MEAM 520

More Motion Control

Katherine J. Kuchenbecker, Ph.D.

General Robotics, Automation, Sensing, and Perception Lab (GRASP)

MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 25: December 3, 2013

Homework 8: Input/Output Calculations for the Phantom Robot

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

November 12, 2013

This assignment is due on **Tuesday, November 19, by midnight (11:59:59 p.m.)**. You should aim to turn the paper part in during class that day. If you don't finish before class, you can turn the paper part in to the bin outside Professor Kuchenbecker's office, Towne 224. Your code should be submitted via email according to the instructions at the end of this document. Late submissions will be accepted until Thursday, November 21, by midnight (11:59:59 p.m.), but they will be penalized by 10% for each partial or full day late, up to 20%. After the late deadline, no further assignments may be submitted.

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 write down must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. If you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

You may do this assignment either individually or with a partner. If you do this homework with a partner, you may work with anyone in the class. If you are in a pair, you should work closely with your partner throughout this assignment, following the paradigm of pair programming. You will turn in one paper assignment and one set of MATLAB files for which you are both jointly responsible, and you will both receive the same grade. Please follow the pair programming guidelines that have been shared before, and name your files with both of your PennKeys separated by an underscore character. Do not split the assignment in half; both of you should understand all steps of this assignment.

SensAble Phantom Premium 1.0

This entire assignment is focused on a particular robot – the SensAble Phantom Premium 1.0. As shown in the photo below, the Phantom is an impedance-type haptic interface with three actuated rotational joints. Designed to be lightweight, stiff, smooth, and easily backdrivable, this type of robotic device enables a human user to interact with a virtual environment or control the movement of a remote robot through the movement of their fingertip while simultaneously feeling force feedback.

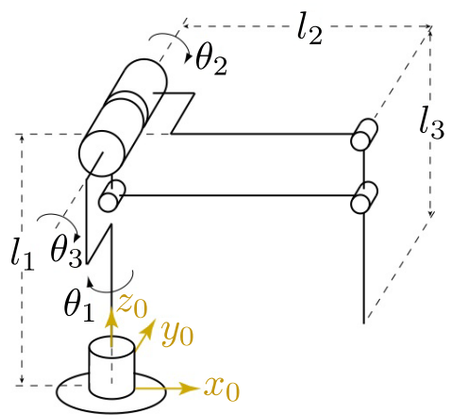
Homework 8 is graded and ready for pickup.

Common mistakes:

- Converting forces to joint torques.
- Forgetting gear rations when converting joint torques to voltages.
- Sign errors and other mistakes in motor dynamics.

Solutions to Homework 8
Input/Output Calculations for the Phantom Robot

MEAM 520
Introduction to Robotics
University of Pennsylvania
Professor Kuchenbecker
Fall 2013



Homework 8 solutions are on reserve in the engineering library.

Tentative MEAM 520 Calendar

Tuesday 12/3 – More Motion Control
Thursday 12/5 – Haptics and Teleoperation
Introduce Homework 9
Thursday 12/5 – Project 2 Part 2 (Sim Painting) due

Tuesday 12/10 – Mobile Robots
Tuesday 12/10 – Homework 9 (Haptics) due, with no penalty up to 12/14

Ongoing – Record Light Paintings on PUMA

Wednesday 12/18 – Final Exam noon to 2 p.m.

The official course evaluations are now available.

You will need to respond or opt out in order to see your final grade in this class.

Penn, MEAM, and I value your feedback, so please respond when you have time.

It is fine to wait until after the final exam. I will send a reminder then.

I know you have worked hard on the PUMA IK.

Project 2: Part 1
Inverse Kinematics for the PUMA 260

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

November 22, 2013

This project component is due on **Wednesday, November 27, by midnight (11:59:59 p.m.)**. Your code should be submitted via email according to the instructions at the end of this document. Late submissions will be accepted after this deadline, but they will be penalized by 10% for each partial or full day late. Because Thanksgiving is a holiday, it will not count against the lateness tally; assignments submitted either Thursday, November 28, or Friday, November 29, will be penalized 10%.

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 write down must be your team's own work, not copied from any other student, team, or source. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. If you get stuck, post a question on Piazza or go to office hours!

Team Selection

You should do this project component with your Project 2 team. If you don't yet have a team number, send an email to `meam520@seas.upenn.edu` to declare your team choice. The subject should be "Project 2 Team Selection." List the three full names of your team members in the body of the email. Only one person needs to submit the team. We will respond with a team number and post it in the list on Piazza.

If you are looking for a teammate, consider using the "Search for Teammates!" tool on Piazza. As another alternative, we are happy to assign you to a random partner or two. To ask us to do this, send an email to `meam520@seas.upenn.edu` with the subject "Project 2 Partner Request" with your full name (or two full names if in a pair) in the body of the email.

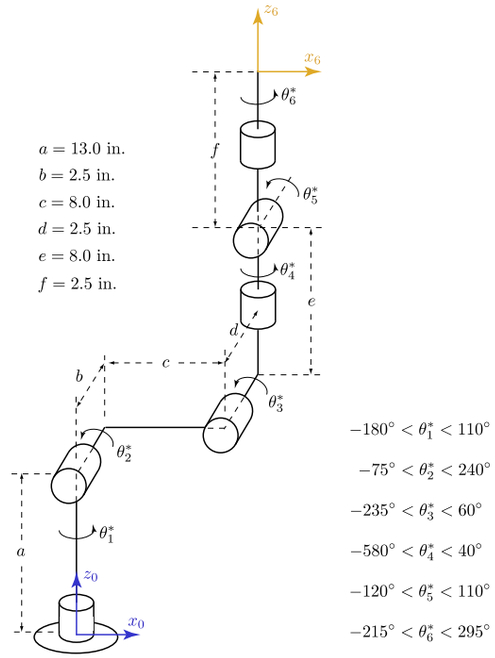
You should work closely with your partners throughout this assignment, following the paradigm of pair programming (but with three people). You will turn in one set of MATLAB files for which you are all jointly responsible, and you will receive the same baseline grade. Please follow the pair programming guidelines that have been shared before. After the project is over, we will administer a simple survey that asks each student to rate how well each teammate (including him/herself) contributed to the project; when teammates agree that the workload was not shared evenly, individual grades will be adjusted accordingly.

Light Painting

Project 2 is PUMA Light Painting. Each team of three students will write MATLAB code to make our PUMA 260 robot draw something interesting in the air with a colored light, which we will capture by taking a long-exposure photograph. Drawing precise, arbitrary shapes with a robot requires you to solve the robot's full inverse kinematics, so that is the first component of this project.

System Description

Like Homework 5, Project 1, and Homework 7, this assignment centers on the PUMA 260, an articulated (RRR) robot with lateral offsets plus a spherical wrist (RRR). The image below on the left shows a photo of the robot. The schematic below on the right shows the zero configuration we have designated for the PUMA in this class. All of the joints are shown at $\theta_i = 0$. The joint angle arrows show the positive direction for each revolute joint (θ_1 to θ_6). The diagram also defines the location and orientation of the robot's base frame (frame 0) and the end-effector frame (frame 6). The text on the sides of the diagram give the measurements for the constant dimensions (a to f), all in inches, and the minimum and maximum angles for each of the PUMA's six revolute joints.



$a = 13.0$ in.
$b = 2.5$ in.
$c = 8.0$ in.
$d = 2.5$ in.
$e = 8.0$ in.
$f = 2.5$ in.

$-180° < \theta_1^* < 110°$
$-75° < \theta_2^* < 240°$
$-235° < \theta_3^* < 60°$
$-580° < \theta_4^* < 40°$
$-120° < \theta_5^* < 110°$
$-215° < \theta_6^* < 295°$

Task Overview

Your task is to write a MATLAB function that solves the full inverse kinematics for our PUMA robot. This function must take in the desired position and orientation of the end-effector frame and return sets of joint angles that will put the robot's end-effector in the desired pose. Before you begin programming, you will need to spend considerable time figuring out how to approach and solve each step of this problem. Here are some specific suggestions of things to do at the start:

- Remind yourself of the principle of **kinematic decoupling** and plan how to apply it to the PUMA.
- Turn the **SCARA inverse kinematics v2 code** that was posted on Piazza (under Lecture Resources in a zip file of code for Lecture 23) into the fully functional v3, which Professor Kuchenbecker showed in class but did not post. Doing the full inverse kinematics of the PUMA requires the same sequence of steps, modified slightly, as explained in the next point.

Editor - /Users/kuchenbe/Documents/teaching/meam 520/projects/2 puma paint/part 1 puma_ik/testing/test_puma_ik.m

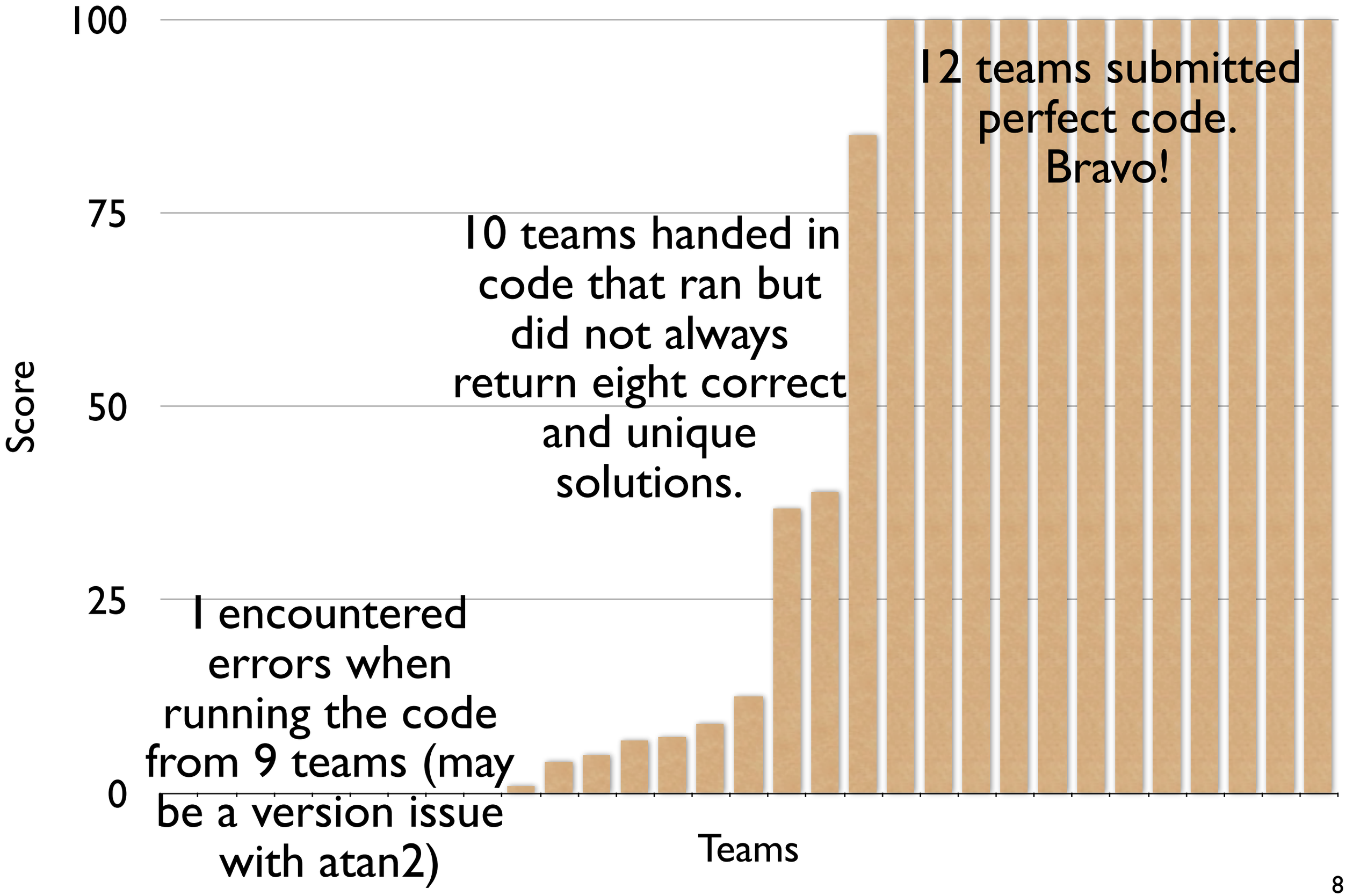
EDITOR | PUBLISH | VIEW

test_puma_ik.m

```
1   %% test_puma_ik.m
2   % Tests the full inverse kinematics for the PUMA 260 for one team.
3   % Written by Katherine J. Kuchenbecker for MEAM 520 at the University of
4   % Pennsylvania in Fall of 2013.
5
6
7   %% Clear
8
9   % Clear the workspace and the console to make it easier to find outputs and errors.
10  clear all
11  home
12
13
14  %% Set IK function name
15
16  % Define the name of your the inverse kinematics function you want to test.
17  % The @ symbol in front of the function name creates a function handle, so
18  % that we can call the function indirectly.  We do this so that you can
19  % quickly change the IK function that you want to test, without having to
20  % change it in many places down below in the code.  This feature will also
21  % let the grader test your test code on IK that we know to work well or
22  % poorly, to see what score your test function assigns it.
23  puma_ik = @team200_puma_ik;
24  disp(['Testing function ' func2str(puma_ik) '.m'])
25
26
27  %% Set up empty matrices
28
29  % Set the number of tests we want to run.
30  nTests = 2000;
31
32  % Initialize history vectors with all zeros.
33  ox_history = zeros(nTests,1);
34  oy_history = zeros(nTests,1);
35  oz_history = zeros(nTests,1);
36  phi_history = zeros(nTests,1);
37  theta_history = zeros(nTests,1);
38  psi_history = zeros(nTests,1);
39
```

script | Ln 67 Col 31

PUMA Inverse Kinematics Test Results



If you're still improving your IK, consider testing it with this tool.

MEAM 520 | Class Profile | Piazza

https://piazza.com/upenn/fall2013/meam520/resources

Google

Homework

Homework	Due Date	Actions
puma_ik_testing.zip	Dec 5, 2013	Edit · Post a note · Delete
puma_paint.zip	Dec 5, 2013	Edit · Post a note · Delete
project2.2.pdf	Dec 5, 2013	Edit · Post a note · Delete
puma_ik.zip	Nov 27, 2013	Edit · Post a note · Delete
project2.1.pdf	Nov 27, 2013	Edit · Post a note · Delete
phantom_kinematics.mov	Nov 19, 2013	Edit · Post a note · Delete
phantom_files.zip	Nov 19, 2013	Edit · Post a note · Delete
hw08.pdf	Nov 19, 2013	Edit · Post a note · Delete
dh_kuchenbe.m	Nov 3, 2013	Edit · Post a note · Delete
plot_puma_starter.m	Nov 3, 2013	Edit · Post a note · Delete
analyze_puma_starter.m	Nov 3, 2013	Edit · Post a note · Delete

Show all resources

Add Links · Add Files

Let me know if you find any problems with the testing script.

Project 2: Part 2
Light Painting with the PUMA 260

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

November 26, 2013

This project component is due on **Thursday, December 5, by midnight (11:59:59 p.m.)**. Your code should be submitted via email according to the instructions at the end of this document. Late submissions will be accepted after this deadline, but they will be penalized by 10% for each partial or full day late, up to a maximum of 30% if submitted by midnight on Sunday, December 8. After this late deadline, no further submissions will be accepted.

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 write down must be your team's own work, not copied from any other student, team, or source. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. If you get stuck, post a question on Piazza or go to office hours!

Teamwork

You should work closely with your Project 2 teammates throughout this assignment. You will turn in one set of MATLAB files for which you are all jointly responsible, and you will receive the same baseline grade. Please follow the pair programming guidelines that have been shared before. After the project is over, we will administer a simple survey that asks each student to rate how well each teammate (including him/herself) contributed to the project; when teammates agree that the workload was not shared evenly, individual grades will be adjusted accordingly.

Light Painting

Project 2 is PUMA Light Painting. Each team of three students will write MATLAB code to make our PUMA 260 robot draw something interesting in the air with a colored light, which we will capture by taking a long-exposure photograph. Drawing precise, arbitrary shapes with a robot requires you to solve the robot's full inverse kinematics (IK), so that was the first component of this project. This is the second part – using your inverse kinematics to create the actual light painting.

Task Overview

Your task is to write two MATLAB functions that can be combined with your inverse kinematics function to enable our PUMA robot to create an original light painting. The first function defines the painting; it needs to be able to calculate the position, orientation, and color that the robot's light-emitting diode (LED) should have at any given time during the performance of the painting. At each time step, this position and orientation will be passed into your IK code, which will return sets of joint angles that will put the robot's LED in the desired pose. These solutions will then be passed into the second function, which will select the best solution from the options found, based both on the characteristics of the PUMA 260 robot and on the robot's current configuration. The robot is commanded to move to these selected joint angles, and the cycle

Part 2 is due this Thursday by midnight.

