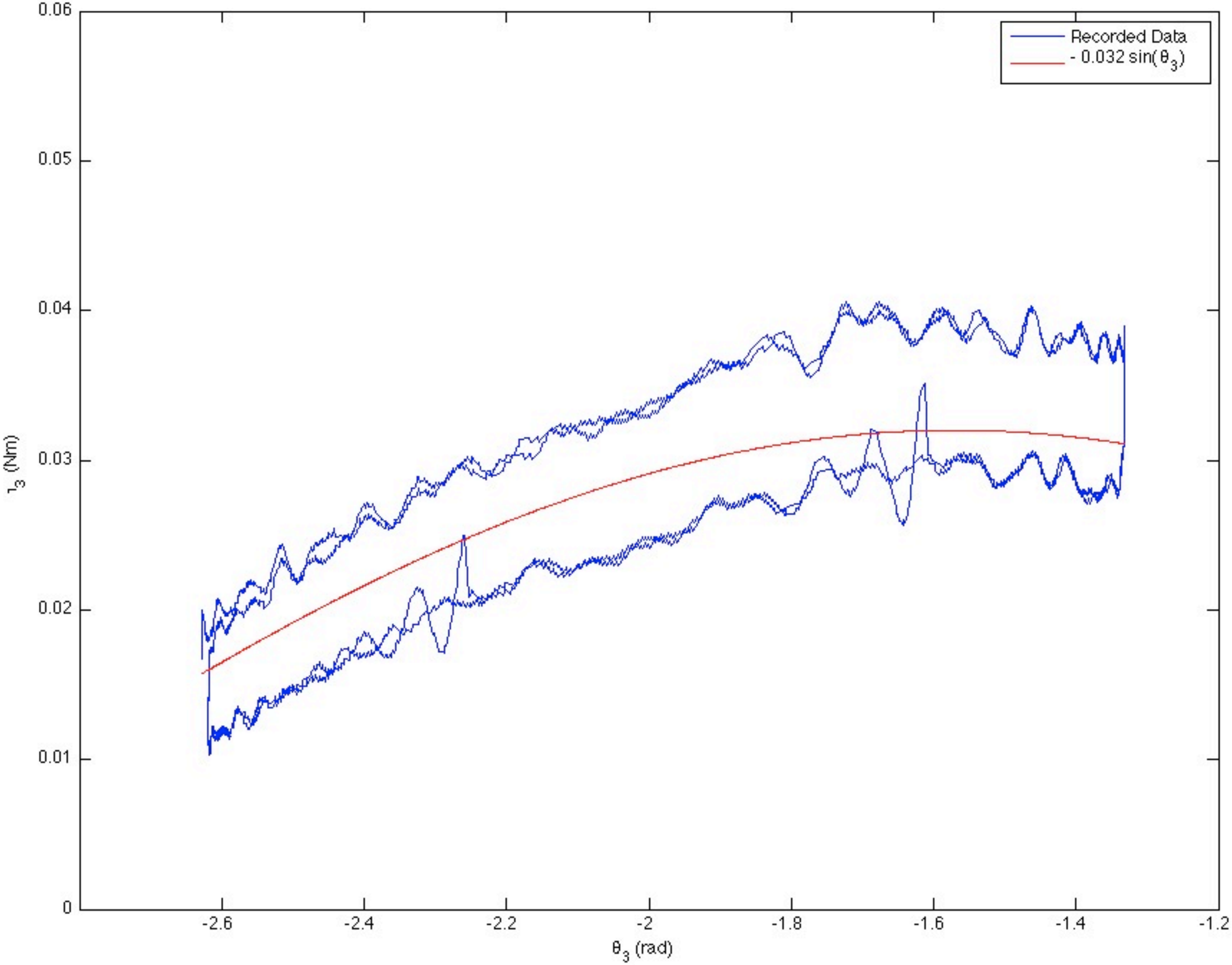
cal3newnewbslow



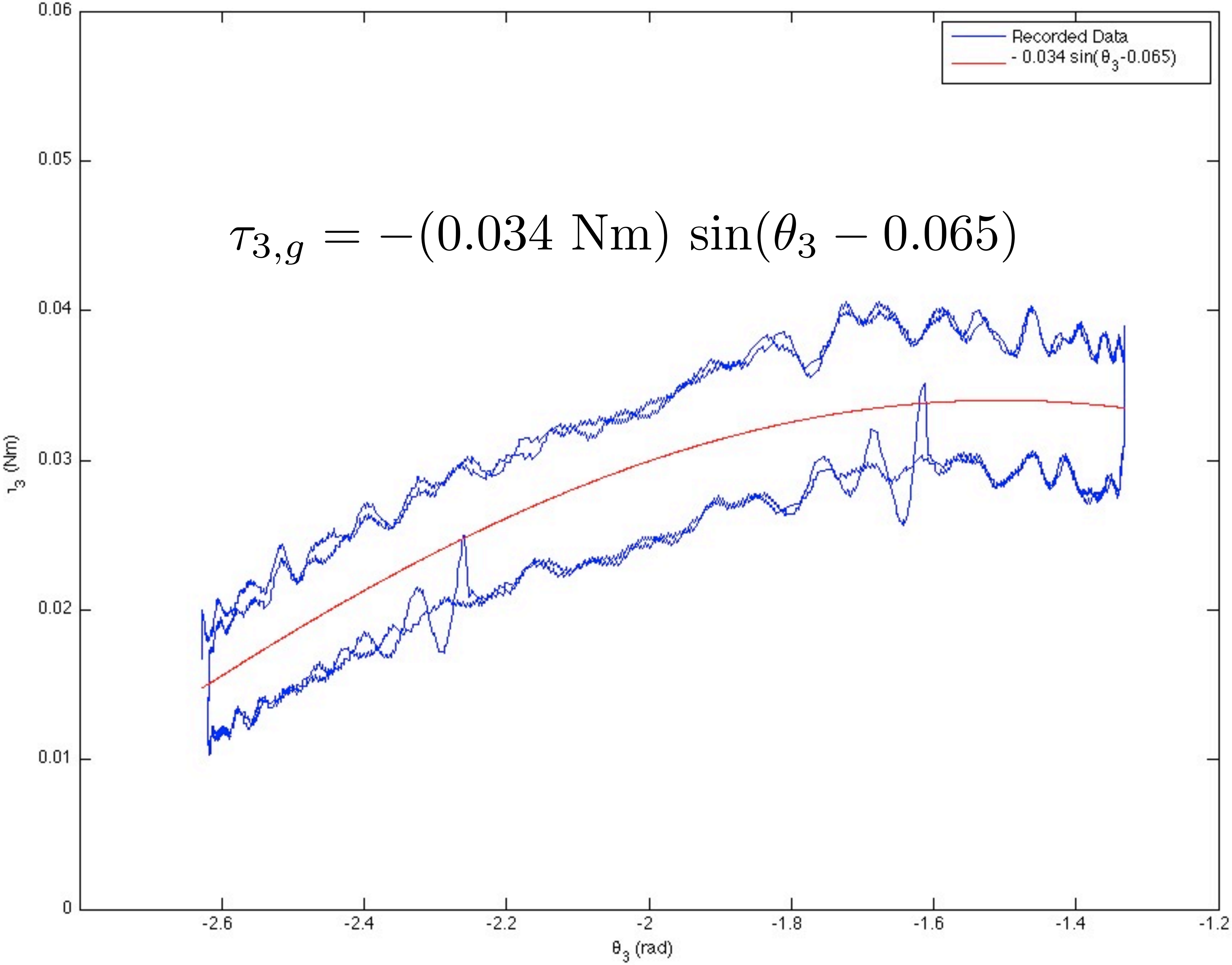
cal3newnewb



cal3newnewb slow



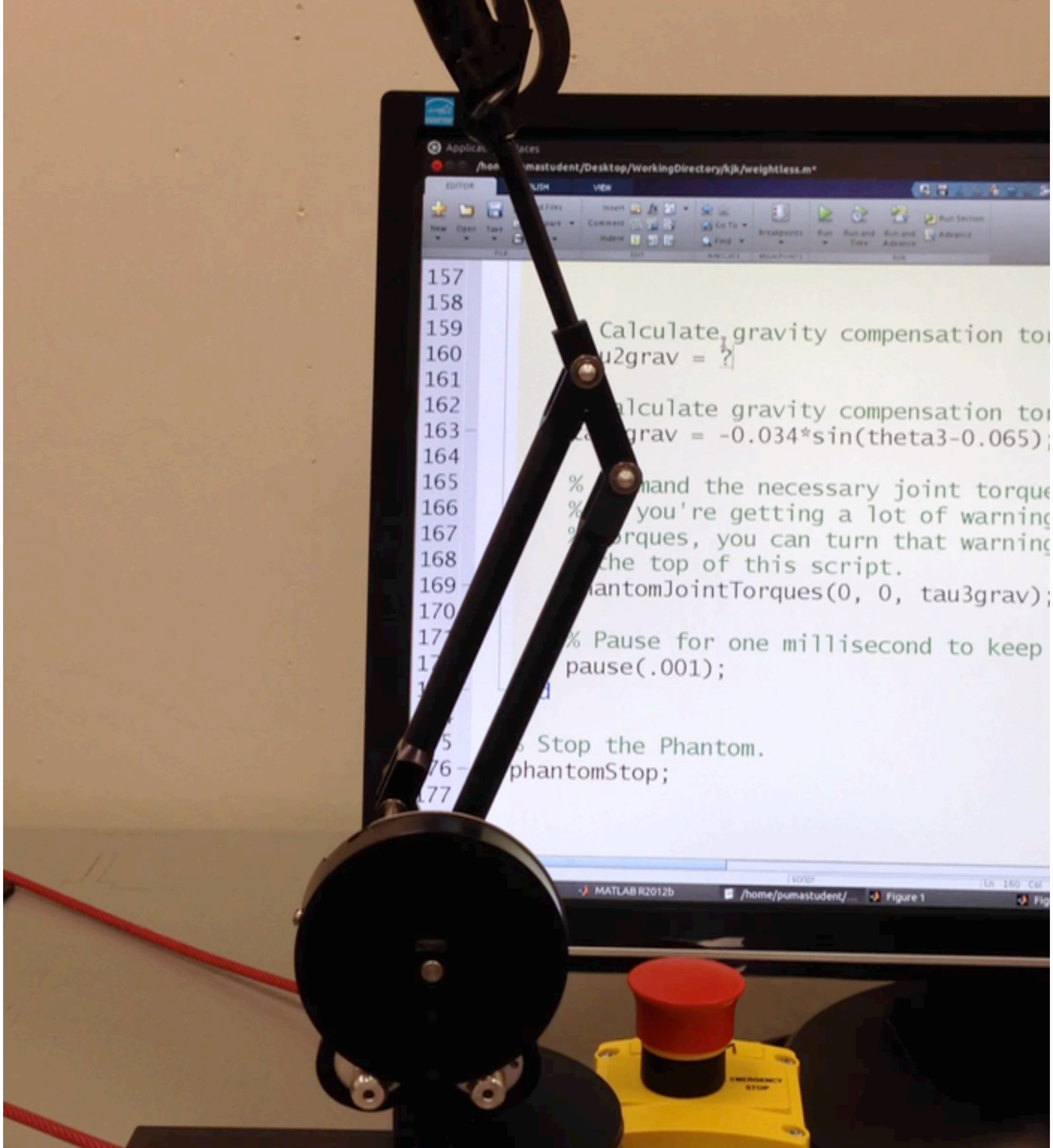
cal3newnewb slow refit




% Calculate gravity compensation tor
tau3grav = -0.034*sin(theta3-0.065);
Command the necessary joint torque