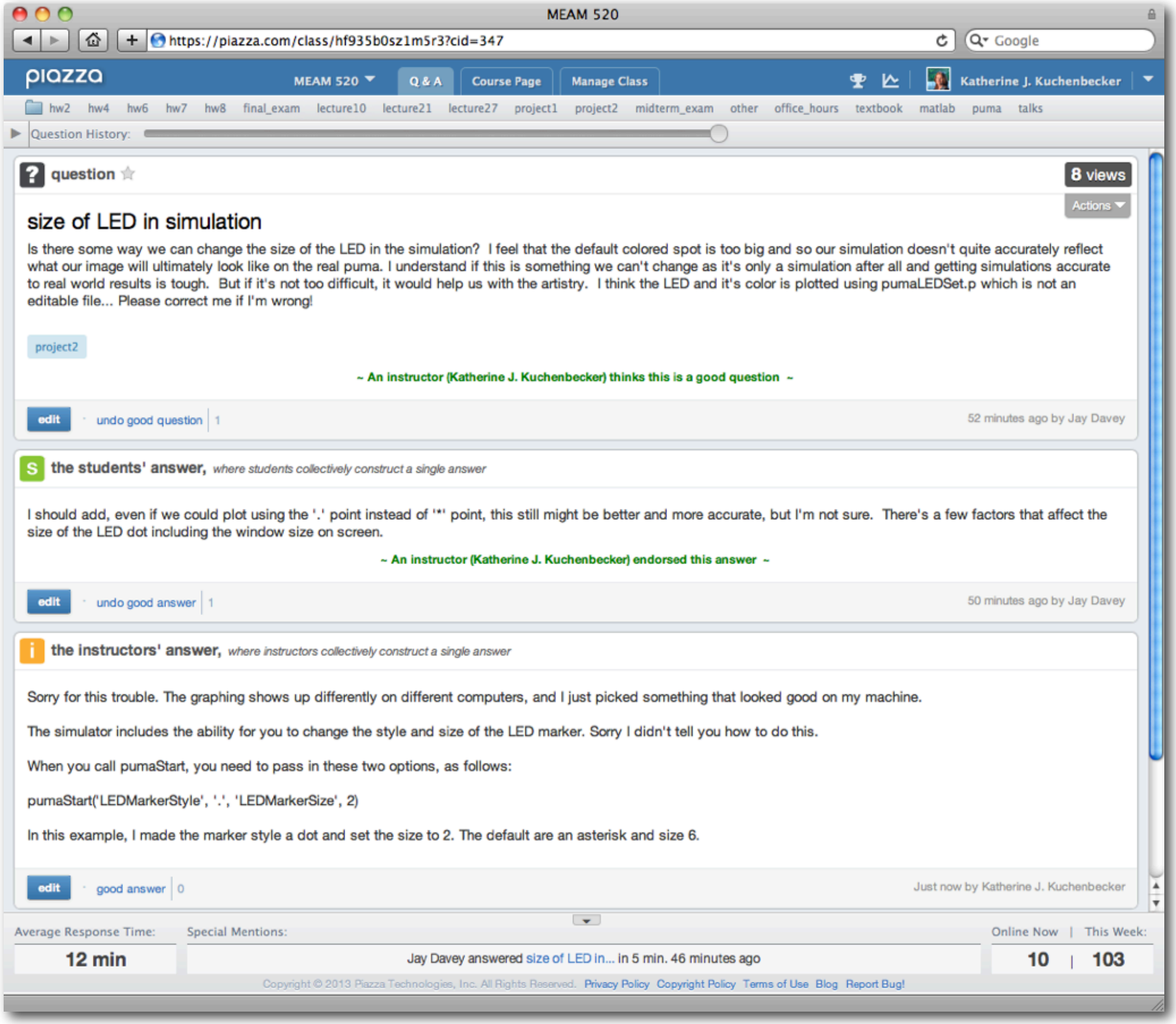
Late deadline is Sunday by midnight, with 10 % penalty per day.

Include a new version of your IK.

Need help?

I have office hours today from 1:30 to 3:00 p.m.
And on Thursday from 2:30 to 4:00
(both extended)

If anyone who already solved the IK is interested
in helping other classmates figure this out,
please let me know. A good way to earn extra credit
if you have extra time this week.


MEAM 520
https://piazza.com/class/hf935b0sz1m5r3?cid=347
Google
piazza
MEAM 520
Q & A
Course Page
Manage Class
Katherine J. Kuchenbecker
hw2 hw4 hw6 hw7 hw8 final_exam lecture10 lecture21 lecture27 project1 project2 midterm_exam other office_hours textbook matlab puma talks
Question History:
question
8 views
Actions
size of LED in simulation
Is there some way we can change the size of the LED in the simulation? I feel that the default colored spot is too big and so our simulation doesn't quite accurately reflect what our image will ultimately look like on the real puma. I understand if this is something we can't change as it's only a simulation after all and getting simulations accurate to real world results is tough. But if it's not too difficult, it would help us with the artistry. I think the LED and it's color is plotted using pumaLEDSet.p which is not an editable file... Please correct me if I'm wrong!
project2
~ An instructor (Katherine J. Kuchenbecker) thinks this is a good question ~
edit
undo good question 1
52 minutes ago by Jay Davey
the students' answer, where students collectively construct a single answer
I should add, even if we could plot using the '.' point instead of '*' point, this still might be better and more accurate, but I'm not sure. There's a few factors that affect the size of the LED dot including the window size on screen.
~ An instructor (Katherine J. Kuchenbecker) endorsed this answer ~
edit
undo good answer 1
50 minutes ago by Jay Davey
the instructors' answer, where instructors collectively construct a single answer
Sorry for this trouble. The graphing shows up differently on different computers, and I just picked something that looked good on my machine.
The simulator includes the ability for you to change the style and size of the LED marker. Sorry I didn't tell you how to do this.
When you call pumaStart, you need to pass in these two options, as follows:
pumaStart('LEDMarkerStyle', '.', 'LEDMarkerSize', 2)
In this example, I made the marker style a dot and set the size to 2. The default are an asterisk and size 6.
edit
good answer 0
Just now by Katherine J. Kuchenbecker
Average Response Time:
12 min
Special Mentions:
Jay Davey answered size of LED in... in 5 min. 46 minutes ago
Online Now | This Week:
10 | 103
Copyright © 2013 Piazza Technologies, Inc. All Rights Reserved. Privacy Policy Copyright Policy Terms of Use Blog Report Bug!

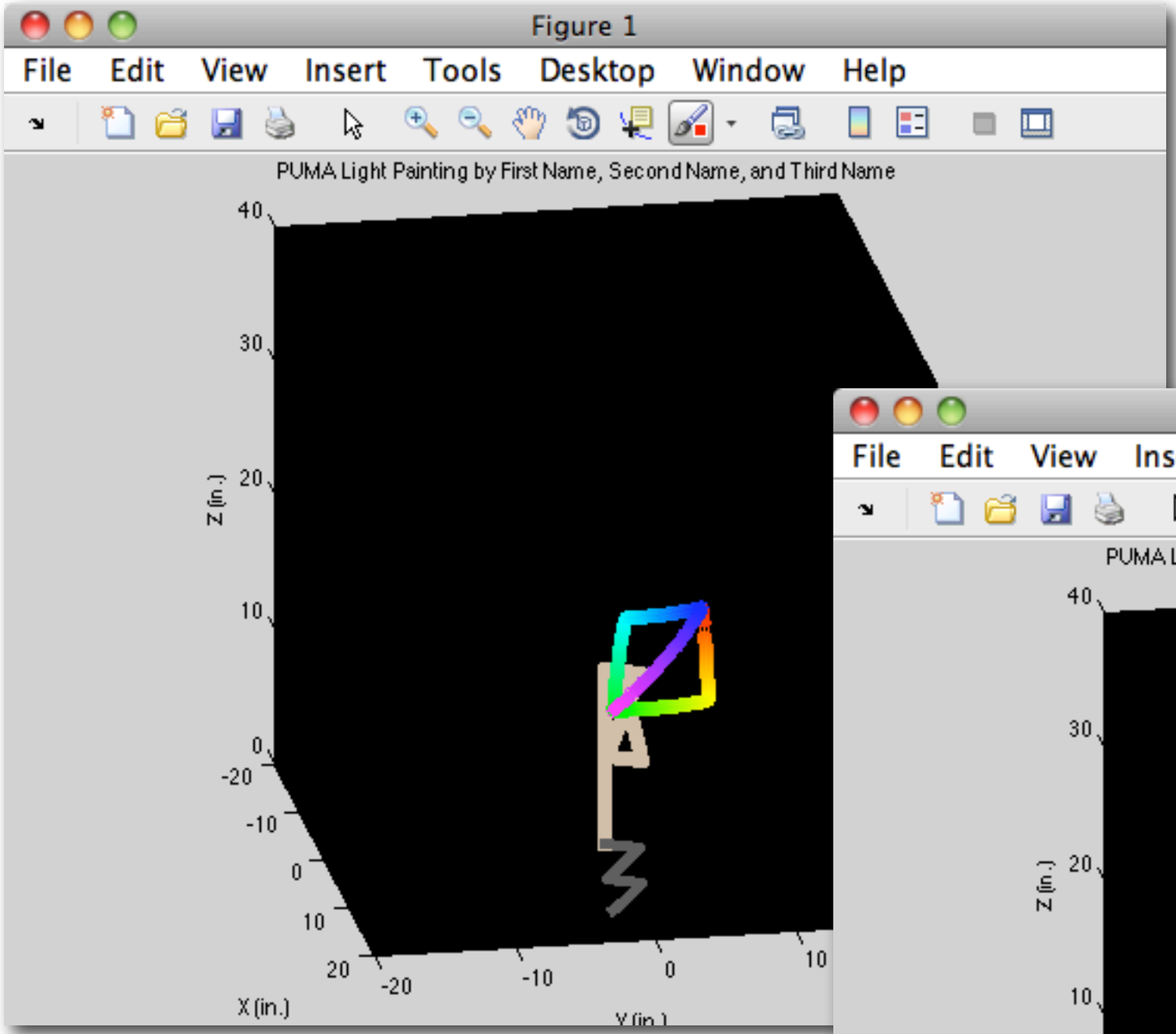


Figure 1
File Edit View Insert Tools Desktop Window Help
PUMA Light Painting by First Name, Second Name, and Third Name
Z (in.)
X (in.)
Y (in.)

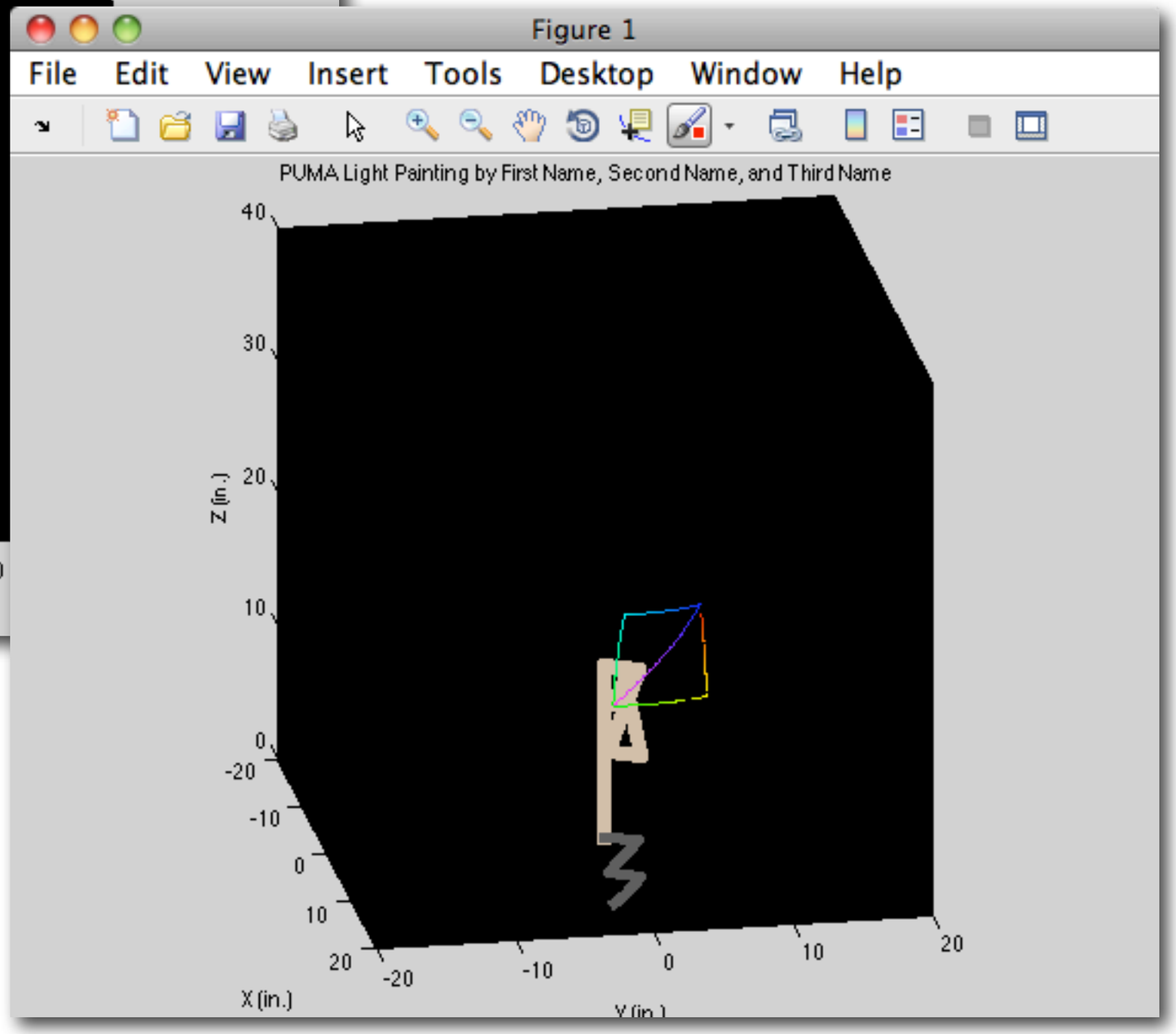
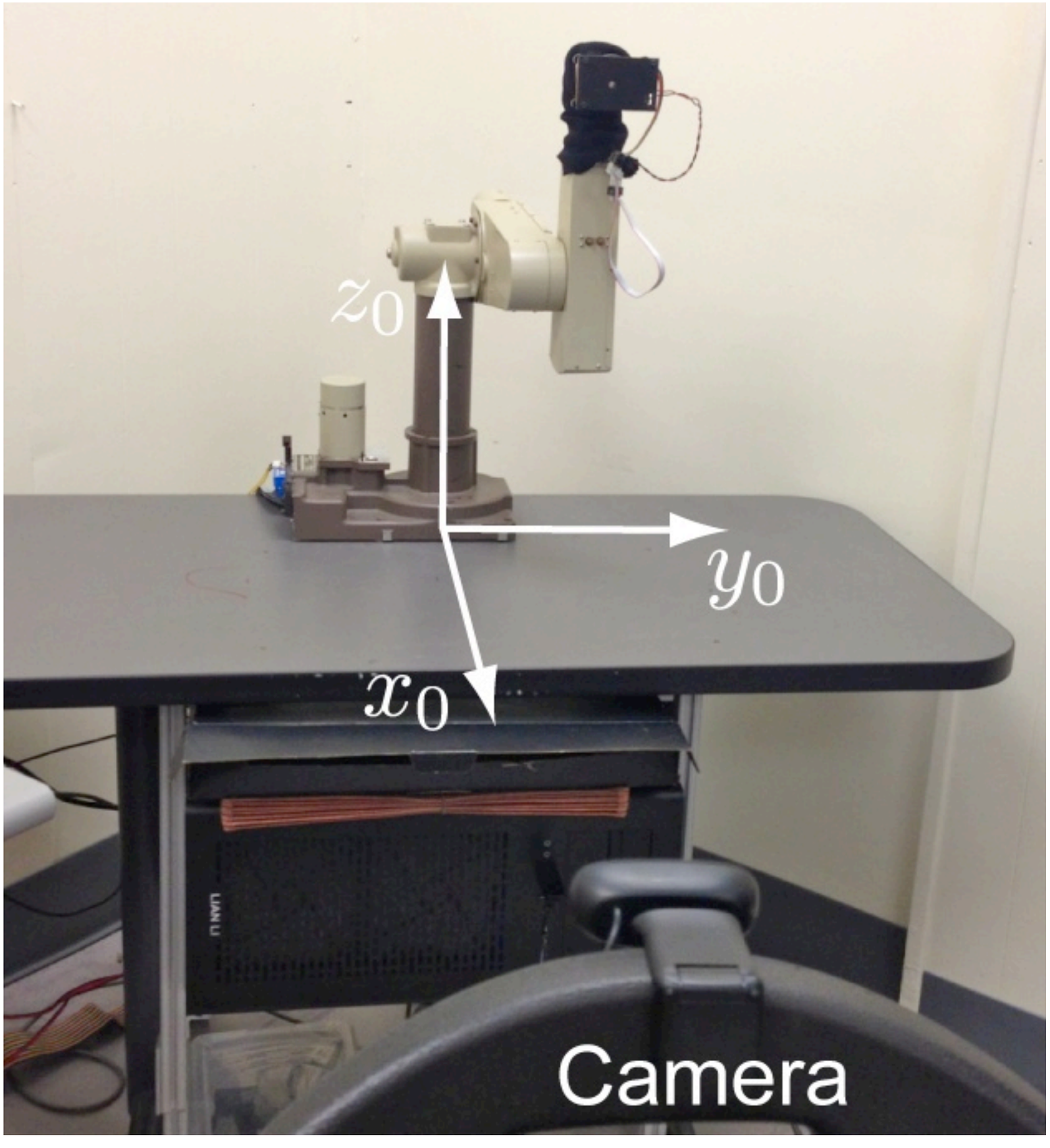


Figure 1
File Edit View Insert Tools Desktop Window Help
PUMA Light Painting by First Name, Second Name, and Third Name
Z (in.)
X (in.)
Y (in.)



The colored LED is now attached to the PUMA's end-effector.

The TA's fixed the controller on joint 2 so the robot will move more smoothly. Yay!

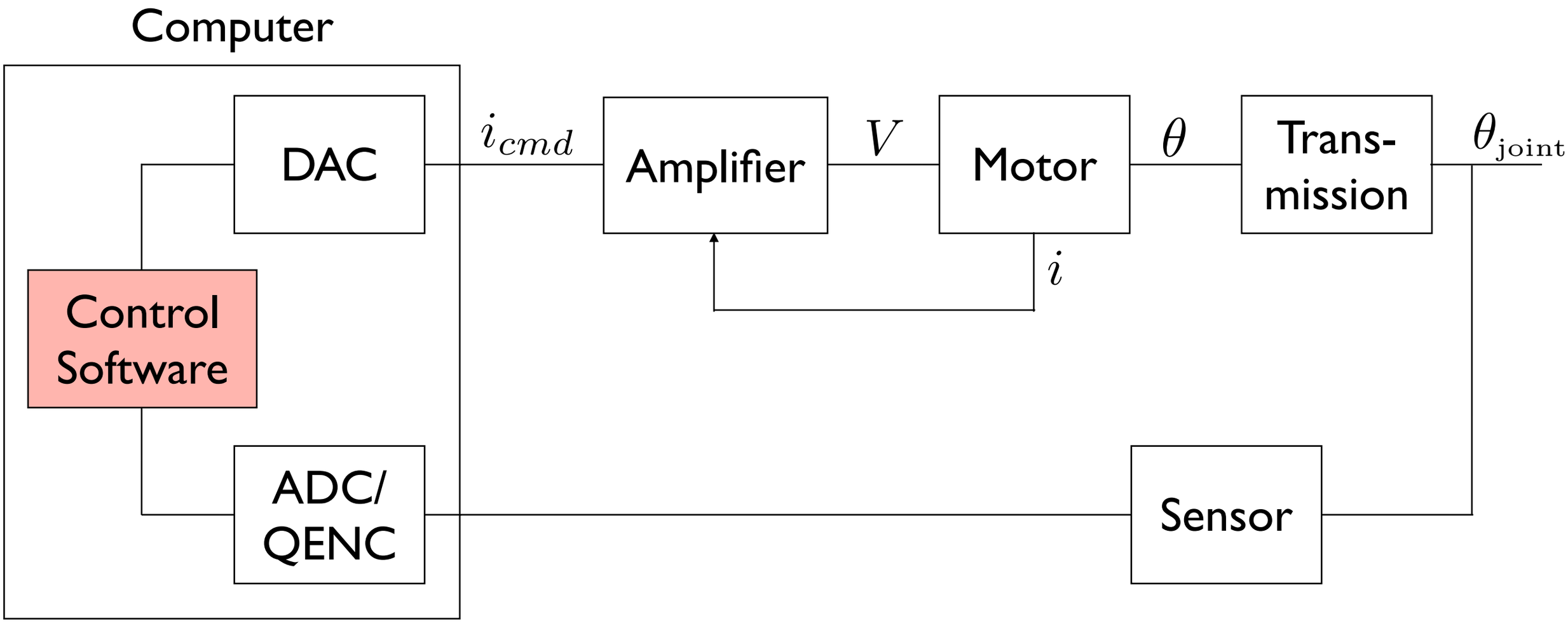
We are setting up the camera and recording slots.

What questions do you have about Project 2 ?

We will use the PHANToM as a haptic interface in Homework 9.

What else can we do with a PHANToM?

We can use it as an autonomous robot to learn about motion control.


Computer
DAC
i_{cmd}
Amplifier
V
Motor
θ
Trans-
mission
θ_{joint}
i
Control
Software
ADC/
QENC
Sensor

Common Controllers

Proportional Feedback

$$\tau_{\text{joint}} = K_P(\theta_{\text{desired}} - \theta_{\text{measured}})$$

Proportional Feedback Plus Viscous Damping

$$\tau_{\text{joint}} = K_P(\theta_{\text{desired}} - \theta_{\text{measured}}) - K_D\,\dot{\theta}_{\text{measured}}$$

Proportional Feedback Plus Derivative Feedback

$$\tau_{\text{joint}} = K_P(\theta_{\text{desired}} - \theta_{\text{measured}}) + K_D(\dot{\theta}_{\text{desired}} - \dot{\theta}_{\text{measured}})$$

Proportional Feedback Plus Viscous Damping plus Feedforward Gravity Compensation

$$\tau_{\text{joint}} = K_P(\theta_{\text{desired}} - \theta_{\text{measured}}) - K_D\,\dot{\theta}_{\text{measured}} + K_G \cdot fn(\theta_{\text{meas}})$$

What we use on the PUMA (plus an integral term)!

I wrote code that records movements of the Phantom.

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

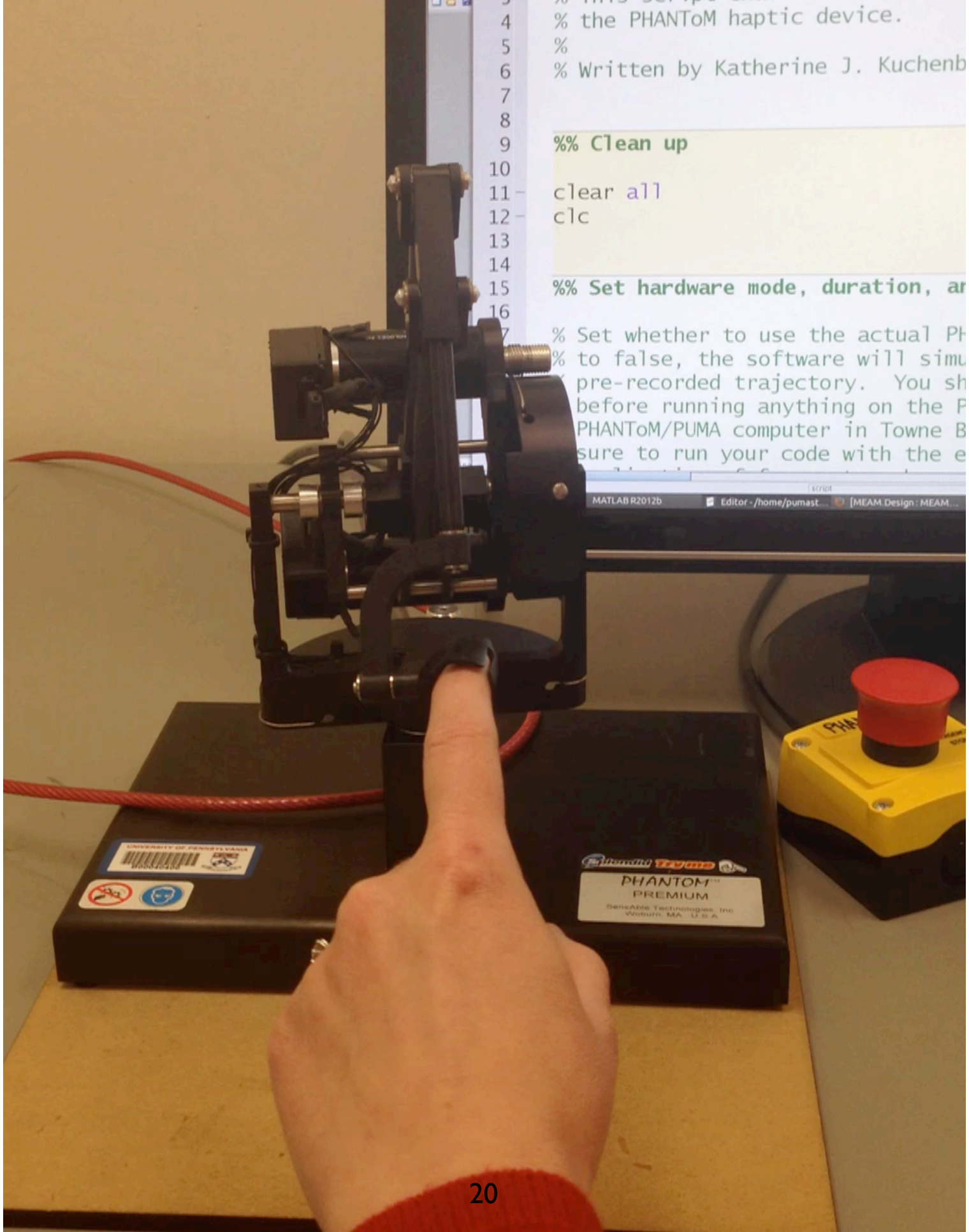
File Edit Text Go Cell Tools Debug Desktop Window Help

```
106
107     %% Run the servo loop
108 -   for i = 1:nCycles
109         % Measure the time and store it in our vector of time stamps.
110         % The units are seconds.
111 -       t(i) = toc;
112
113         % Get the Phantom's joint angles in radians.
114 -       theta123 = phantomJointAngles;
115
116         % Store joint angles in the matrix.
117 -       thetas(i,:) = theta123';
118
119         % Pull the three joint angles out of the vector to simplify calcul
120 -       theta1 = theta123(1);
121 -       theta2 = theta123(2);
122 -       theta3 = theta123(3);
123
124         % Calculate the position of the haptic device's tip using forward
125         % We store the result in hx, hy, and hz.  The units are millimeter
126 -       r = l2*cos(theta2) - l3*sin(theta3);
127 -       hx = r*cos(theta1);
128 -       hy = -r*sin(theta1);
129 -       hz = l1 - l2*sin(theta2) - l3*cos(theta3);
130
131         % Set all three force components to zero.
132 -       Fx = 0;
133 -       Fy = 0;
```

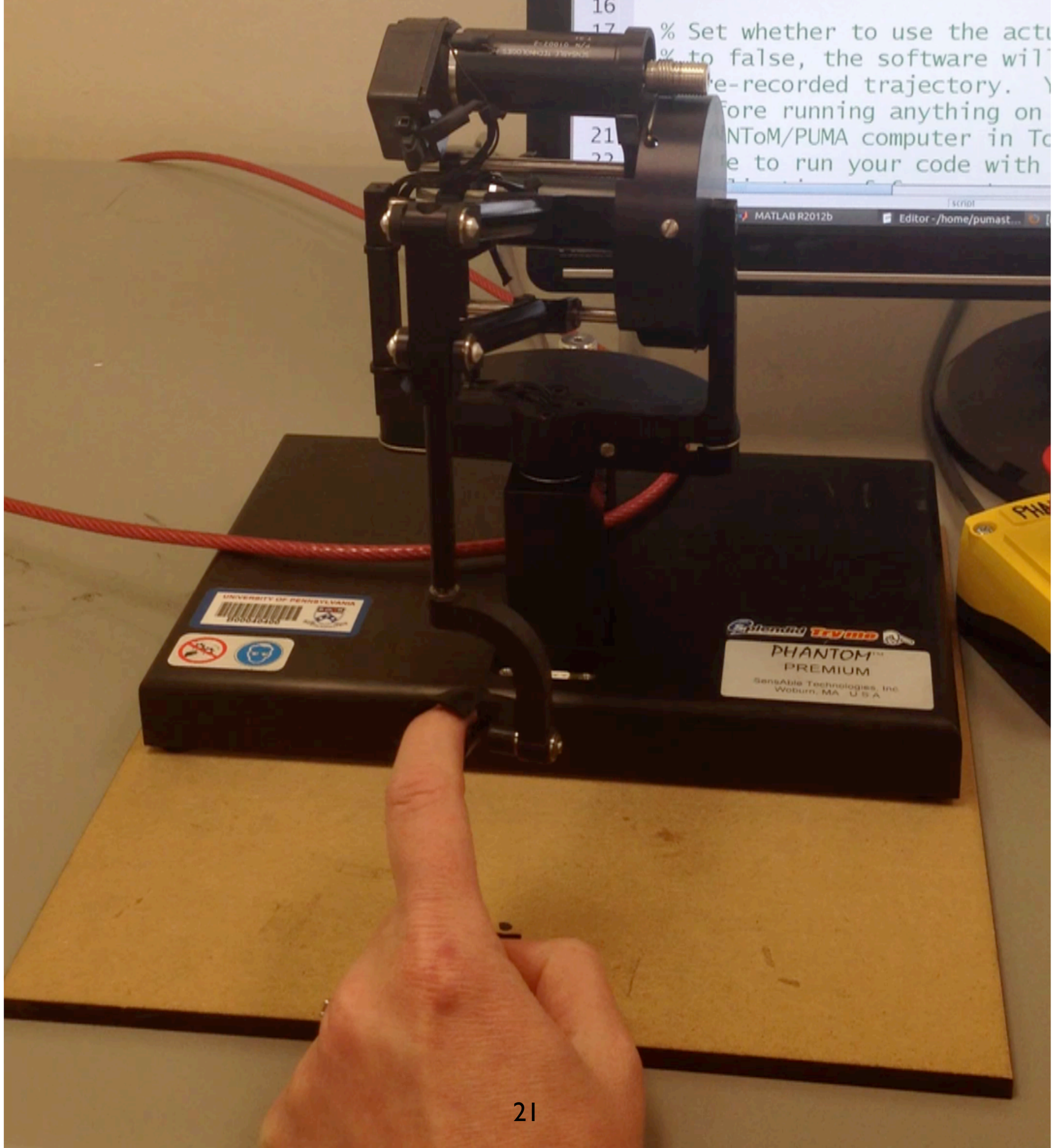
haptic_box_demo.m | haptic_ball_team50.m | haptic_damping_team50.m | joint_angle_recording.m

script Ln 116 Col 28

record loops



record taps


% Set whether to use the actu
% to false, the software wil
e-recorded trajectory.
ore running anything on
NToM/PUMA computer in To
e to run your code with
MATLAB R2012b
Editor - /home/pumast...
PHANTOM
PREMIUM
SensAble Technologies, Inc.
Woburn, MA U S A

I designed a controller to try to make the PHANToM replicate these recorded movements.

Desired Position

$$\vec{x}_{h,\mathrm{des}} = \Lambda(\theta_{1,\mathrm{des}}, \theta_{2,\mathrm{des}}, \theta_{3,\mathrm{des}})$$

$$x_{h,\mathrm{des}}, y_{h,\mathrm{des}}, z_{h,\mathrm{des}}$$

Actual Position

$$\vec{x}_{h} = \Lambda(\theta_{1}, \theta_{2}, \theta_{3})$$

$$x_{h}, y_{h}, z_{h}$$

Proportional Feedback Controller

$$\vec{F} = k(\vec{x}_{h,\mathrm{des}} - \vec{x}_{h})$$

$$F_x = k(x_{h,\mathrm{des}} - x_h)$$
$$F_y = k(y_{h,\mathrm{des}} - y_h)$$
$$F_z = k(z_{h,\mathrm{des}} - z_h)$$

Proportional position feedback acts like a spring with linear stiffness, pulling the robot's tip to the desired position.