f you're getting a lot of warning
rques, you can turn that warning
e top of this script.
omJointTorques(0, 0, tau3grav);
use for one millisecond to keep
se(.001);
end
p the Phantom.
tomStop;
MATLAB R2012b
Figure 1

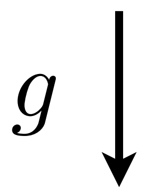
Questions ?

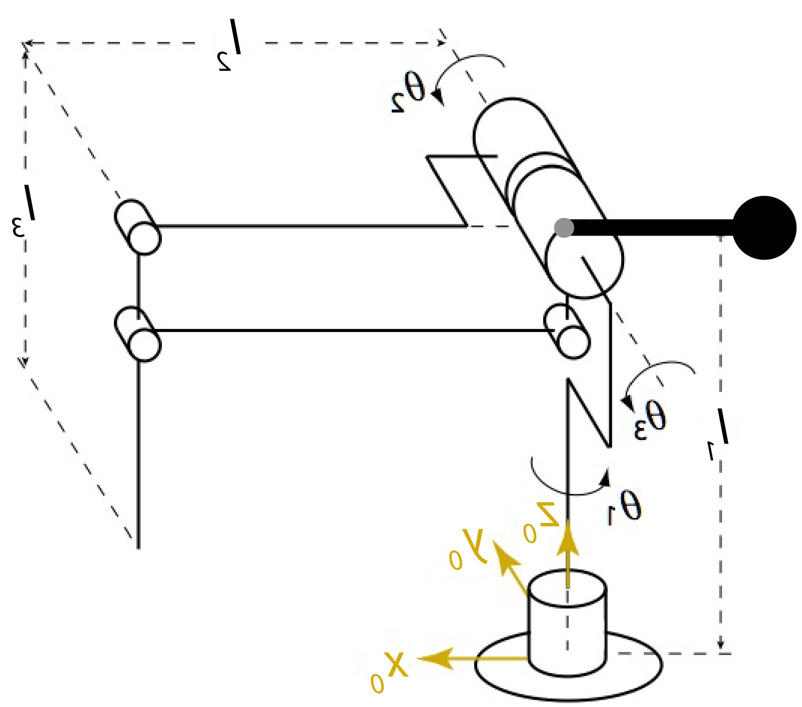
What about joint 2?

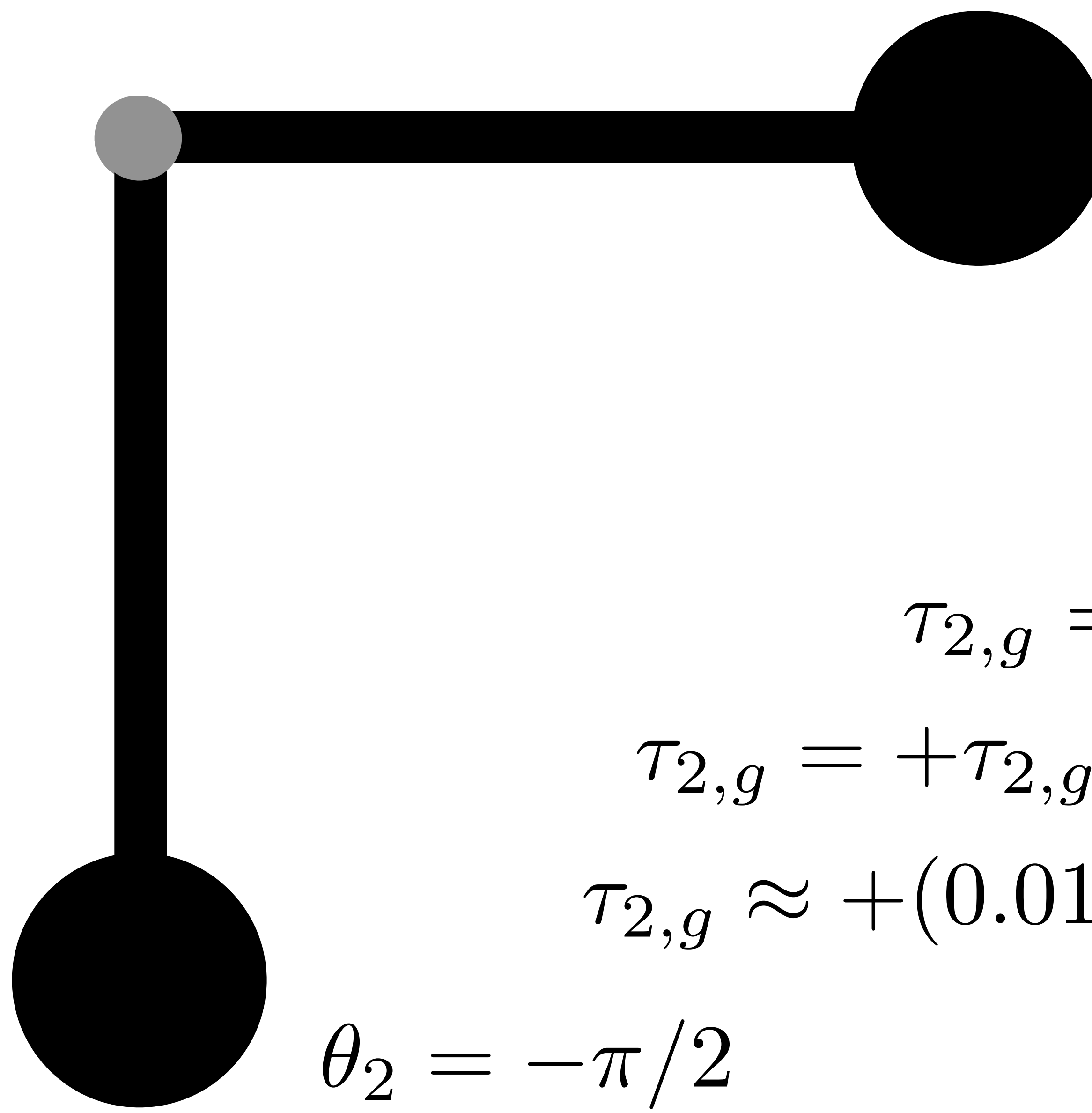


Calculate gravity compensation
grav = -0.034*sin(theta3-0.065);
the top of this script.
% Pause for one millisecond to keep
pause(.001);
Stop the Phantom.
phantomStop;
MATLAB R2012b
Figure 1

What form do you expect the gravity compensation for joint 2 to take?







$$\tau_{2,g} = ?$$

$$\tau_{2,g} = +\tau_{2,g,\max} \cos \theta_2$$

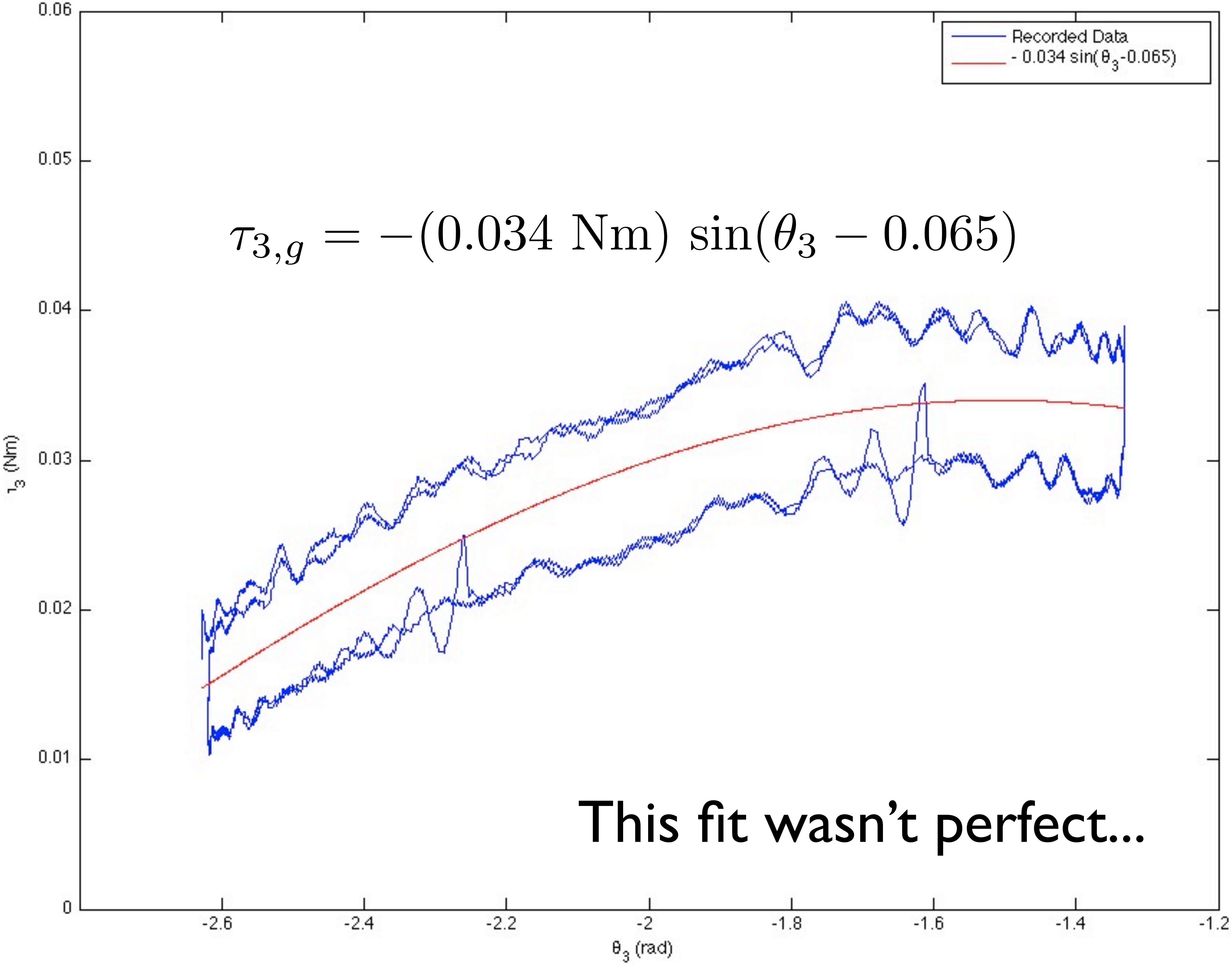
$$\tau_{2,g} \approx +(0.01 \text{ Nm}) \cos \theta_2$$