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

File Edit Text Go Cell Tools Debug Desktop Window Help

```
137      % Pull the three joint angles out of the vector to simplify calcul
138 -    theta1 = theta123(1);
139 -    theta2 = theta123(2);
140 -    theta3 = theta123(3);
141
142      % Calculate the position of the haptic device's tip using forward
143      % We store the result in hx, hy, and hz.  The units are millimeter
144 -    r = l2*cos(theta2) - l3*sin(theta3);
145 -    hx = r*cos(theta1);
146 -    hy = -r*sin(theta1);
147 -    hz = l1 - l2*sin(theta2) - l3*cos(theta3);
148
149      % Pull desired joint angles from loaded trajectory.
150 -    theta1des = thetas_desired(i,1);
151 -    theta2des = thetas_desired(i,2);
152 -    theta3des = thetas_desired(i,3);
153
154      % Calculate the desired position for the haptic device.
155 -    rdes = l2*cos(theta2des) - l3*sin(theta3des);
156 -    hxdes = rdes*cos(theta1des);
157 -    hydes = -rdes*sin(theta1des);
158 -    hzdes = l1 - l2*sin(theta2des) - l3*cos(theta3des);
159
160      % Set all three force components.
161 -    Fx = k*(hxdes - hx);
162 -    Fy = k*(hydes - hy);
163 -    Fz = k*(hzdes - hz);
164
```

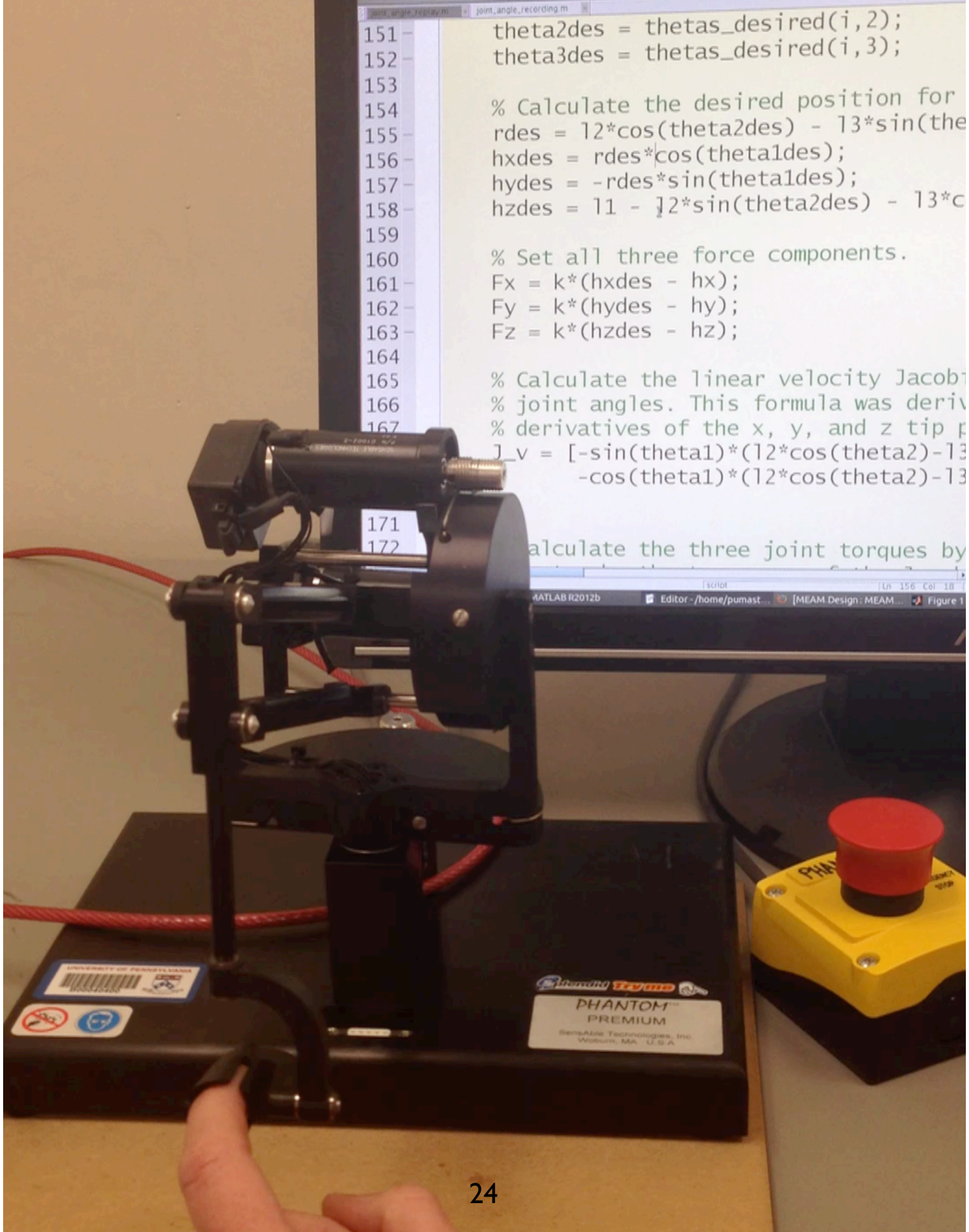
Forward Kinematics

Proportional Position Feedback Controller

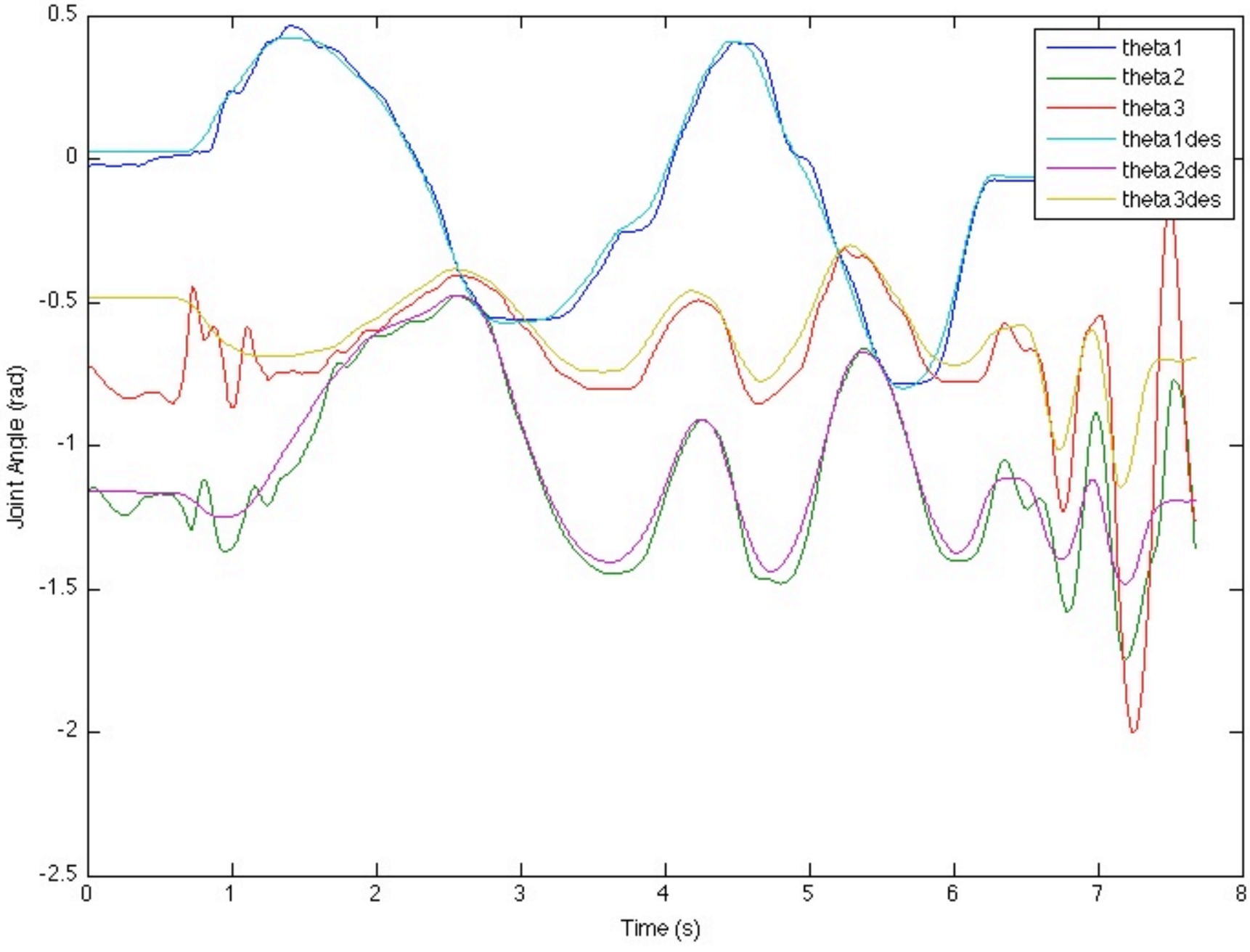
haptic_ball_team50.m | haptic_damping_team50.m | joint_angle_recording.m | joint_angle_replay.m

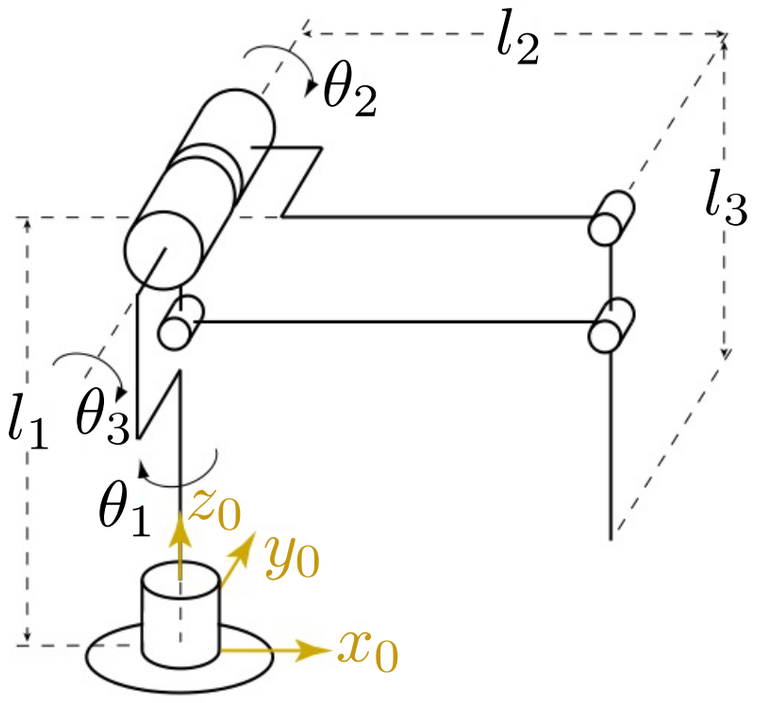
script Ln 156 Col 11

replay loops:
spring force,
fixed kinematics



Pretty good tracking!



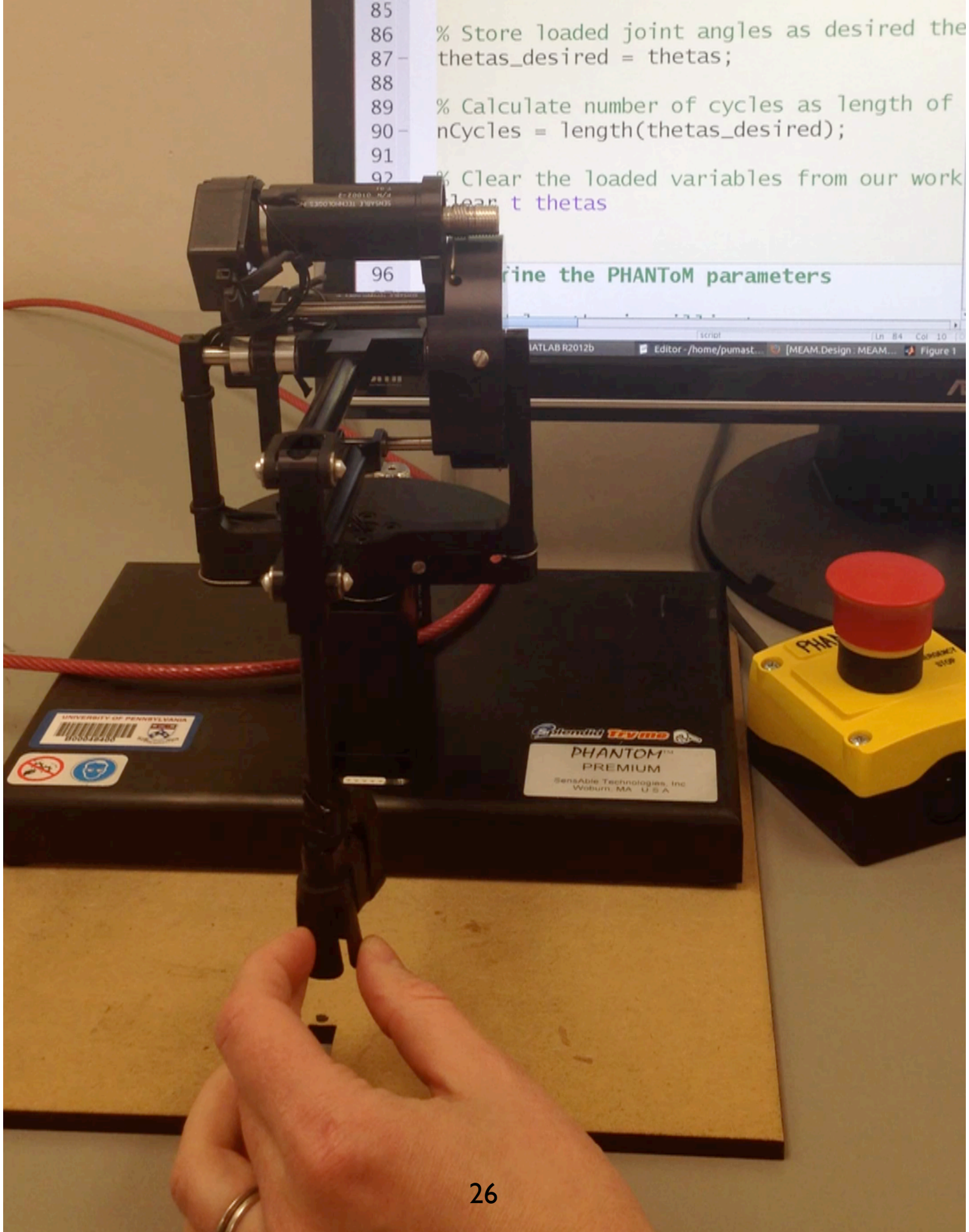


Inertia
Weight
Friction

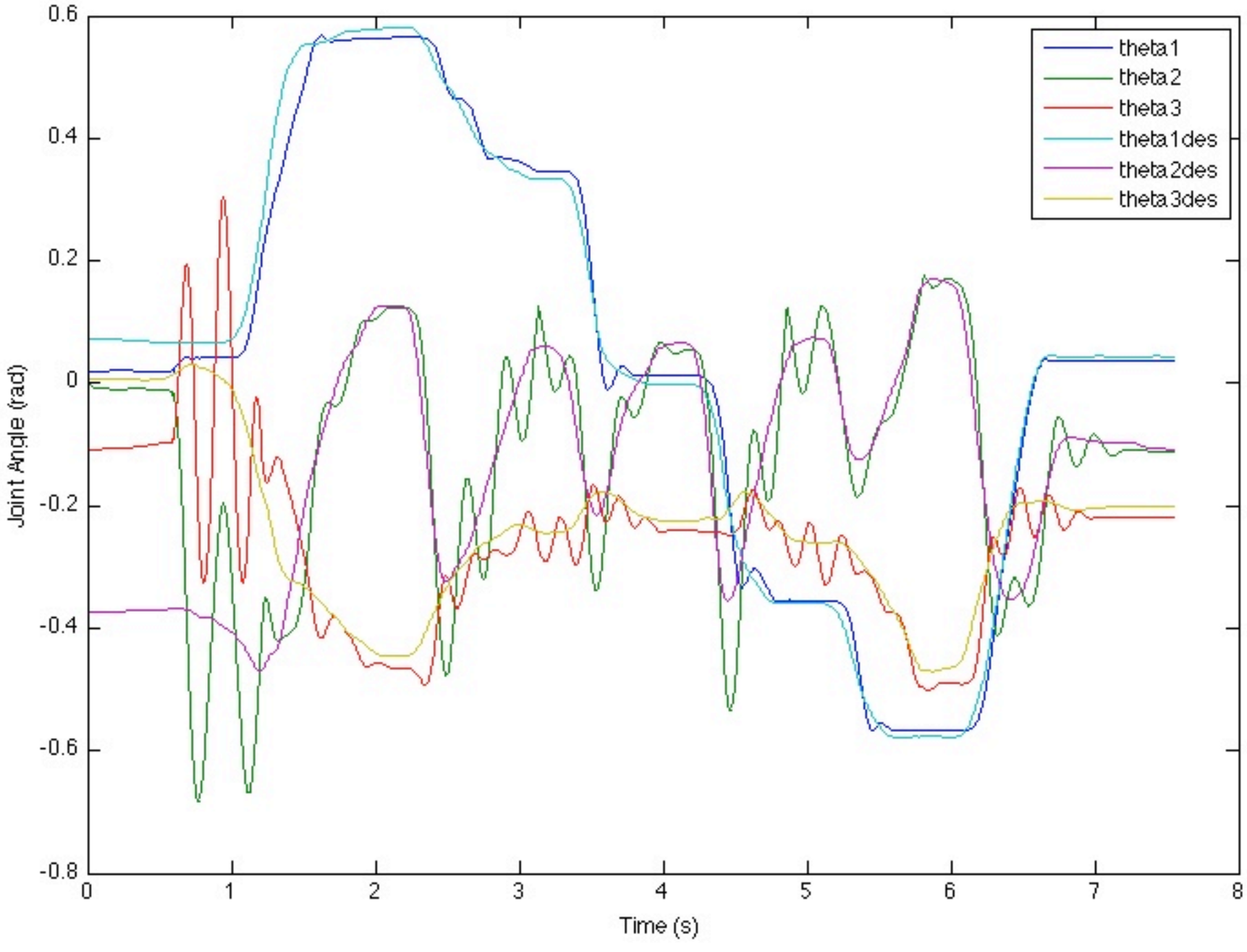
Why isn't it perfect?