% Calculate gravity compensation to
tau2grav = 0.01*cos(theta2);
% Calculate gravity compensation to
tau3grav = -0.034*sin(theta3-0.065)
% Command the necessary joint
% If you're getting a lot of
% torques, you can turn that
% the top of this script.
phantomJointTorques(0, tau2grav,
% Pause for one millisecond to keep
pause(.001);
end
% Stop the Phantom.
phantomStop;
/home/pumastudent/...
Figure 1
EMERGENCY STOP
PHANTOM
PREMIUM

Questions ❓

What would happen if I re-do the joint 3 movement tests with gravity calibration on?

cal3newnewb slow refit

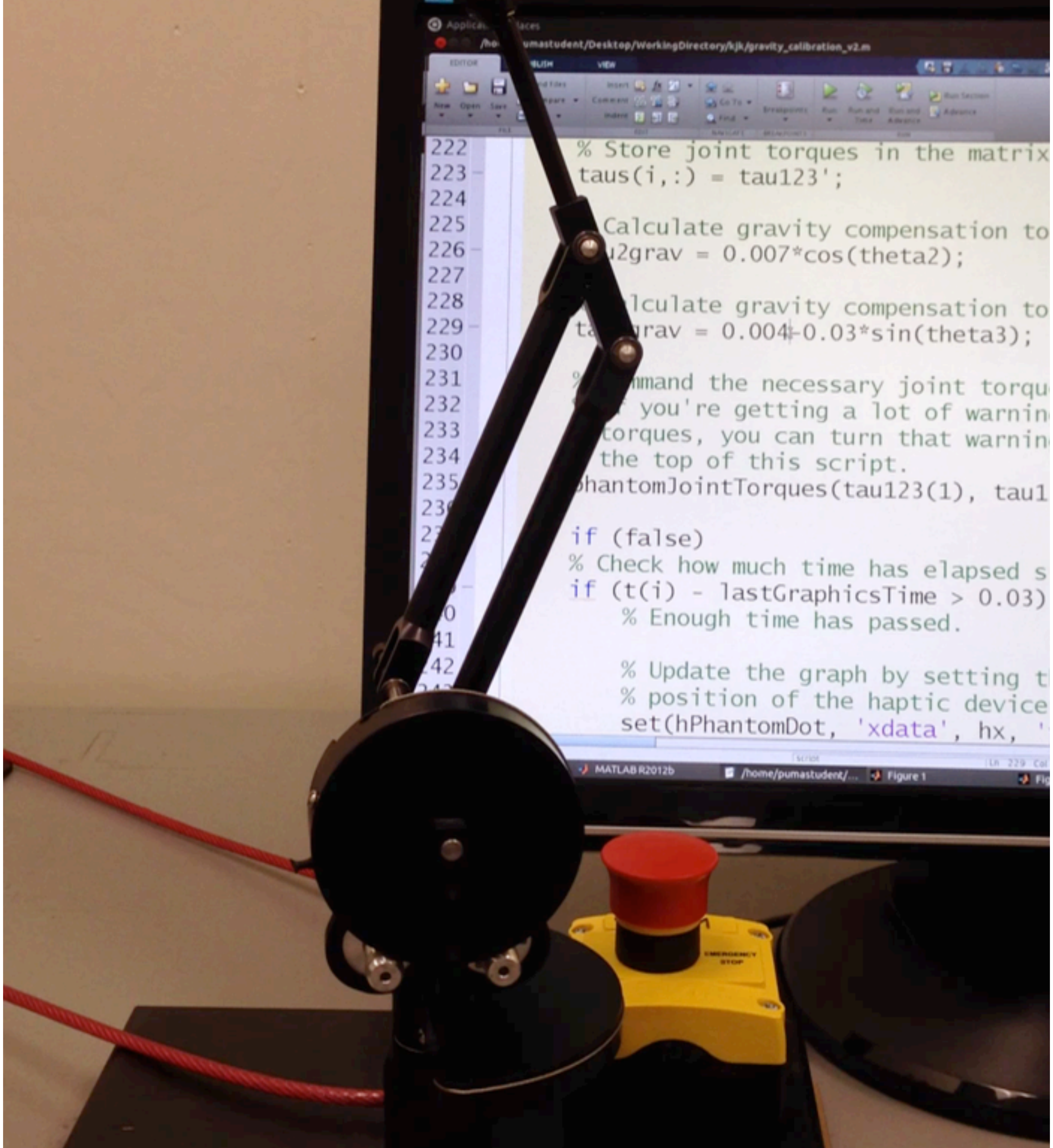


$$\tau_{3,g} = -(0.034\ \mathrm{Nm})\ \sin(\theta_3 - 0.065)$$

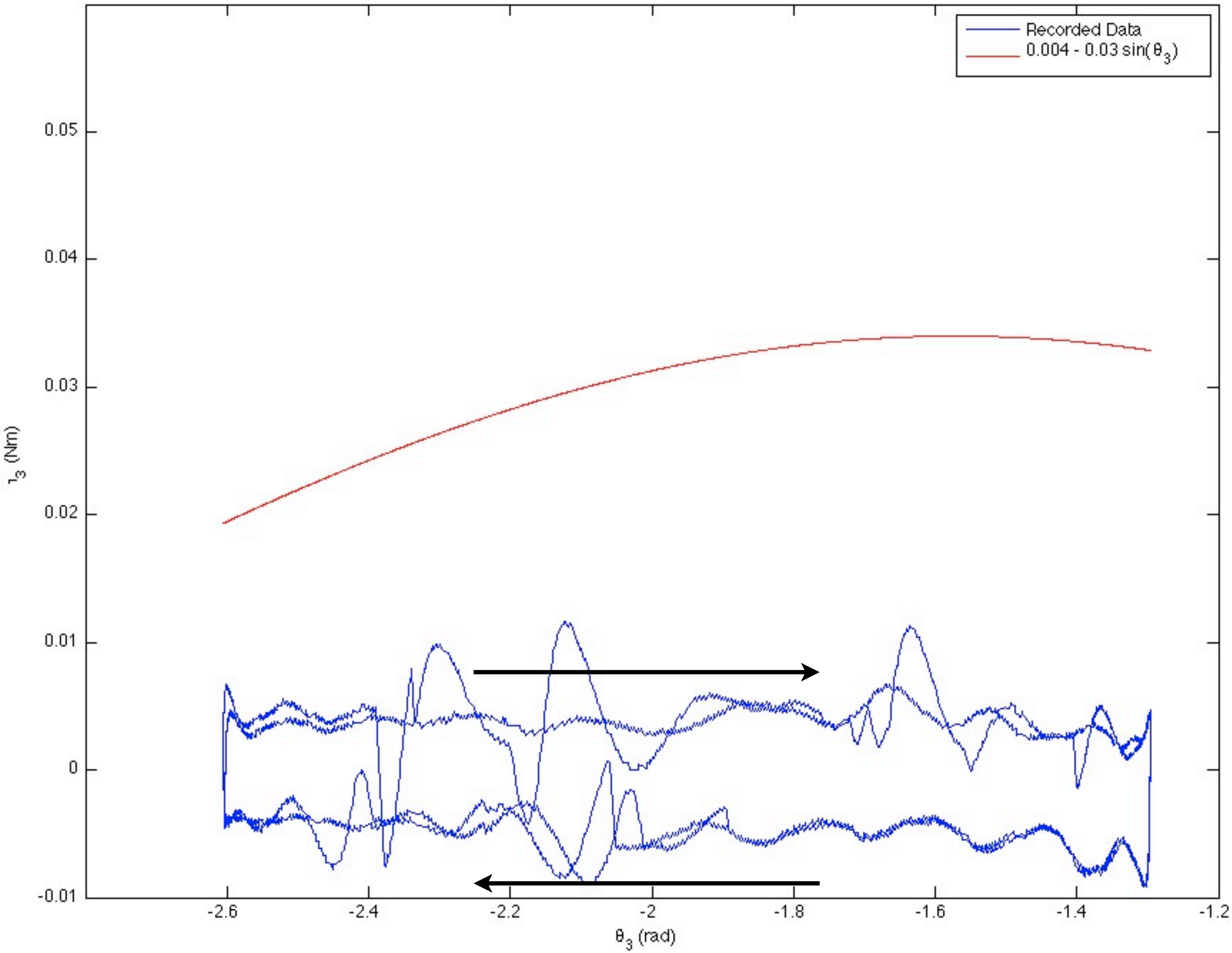
This fit wasn't perfect...

$$\tau_{3,g} = -(0.034\ \mathrm{Nm})\sin(\theta_3 - 0.065)$$

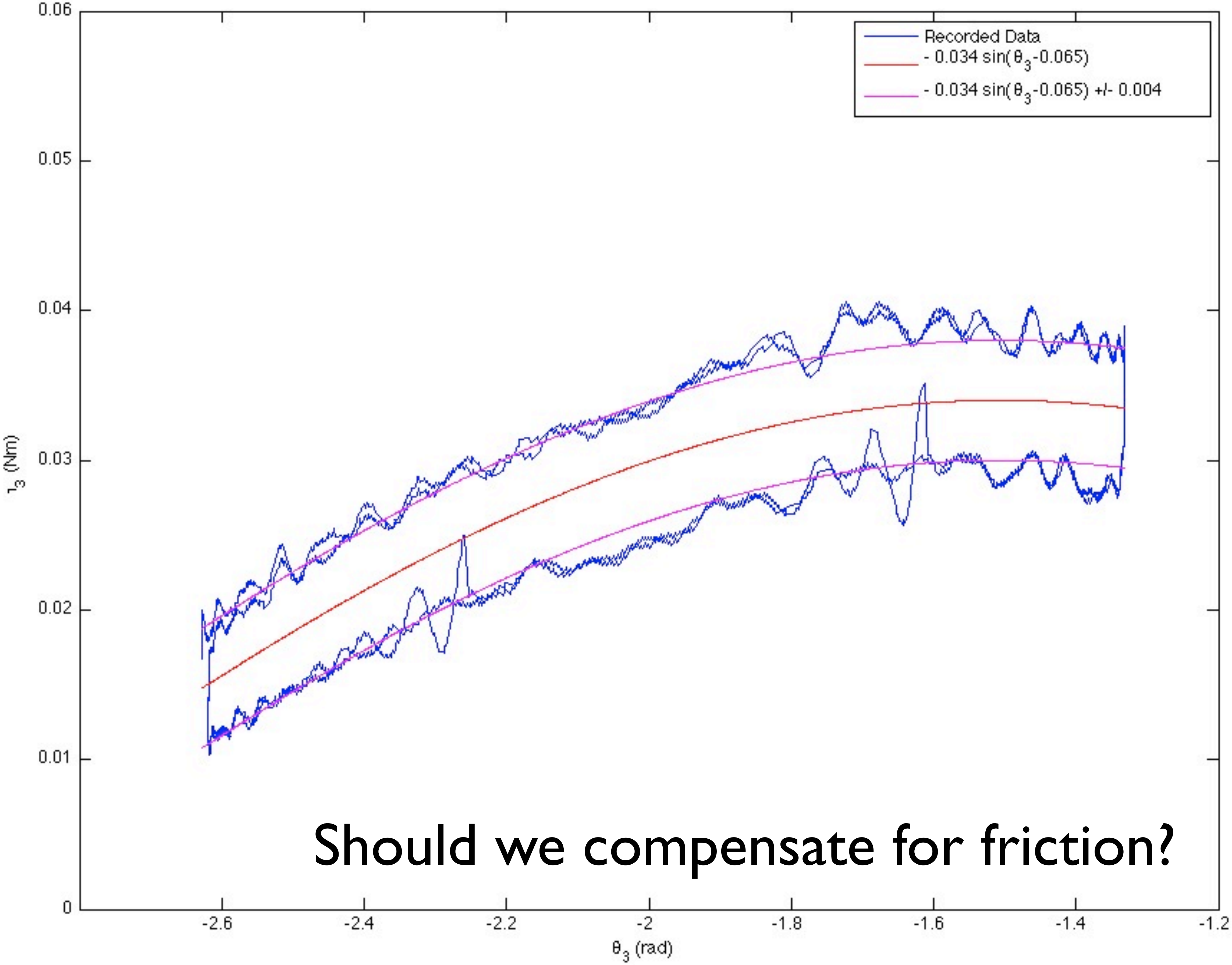
$$\tau_{3,g} = 0.004 - (0.03\ \mathrm{Nm})\sin(\theta_3)$$


% Store joint torques in the matrix
taus(i,:) = tau123';
Calculate gravity compensation to
u2grav = 0.007*cos(theta2);
lculate gravity compensation to
grav = 0.004-0.03*sin(theta3);
mmand the necessary joint torqu
you're getting a lot of warnin
torques, you can turn that warnin
the top of this script.
hantomJointTorques(tau123(1), tau1
if (false)
% Check how much time has elapsed s
if (t(i) - lastGraphicsTime > 0.03)
% Enough time has passed.
% Update the graph by setting t
% position of the haptic device
set(hPhantomDot, 'xdata', hx,
MATLAB R2012b
/home/pumastudent/...
Figure 1

cal3newnewb with updated gravity compensation



cal3newnewb slow refit with offsets



Should we compensate for friction?

mastudent/Desktop/WorkingDirectory/kjk/weightless.m
EDITOR
VIEW
% Calculate gravity compensation to
tau2grav = 0.007*cos(theta2);
Calculate gravity compensation to
3grav = 0.004-0.03*sin(theta3);
mmand the necessary joint torque
you're getting a lot of warning
ques, you can turn that warning
top of this script.
tomJointTorques(0, tau2grav, tau
Pause for one millisecond to keep
ause(.001);
Stop the Phantom.
hantomStop;
MATLAB R2012b
/home/pumastudent/...
EMERGENCY STOP

Questions ❓

Homework 6: Teleoperation

MEAM 520, University of Pennsylvania
Katherine J. Kuchenbecker, Ph.D.

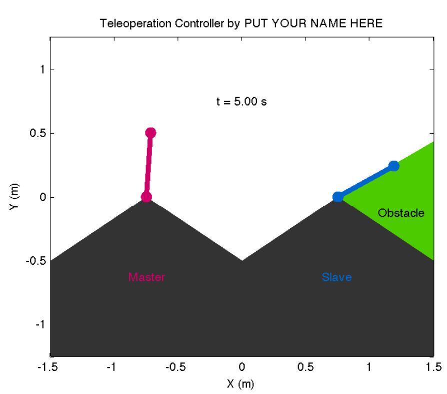
December 3, 2012

This assignment is due on **Friday, December 7**, by 5:00 p.m. If you don't finish by that time, you may turn it in with no penalty by 5:00 p.m. on Wednesday, December 12. After that deadline, no further assignments may be submitted. Because it is short, this assignment is worth 30 points (half the value of homework assignments 1 through 5).

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit should be your own work, not copied from a peer or a solution manual.

Teleoperation Controller *(30 points)*

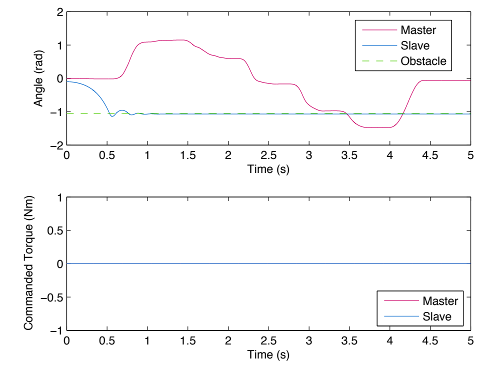
Your task is to write a good controller for a simple simulated teleoperation system. The image below shows a snapshot of the simulated teleoperator. It includes a one-degree-of-freedom master robot (left, in magenta) and an identical one-degree-of-freedom slave robot (right, in blue). Each device consists of a single revolute joint, much like the pair of Immersion Impulse Engine 2000 joysticks that Professor Kuchenbecker discussed in Lecture 18 (on November 20). Each robot's joint angle is measured in radians, with counterclockwise positive and straight up equal to zero.



The stationary bracket to which the robots are attached is shown in dark gray. The robots can move freely through this region because they are not in the same plane. There are no obstacles in the master's workspace, and the robots are too short to touch each other directly. There is one obstacle in the slave's workspace; shown in green, it begins at `obstacleAngle` and extends infinitely in the negative direction. You should move the obstacle around to test different environments; the controller that you write should work for any obstacle location, so it should not use the variable `obstacleAngle` in any way.

To simulate the presence of a human user holding onto the end of the master robot, the master moves through a pre-determined trajectory that you select. Six trajectories are provided (`masterMovement1.mat` ... `masterMovement6.mat`), and you can also write your own. The slave has pre-programmed dynamics that are hidden from your view inside the function `getSlaveTheta.p`. These dynamics include but are not

limited to inertia, gravity, friction, actuator saturation, and encoder quantization. When you first run the starter code, you will see that the slave just falls into the obstacle and stays there, while the master robot follows the default pre-determined trajectory. To help you understand what is happening in the simulation, the starter code animates the entire interaction and graphs the resulting angles and commanded torques over time, as shown in the sample graph below.



The simulated teleoperation system runs a servo loop at 1000 Hz, which you should not change. At each time step, it obtains the new position of the master (`masterTheta`) and the slave (`slaveTheta`) in radians. Your job is to specify the torque to command to the master (`masterTau`) and the slave (`slaveTau`) in newton-meters to yield good transparency (good tracking and good feel in free space, good feel in contact with the obstacle) and good stability (no extraneous ongoing oscillations). There should be no motion scaling or clutching between the two devices. The slave torque that you specify will directly affect the movement of the slave robot, while the master torque that you specify will merely be graphed. Following standard robotics convention, a positive torque moves the joint in the positive direction. It is expected that your controller will include gravity compensation, a proportional term, and a derivative term on both devices.

Download the starter code from this assignment's page on the class wiki, change the name of the provided script (`teleoperation_starter.m`) to include your PennKey, put your name where it says `'PUT YOUR NAME HERE'`, and make sure the starter code works correctly before starting to modify it. Near the top, you can change the movement of the master, the initial position of the slave, the angle of the obstacle, and the speed of the animation. When you're ready, put your controller code between the two lines of stars, modify whatever other simulation settings you want to elucidate the behavior of the system, and comment the final code you write. Follow the instructions below to submit your Matlab files.