The robot's dynamics interfere with tracking.

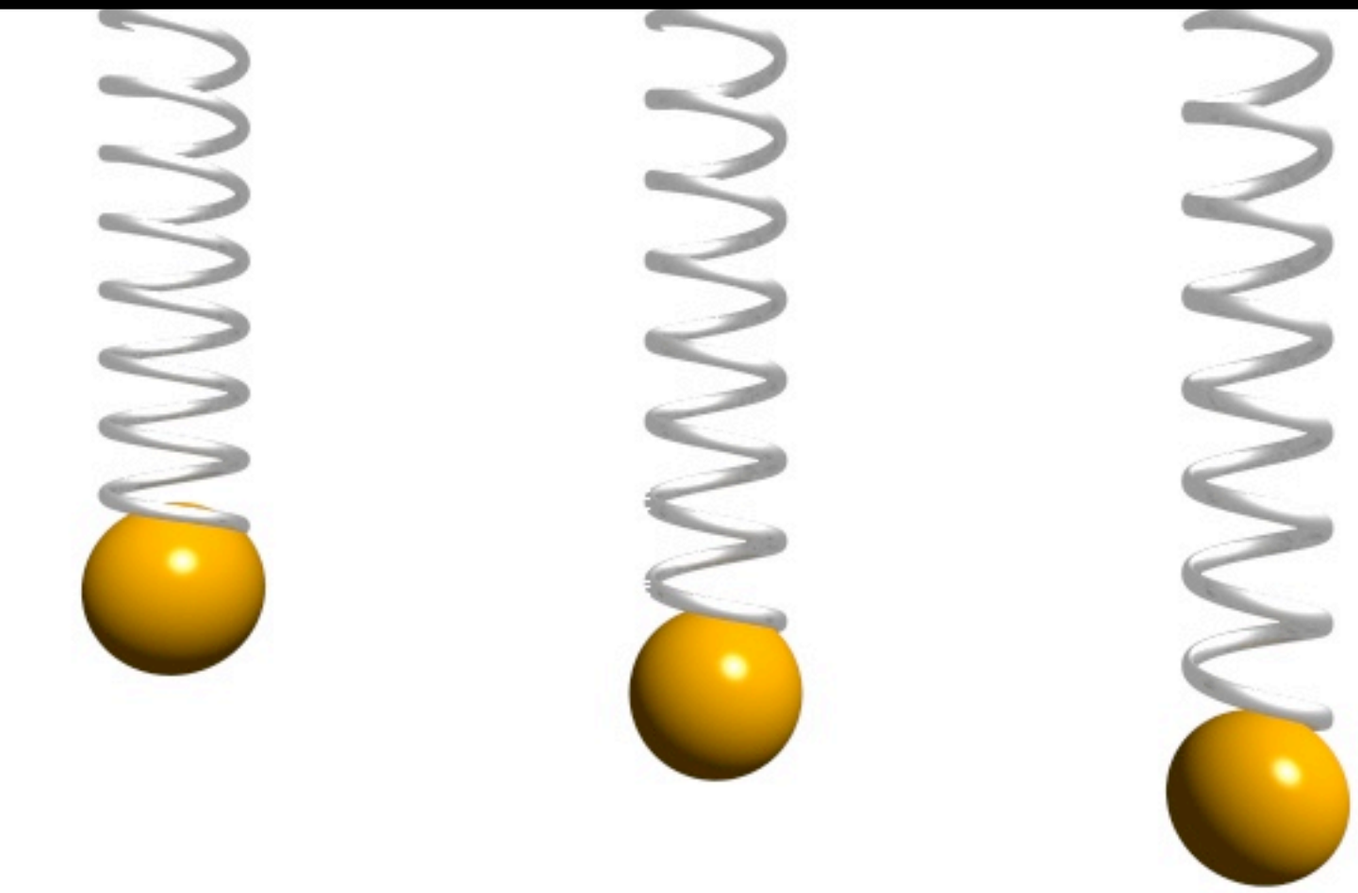
replay tap:
spring force

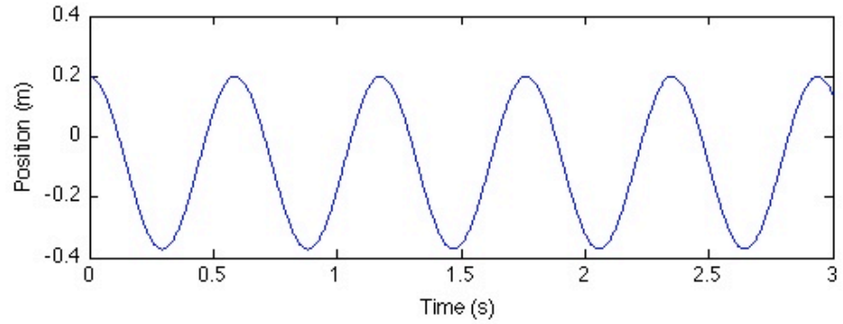


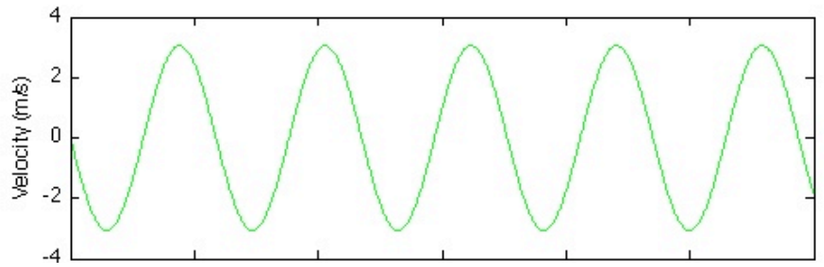
Similar effects – generally tracking, but very bouncy!

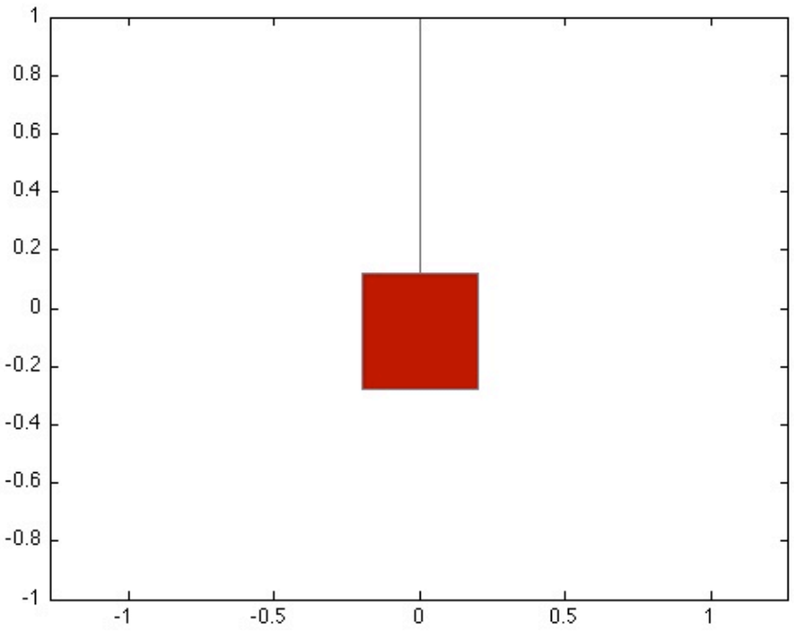


Mass on a spring: simple harmonic oscillator









simulate_mass_on_a_spring_ode45.m

Image from http://lifshitz.ucdavis.edu/~dmartin/phy7/7C.html

$$\Sigma F_y = m\ddot{y}$$

$$-mg - ky - b\dot{y} = m\ddot{y}$$

$$-mg = m\ddot{y} + b\dot{y} + ky$$

$$-g = \ddot{y} + \frac{b}{m}\dot{y} + \frac{k}{m}y$$

Second-order system

$$\frac{k}{m} = \omega_n^2$$

natural frequency

$$-g = \ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y$$

$$\frac{b}{m} = 2\zeta\omega_n$$

$$k_{\text{controller}} = m\omega_{n,\text{desired}}^2$$

$$b_{\text{controller}} = 2m\zeta_{\text{desired}}\omega_n - b_{\text{robot}}$$

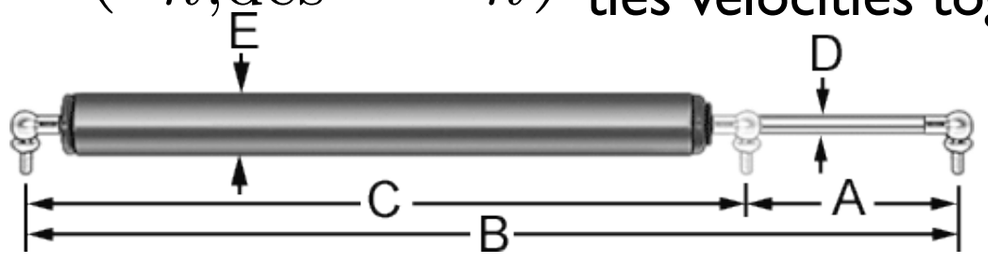
usually, $\zeta_{\text{desired}} = 1$ *damping ratio*

Add a derivative term to our position feedback controller, making it a Proportional Derivative (PD) controller.

$$\vec{F} = k(\vec{x}_{h,\text{des}} - \vec{x}_h) + b(\vec{v}_{h,\text{des}} - \vec{v}_h)$$

damper,
ties velocities together

spring,
ties positions together



$$F_x = k(x_{h,\text{des}} - x_h) + b(\dot{x}_{h,\text{des}} - \dot{x}_h)$$

$$F_y = k(y_{h,\text{des}} - y_h) + b(\dot{y}_{h,\text{des}} - \dot{y}_h)$$

$$F_z = k(z_{h,\text{des}} - z_h) + b(\dot{z}_{h,\text{des}} - \dot{z}_h)$$

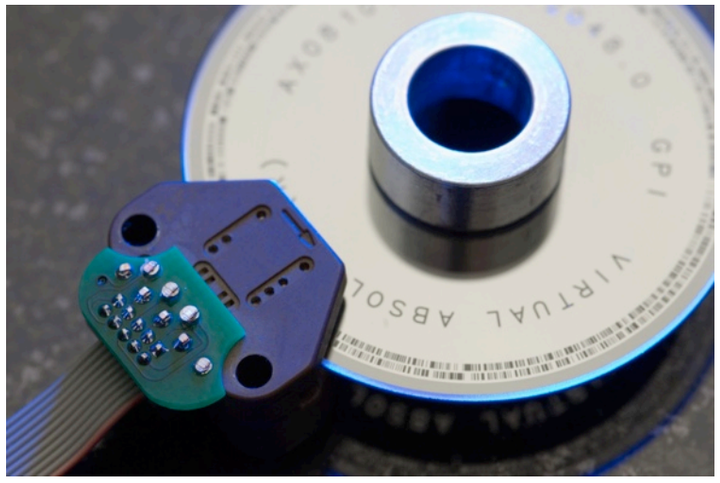
$$\vec{e}_h = \vec{x}_{h,\text{des}} - \vec{x}_h$$

$$\dot{\vec{e}}_h = \vec{v}_{h,\text{des}} - \vec{v}_h$$

$$\vec{F} = k\,\vec{e}_h + b\,\dot{\vec{e}}_h$$

How can we get a velocity signal?

Robots usually do not have dedicated velocity sensors.

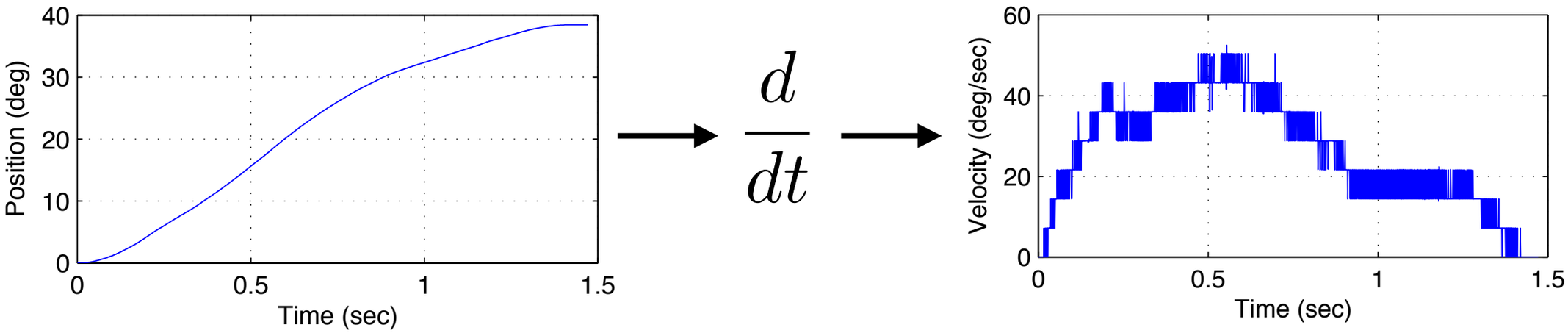


Differentiation of Position

discretized and quantized

usually requires filtering (which adds time delay)

$$\hat{\omega} = \frac{\theta(t_2) - \theta(t_1)}{t_2 - t_1}$$



$$\hat{v} = \frac{x(t_2) - x(t_1)}{t_2 - t_1}$$

$$0 < w < 1$$

filter weight

one minus filter weight

$$v_{\text{smooth},k} = w \cdot \hat{v}_k + (1 - w) \cdot v_{\text{smooth},k-1}$$

the kth smooth velocity estimate

the kth raw velocity estimate

the (k-1)th (previous) smooth velocity estimate

Editor - /Users/kuchenbe/Documents/teaching/haptics/lectures/14 teleoperation/matlab/van...

File Edit Text Go Cell Tools Debug Desktop Window Help

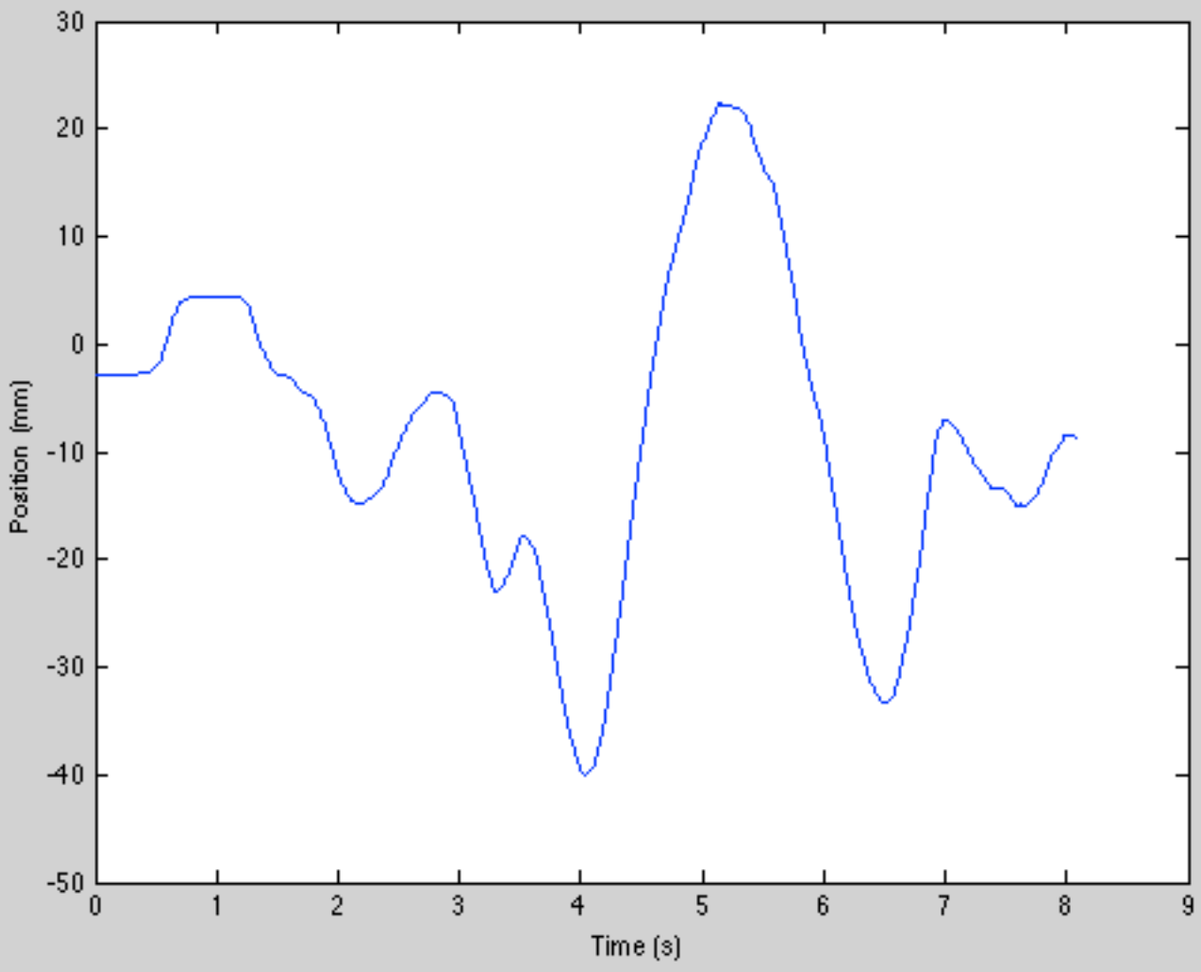
```
%% Clear workspace, load data, and plot position over time.
clear;

load data;

% Plot position over time.
figure(1)
plot(t,x,'b');
xlabel('Time (s)')
ylabel('Position (mm)')

%% Calculate raw velocity.

% Step through the vector and calculate raw velocity,
% being done in real time.
vraw = zeros(length(x),1);
for i = 2:length(x)
    vraw(i) = (x(i) - x(i-1)) / (t(i) - t(i-1));
end

% Plot raw velocity over time.
figure(2)
plot(t(2:end),vraw(2:end),'g')
xlabel('Time (s)')
ylabel('Raw Velocity (mm/s)')

%% Compute smoothed velocity.

% Step through the vector and calculate smooth veloci
% being done in real time.
w = 0.01;
vsmooth = zeros(length(x),1);
for i = 2:length(vraw)
    vsmooth(i) = w*vraw(i) + (1-w) * vsmooth(i-1);
end

% Plot raw and smoothed velocities over time.
figure(3)
plot(t(2:end),vraw(2:end),'g',t(2:end),vsmooth(2:end)

% Compute filter bandwidth.
T = mean(diff(t));
lambda = w / (T - w*T);

fc = lambda / (2*pi);
text(1,175,sprintf('The filter weight is %0.4g',w))
text(1,150,sprintf('The filter bandwidth is %0.3g Hz',fc))
```