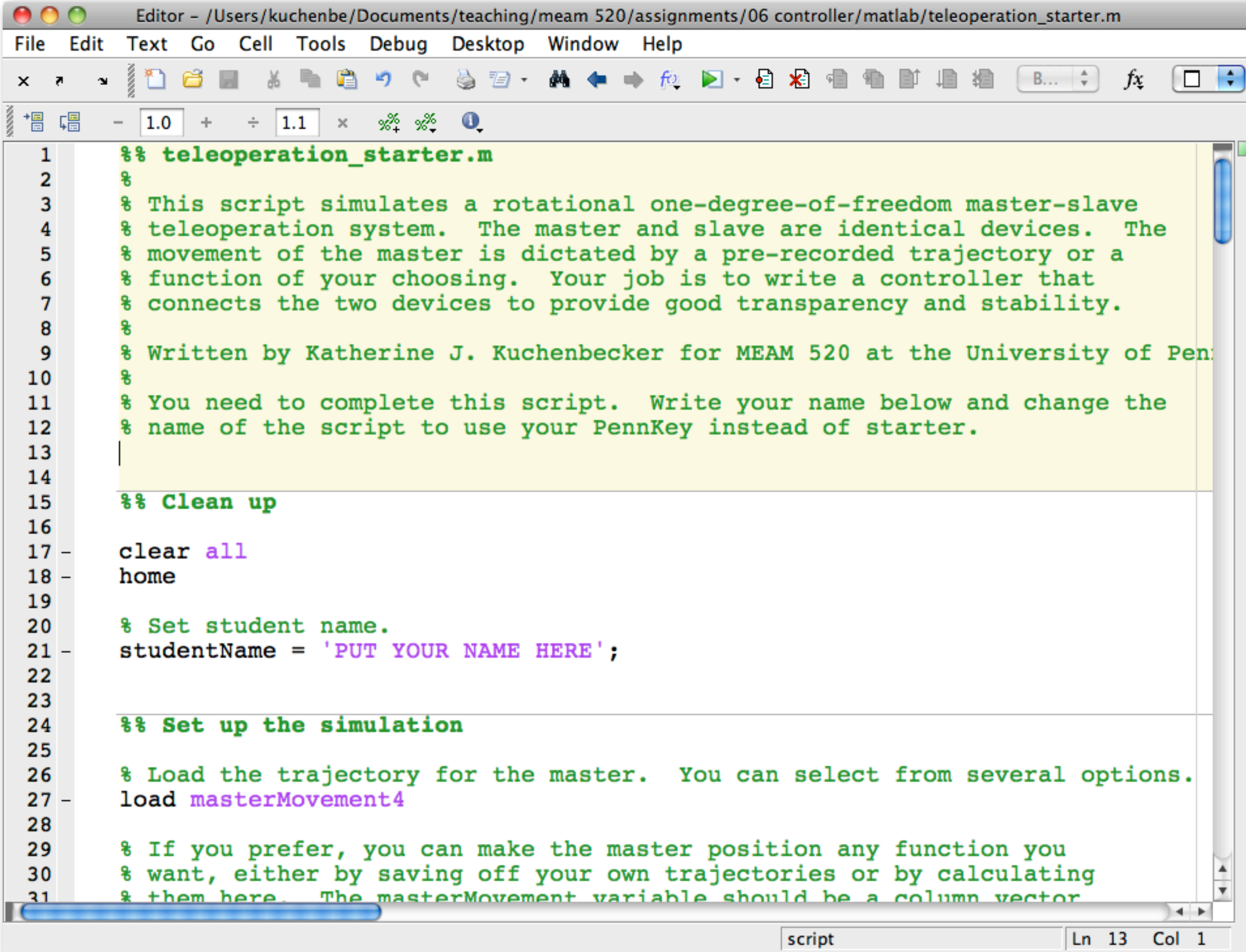
Submitting Your Code

Follow these instructions to submit your code:

1. Start an email to `meam520@seas.upenn.edu`
2. Make the subject *Homework 6: Your Name*, replacing *Your Name* with your name.
3. Attach your correctly named MATLAB script (`teleoperation_yourpennkey.m`) to the email, along with any other files that you created. You do not need to submit the provided `masterMovement.mat` or `getSlaveTheta.p` files. Please **do not zip your files together** before attaching them; just attach them as individual files.
4. Optionally include any comments you have about this assignment.
5. Send the email.

You are welcome to resubmit your code if you want to make corrections. To avoid confusion, please state in the new email that it is a resubmission, and include all of your MATLAB files, even if you have updated only some of them.

Due by 5:00 p.m. on Wednesday 12/12

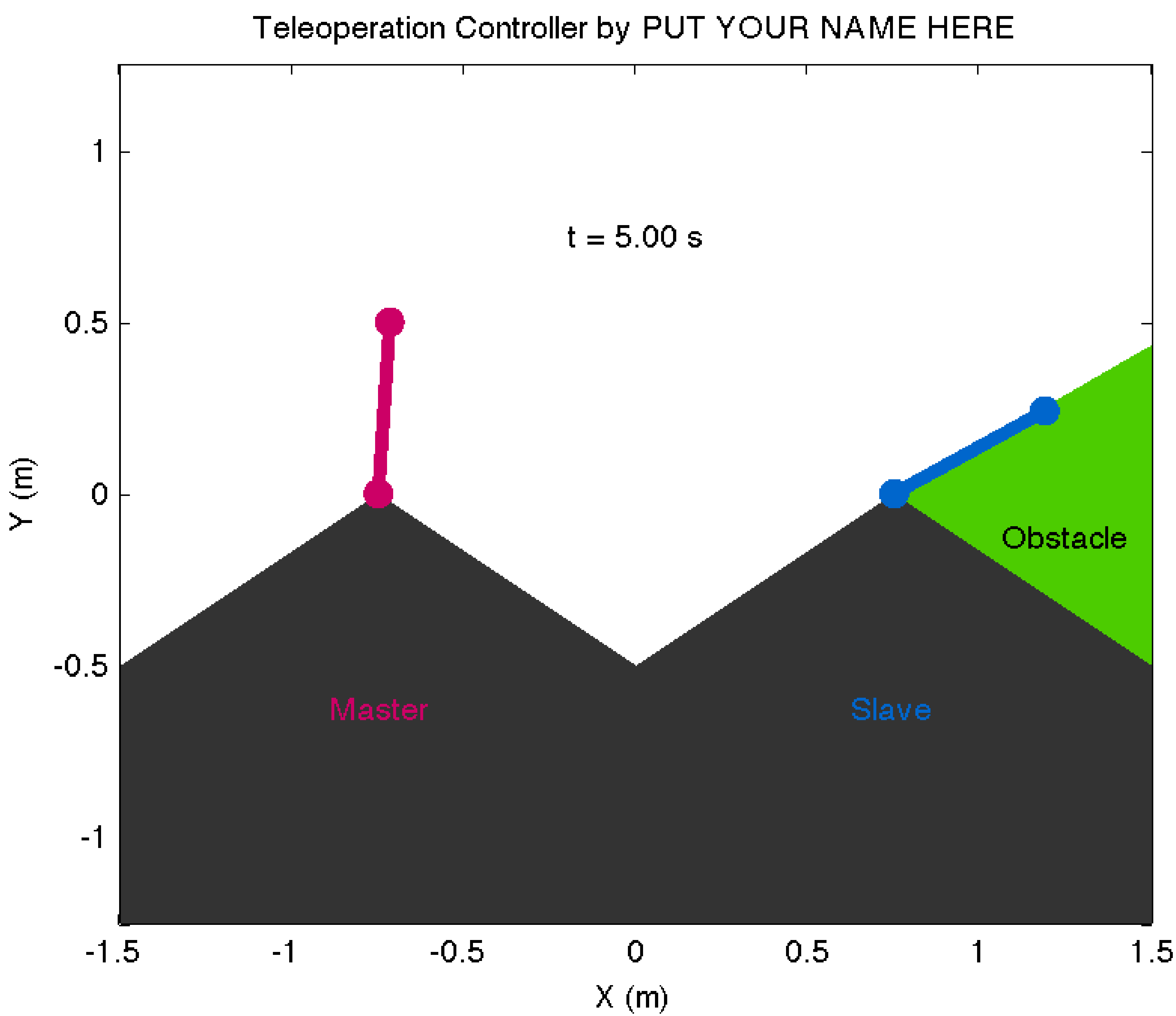

Editor - /Users/kuchenbe/Documents/teaching/meam 520/assignments/06 controller/matlab/teleoperation_starter.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
%% teleoperation_starter.m
%
% This script simulates a rotational one-degree-of-freedom master-slave
% teleoperation system. The master and slave are identical devices. The
% movement of the master is dictated by a pre-recorded trajectory or a
% function of your choosing. Your job is to write a controller that
% connects the two devices to provide good transparency and stability.
%
% Written by Katherine J. Kuchenbecker for MEAM 520 at the University of Pen
%
% You need to complete this script. Write your name below and change the
% name of the script to use your PennKey instead of starter.
%% Clean up
clear all
home
% Set student name.
studentName = 'PUT YOUR NAME HERE';
%% Set up the simulation
% Load the trajectory for the master. You can select from several options.
load masterMovement4
% If you prefer, you can make the master position any function you
% want, either by saving off your own trajectories or by calculating
script
Ln 13 Col 1