lambda_plot.m | vanalysis.m

script Ln 4 Col 11

Figure 1

File Edit View Insert Tools Desktop Window Help



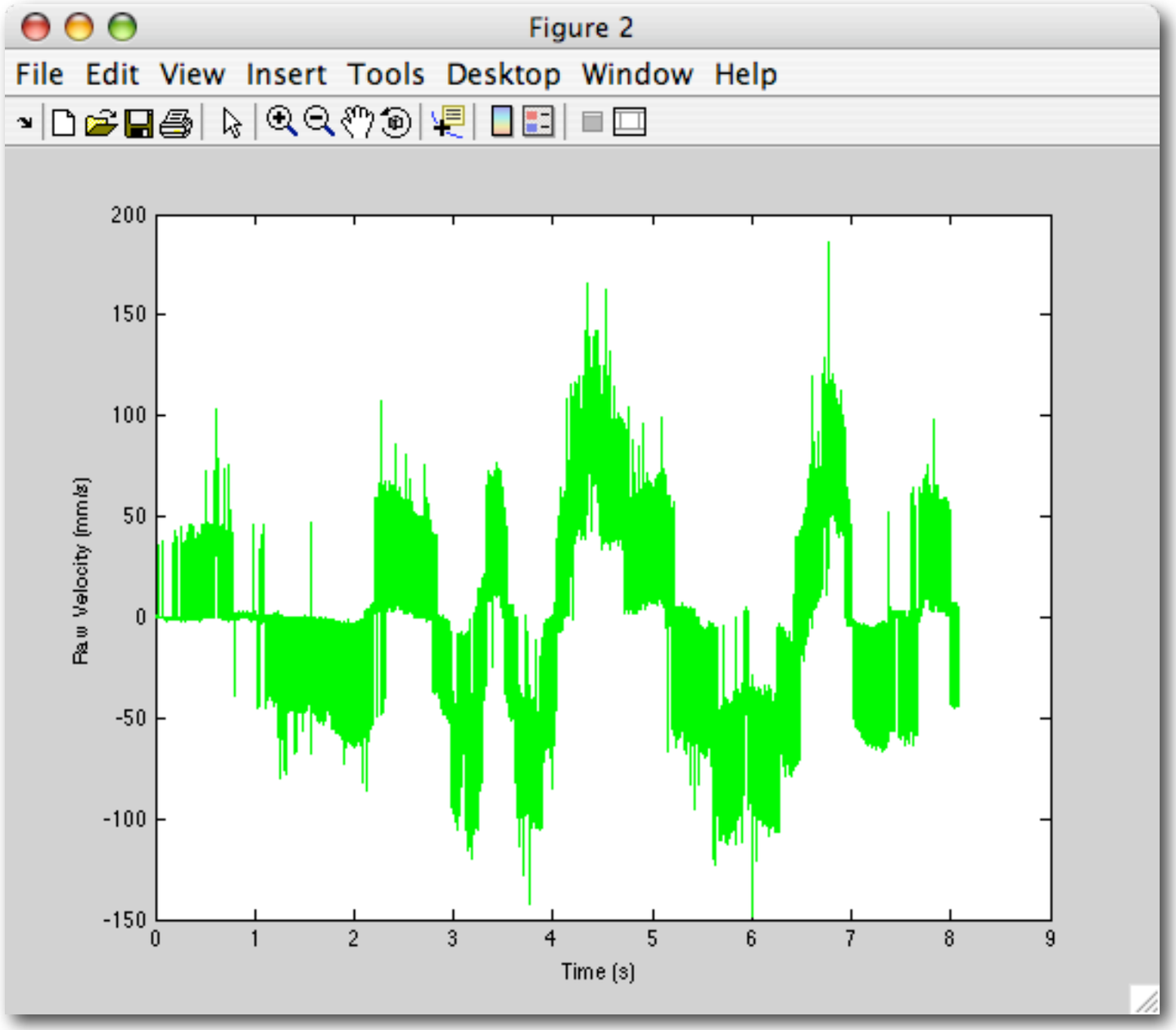


Figure 2
File Edit View Insert Tools Desktop Window Help
200
150
100
50
0
-50
-100
-150
Raw Velocity (mm/s)
0 1 2 3 4 5 6 7 8 9
Time (s)

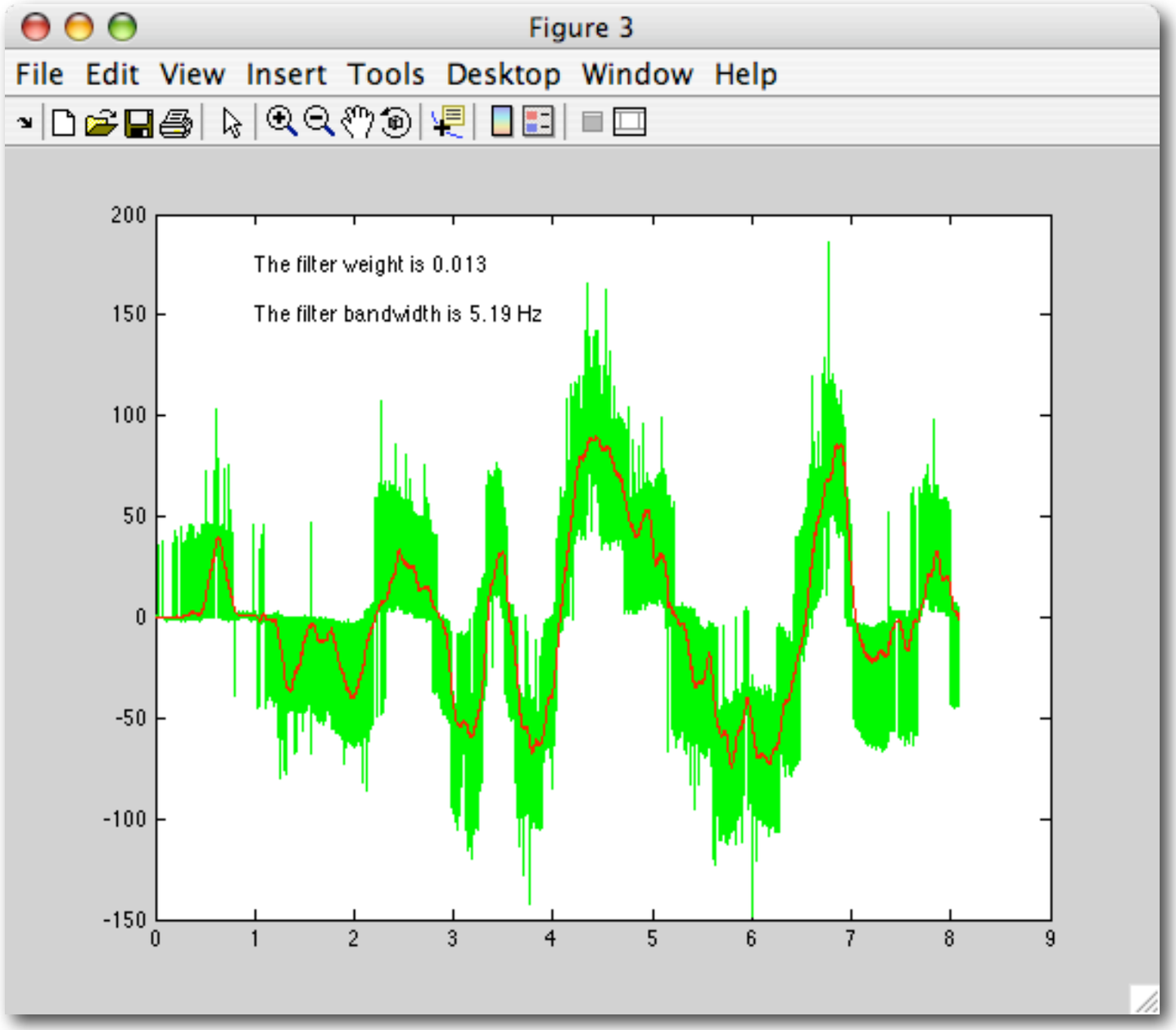


Figure 3
File Edit View Insert Tools Desktop Window Help
The filter weight is 0.013
The filter bandwidth is 5.19 Hz
200
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6
7
8
9

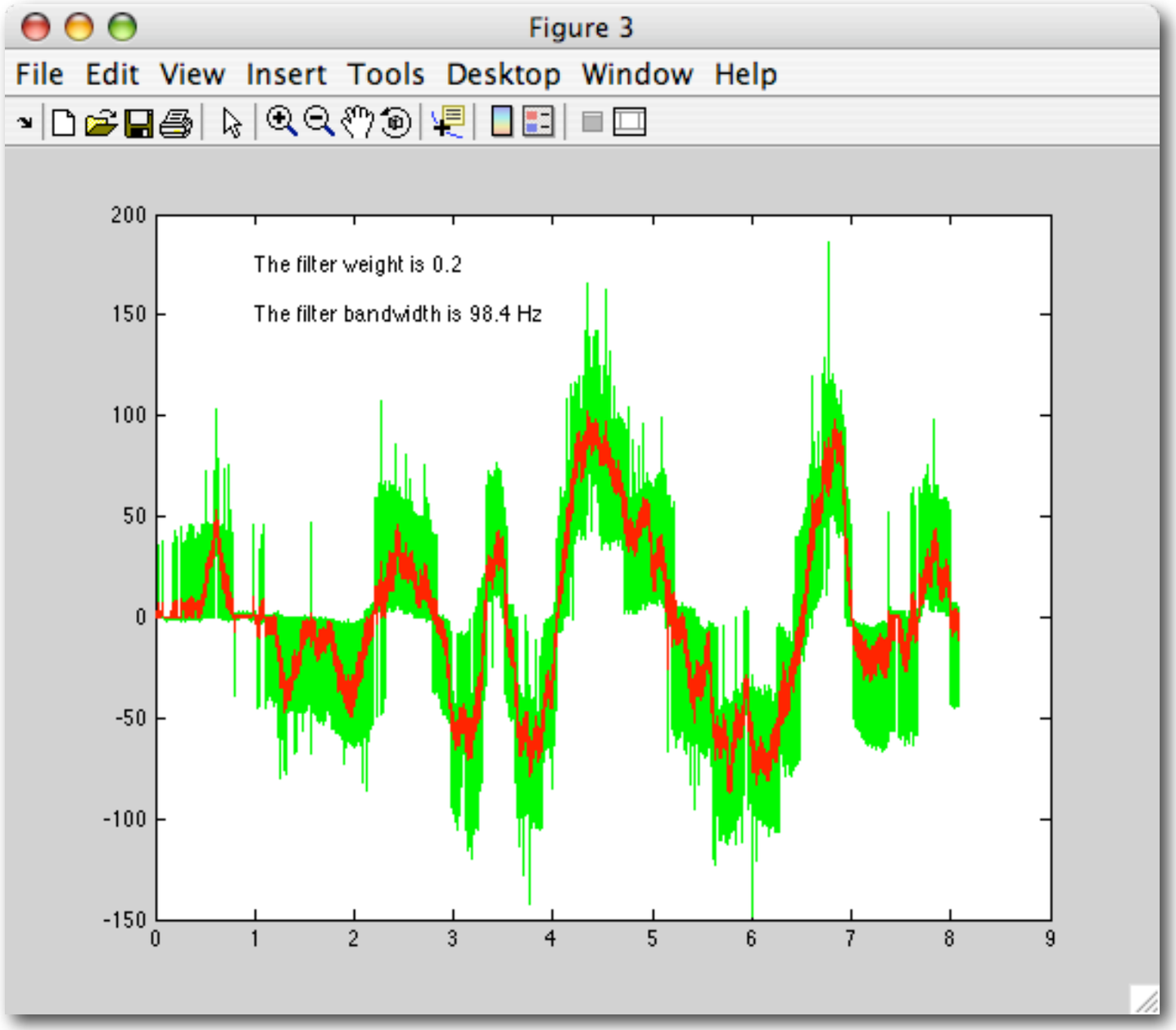


Figure 3
File Edit View Insert Tools Desktop Window Help
The filter weight is 0.2
The filter bandwidth is 98.4 Hz
200
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6
7
8
9

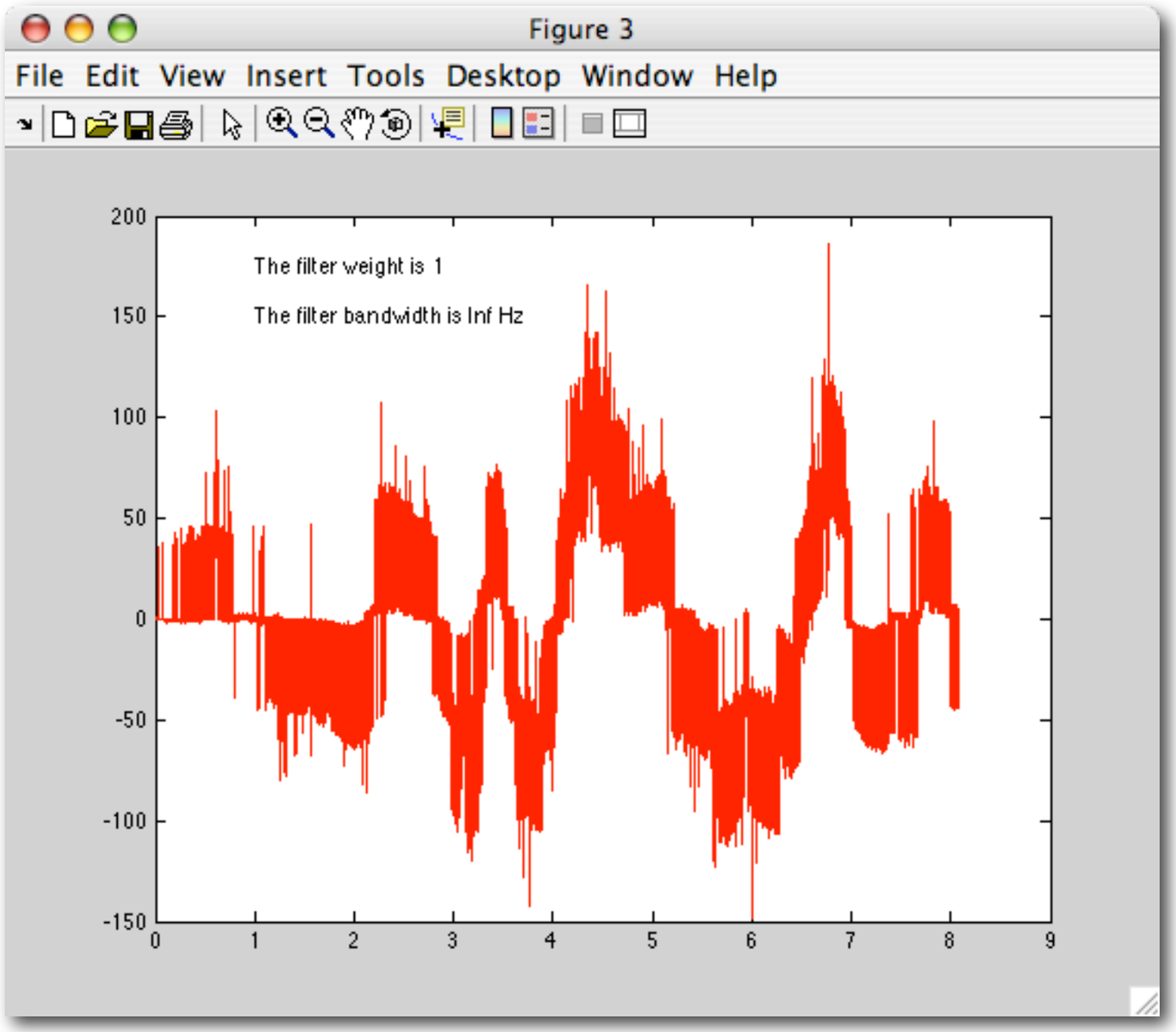


Figure 3
File Edit View Insert Tools Desktop Window Help
The filter weight is 1
The filter bandwidth is Inf Hz
200
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6
7
8
9

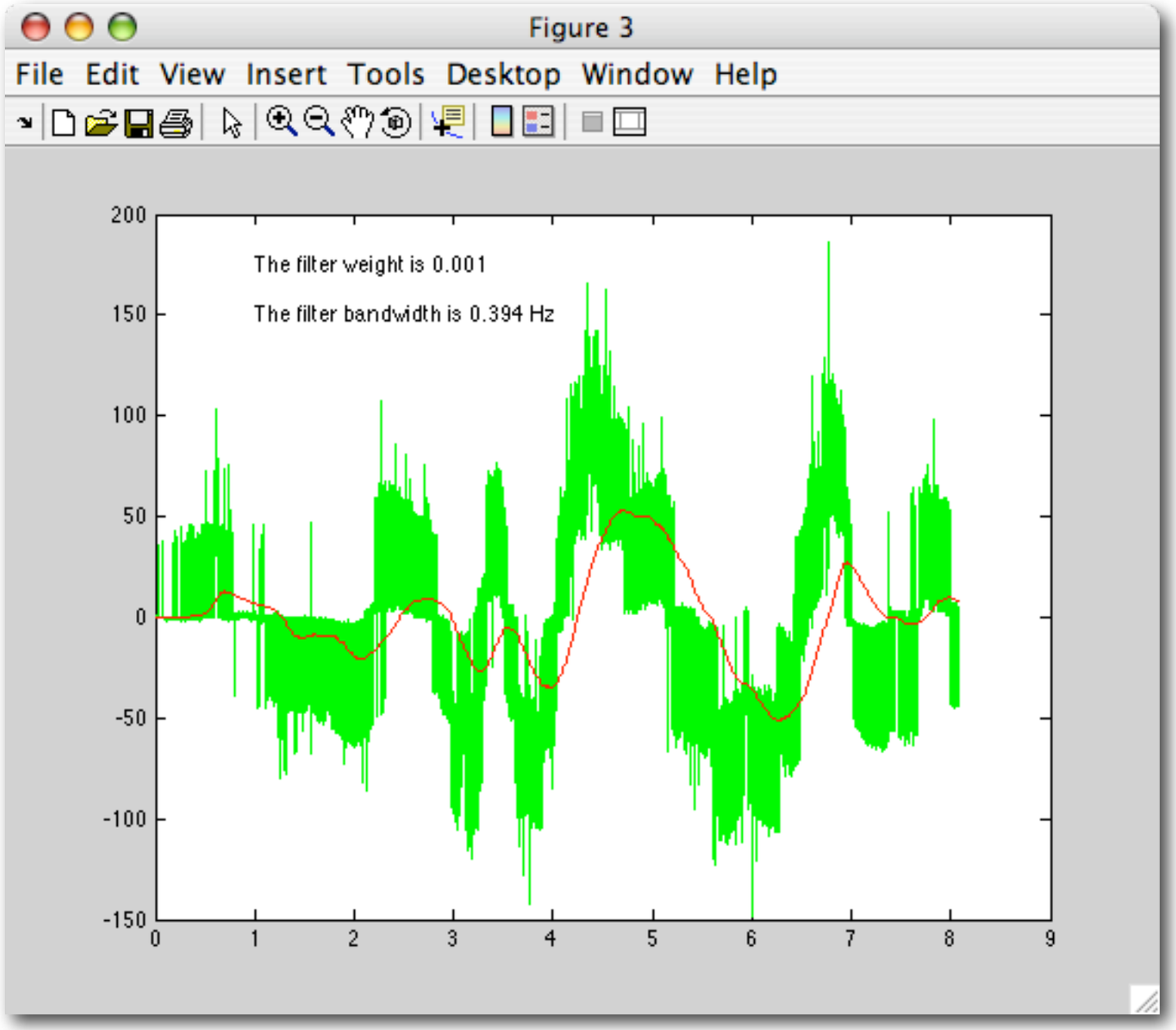


Figure 3
File Edit View Insert Tools Desktop Window Help
The filter weight is 0.001
The filter bandwidth is 0.394 Hz
200
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6
7
8
9

Infinite horizon (fading-memory) low-pass filter

KJK
24 Jan 2007

Begin with a first-order continuous-time low-pass filter, where $Y(s)$ is the Laplace transform of the filtered output and $X(s)$ is that of our signal.

$$\frac{Y(s)}{X(s)} = \frac{\lambda}{s+\lambda}$$

output $Y(s)$; input $X(s)$; gain = 1 @ $s=0$

one pole at $-\lambda$, λ is filter cutoff in rad/s

Convert from continuous time (smooth derivatives) to discrete time (sampled at $f=\frac{1}{T}$ intervals of T seconds). This requires us to choose a method for approximating the derivative. Other options would work too, but the simplest is backward differencing:

$$s = \frac{(1 - z^{-1})}{T}$$

makes sense: this value minus last value divided by T.

z transform acts like a shift operator:

$Y(z) * z^{-1}$ is previous y value

$Y(z) * z^0 = Y(z)$ is this y value

$Y(z) * z$ is next y value

Substitute this in for s in the above equ.

$$\frac{Y(z)}{X(z)} = \frac{\lambda}{\frac{(1-z^{-1})}{T} + \lambda}$$

$$Y(z)\left[\frac{(1-z^{-1})}{T} + \lambda\right] = \lambda X(z)$$

$$Y(z) - Y(z) * z^{-1} + \lambda T Y(z) = \lambda T X(z)$$

$$(1+\lambda T) Y(z) = \lambda T X(z) + z^{-1} * Y(z)$$

$$Y(z) = \frac{\lambda T}{1+\lambda T} X(z) + \frac{1}{1+\lambda T} z^{-1} Y(z)$$

do inverse z transform

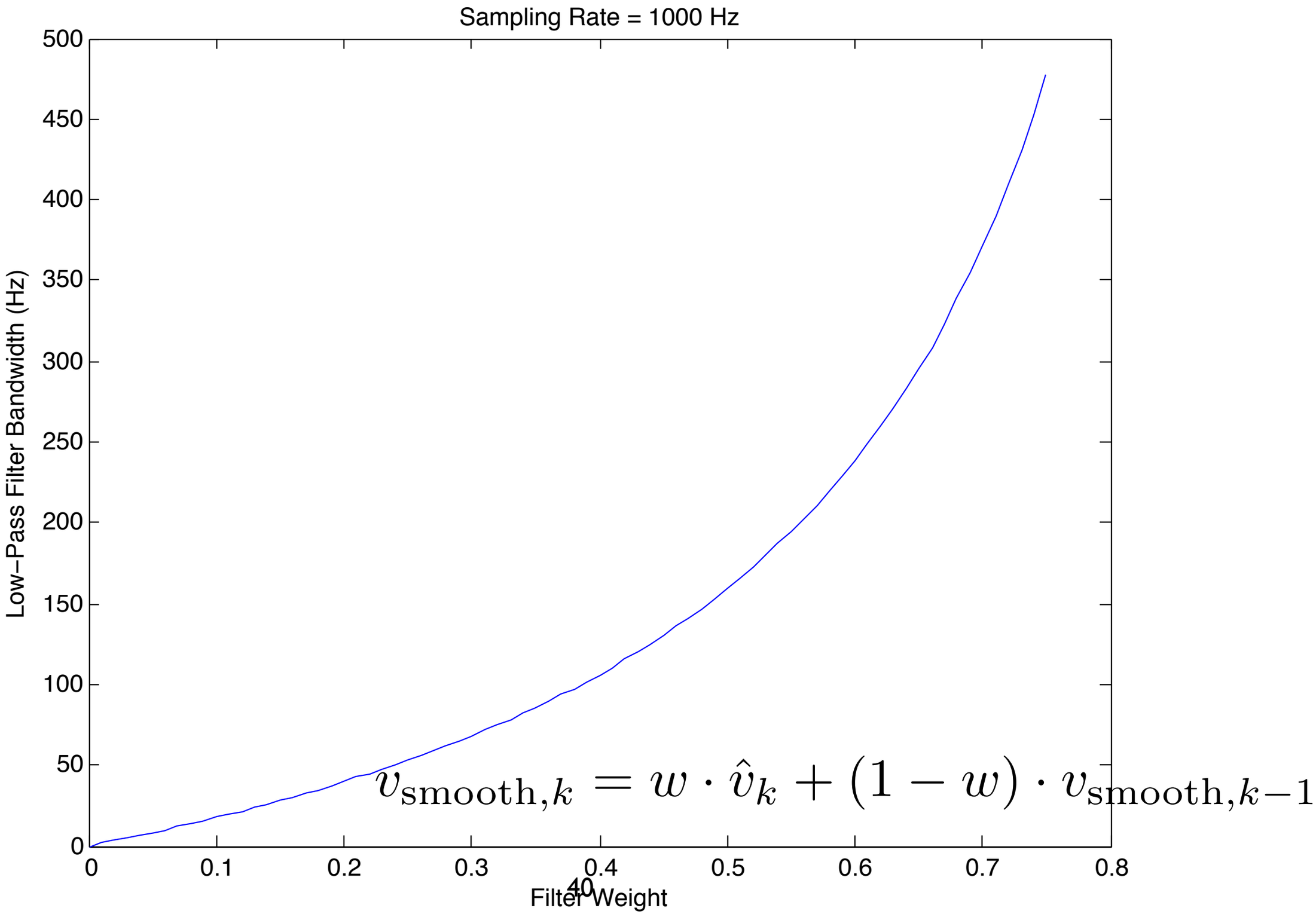
$$y(k) = \frac{\lambda T}{1+\lambda T} x(k) + \frac{1}{1+\lambda T} y(k-1)$$

index

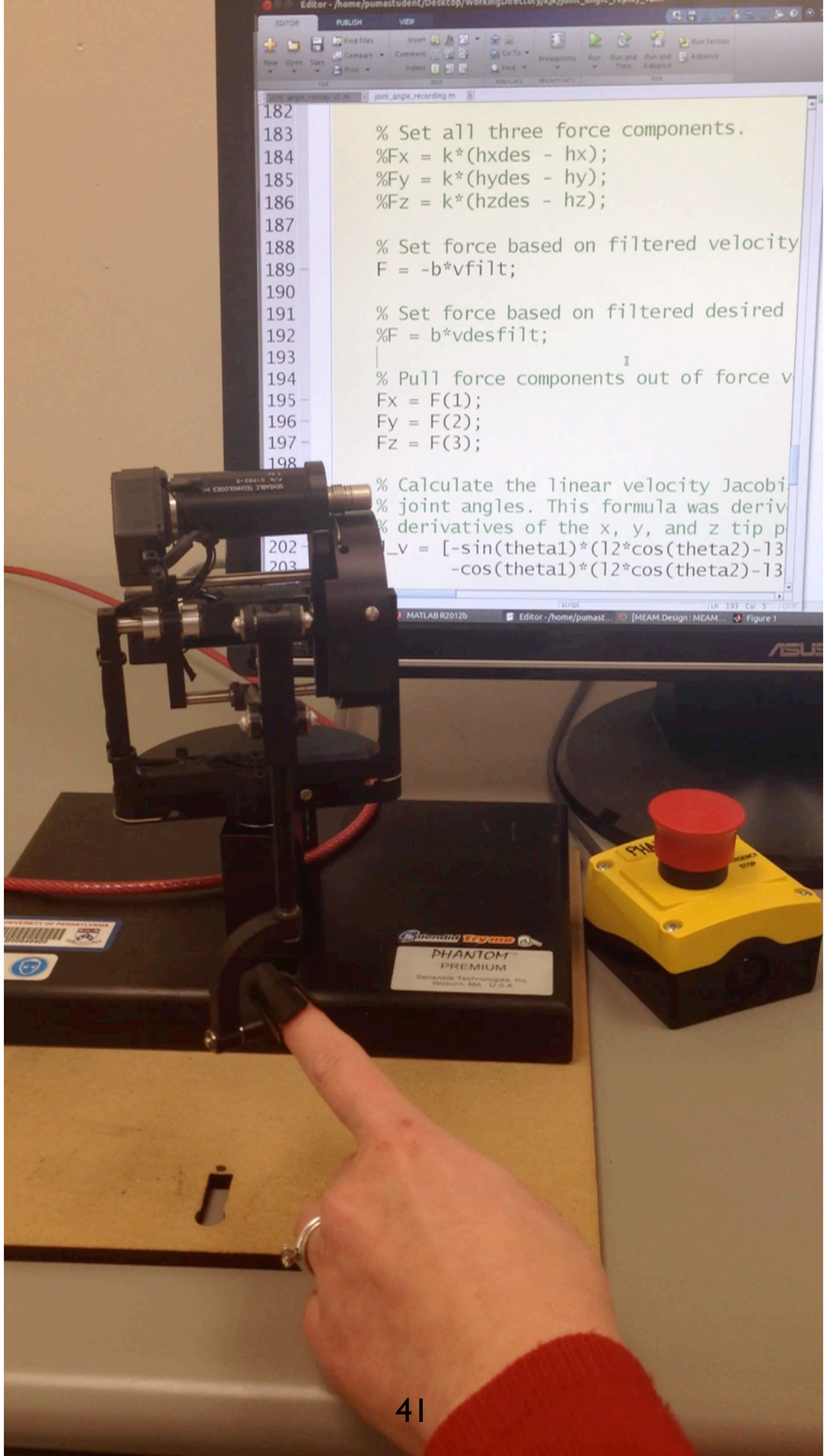
$$y(k) = w \cdot x(k) + (1-w) y(k-1)$$

$$\text{filter\_weight} = \frac{\lambda T}{1+\lambda T}$$

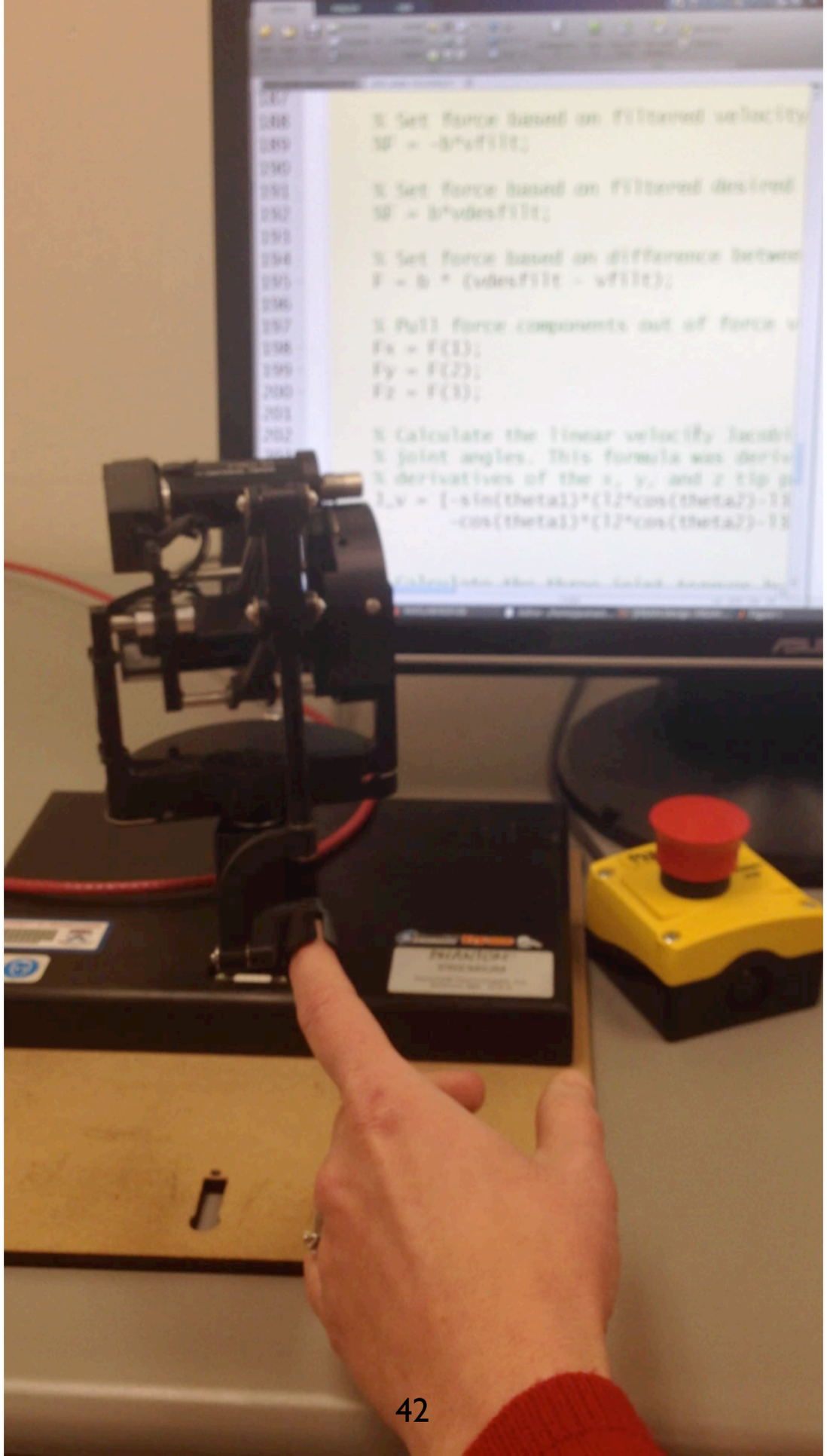
$$\lambda = \frac{w}{T(1-w)} \qquad f = \lambda \cdot \frac{1 \text{ cycle}}{2\pi \text{ rad}} = \frac{w}{T(1-w)} \cdot \frac{1 \text{ cycle}}{2\pi \text{ rad}}$$