Editor – /Users/kuchenbe/Documents/teaching/meam 520/assignments/06 controller/matlab/teleoperation_starter.m

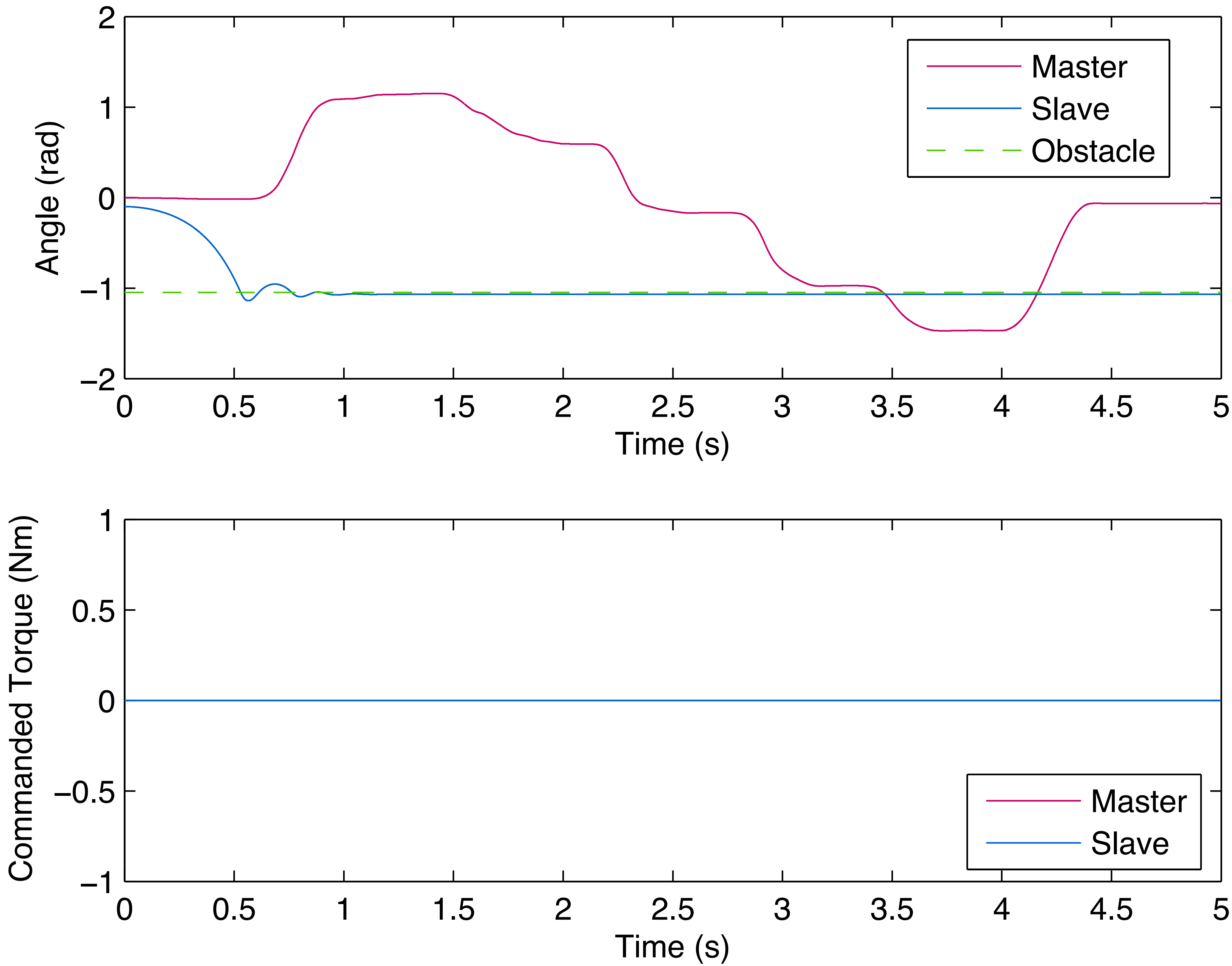
File Edit Text Go Cell Tools Debug Desktop Window Help

```matlab
%% Run the servo loop
for i = 1:nCycles

    % Calculate the present time.
    t = (i-1)*T;

    % Pull the next master position from the movement trajectory.
    masterTheta = masterMovement(i);

    % Measure the slave's current position (calculated from its dynamics).
    slaveTheta = getSlaveTheta;

    % All of your commands should be below this line of stars.
    % ******************************************************************

    % Calculate the torque to command on the master, in Nm, and store in
    % masterTau.  For now we set it to zero.
    masterTau = 0;

    % Calculate the torque to command on the slave, in Nm, and store in
    % slaveTau.  For now we set it to zero.
    slaveTau = 0;

    % ******************************************************************
    % All of your commands should be above this line of stars.

    % Store data in history vectors for plotting.
    timeHistory(i) = t;
    masterThetaHistory(i) = masterTheta;
    slaveThetaHistory(i) = slaveTheta;
    masterTauHistory(i) = masterTau;
```

script | Ln 181 Col 18



Teleoperation Controller by PUT YOUR NAME HERE
t = 5.00 s
Obstacle
Master
Slave
X (m)
Y (m)
-1.5
-1
-0.5
0
0.5
1
1.5



Angle (rad)
Time (s)
Master
Slave
Obstacle
Commanded Torque (Nm)
Time (s)
Master
Slave

Questions ?

Sign Up for da Vinci Robot Demos – First One Tonight at 6:30pm ...

Delete Junk Reply Reply All Forward Print To Do

From: Katherine Kuchenbecker
Subject: **Sign Up for da Vinci Robot Demos - First One Tonight at 6:30pm**
Date: December 4, 2012 11:25:10 AM EST
To: MEAM520-001-12C@LISTS.UPENN.EDU
Reply-To: Katherine Kuchenbecker

Dear MEAM 520,

I'd like to give each of you the chance to try out the Intuitive da Vinci surgery robot that my lab uses in our research. It is a very well engineered robotic teleoperation system, and many of the topics we have discussed this semester directly pertain to how it works.

Each demo will take about one hour and include up to ten people from our class. Each person will have the chance to try the robot for a few minutes, and we'll talk about how it works.

I'm offering three times this week and two times next week. The first time is TONIGHT at 6:30pm. To sign up for a time, please enter your name on this Doodle poll:
http://doodle.com/emqgt6v332p8bran

Meet on the fourth floor of Levine near the elevators, outside the main GRASP Lab. If you would like to try the da Vinci but cannot make any of the available slots, please email me directly.

Thanks!
kjk


The Ben Franklin
Racing Team

Next time:
Kinematics of Mobile Robots
Overview of Final Exam