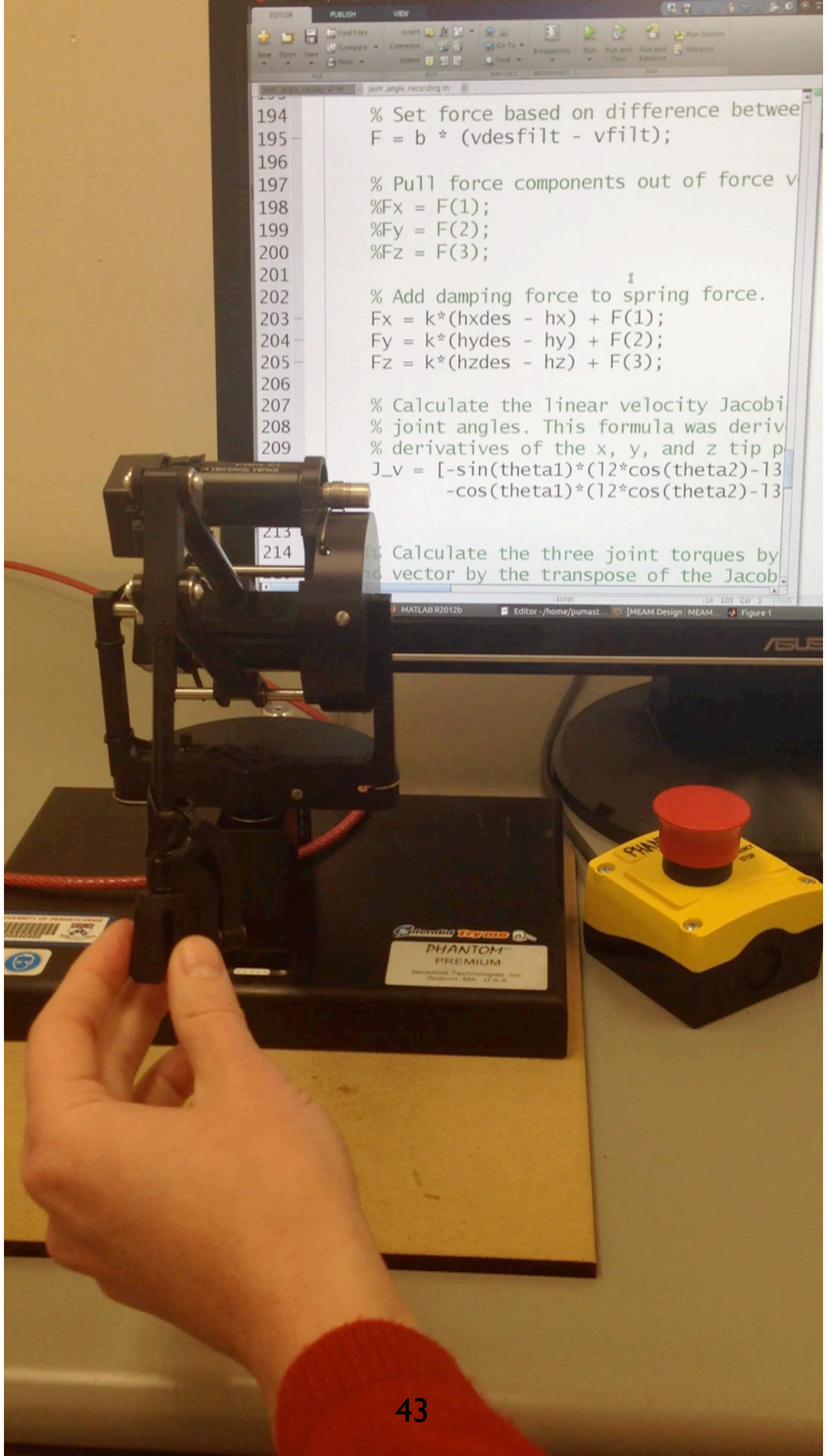
damping individual



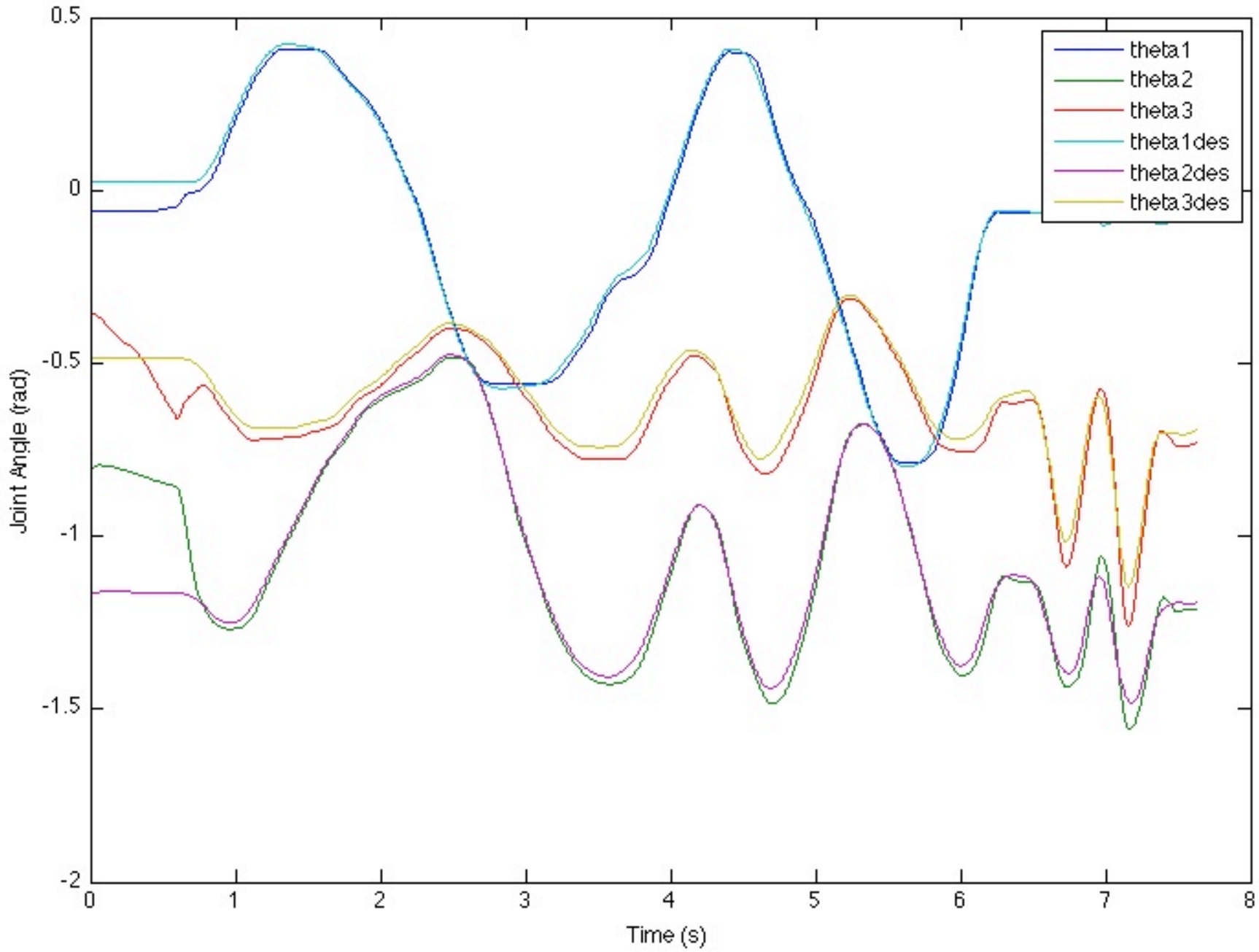
damping both (blurry)



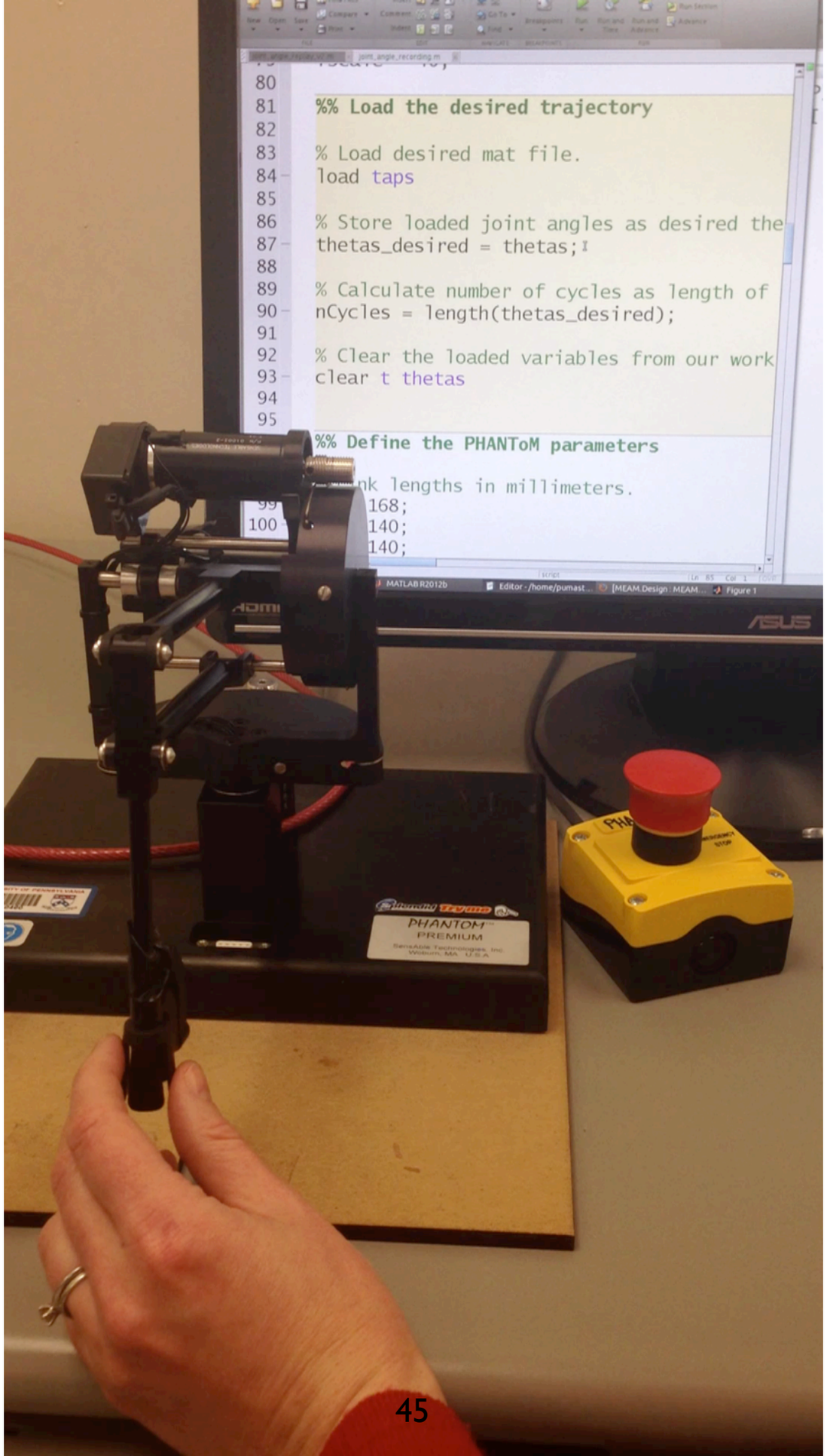
replay loops final



Very nice!


theta1
theta2
theta3
theta1des
theta2des
theta3des
Joint Angle (rad)
Time (s)
0.5
0
-0.5
-1
-1.5
-2
0
1
2
3
4
5
6
7
8

replay taps final

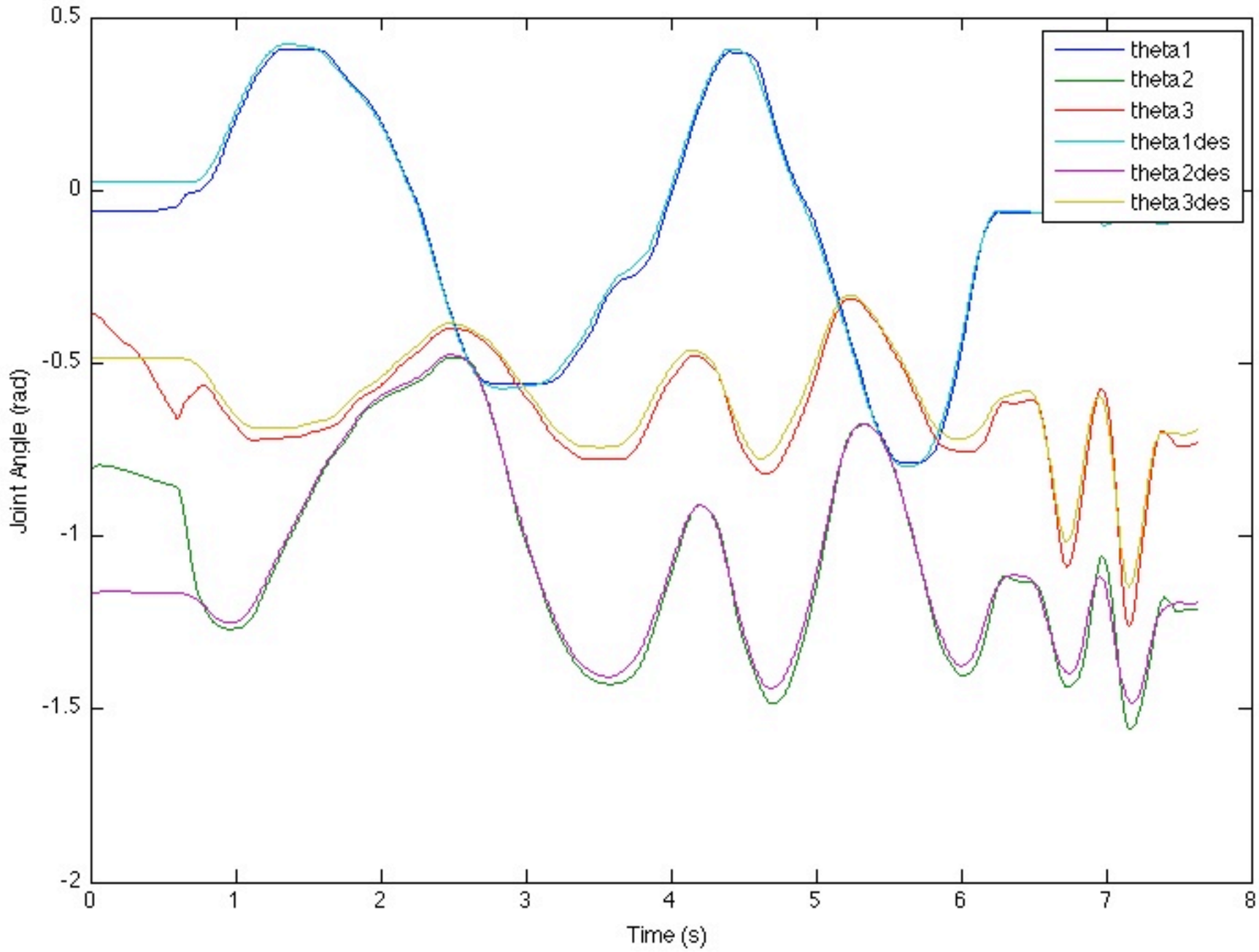


Is it perfect?
Of course not.

What questions do you have about the PD controller ?

Let's watch Teams 101–108.

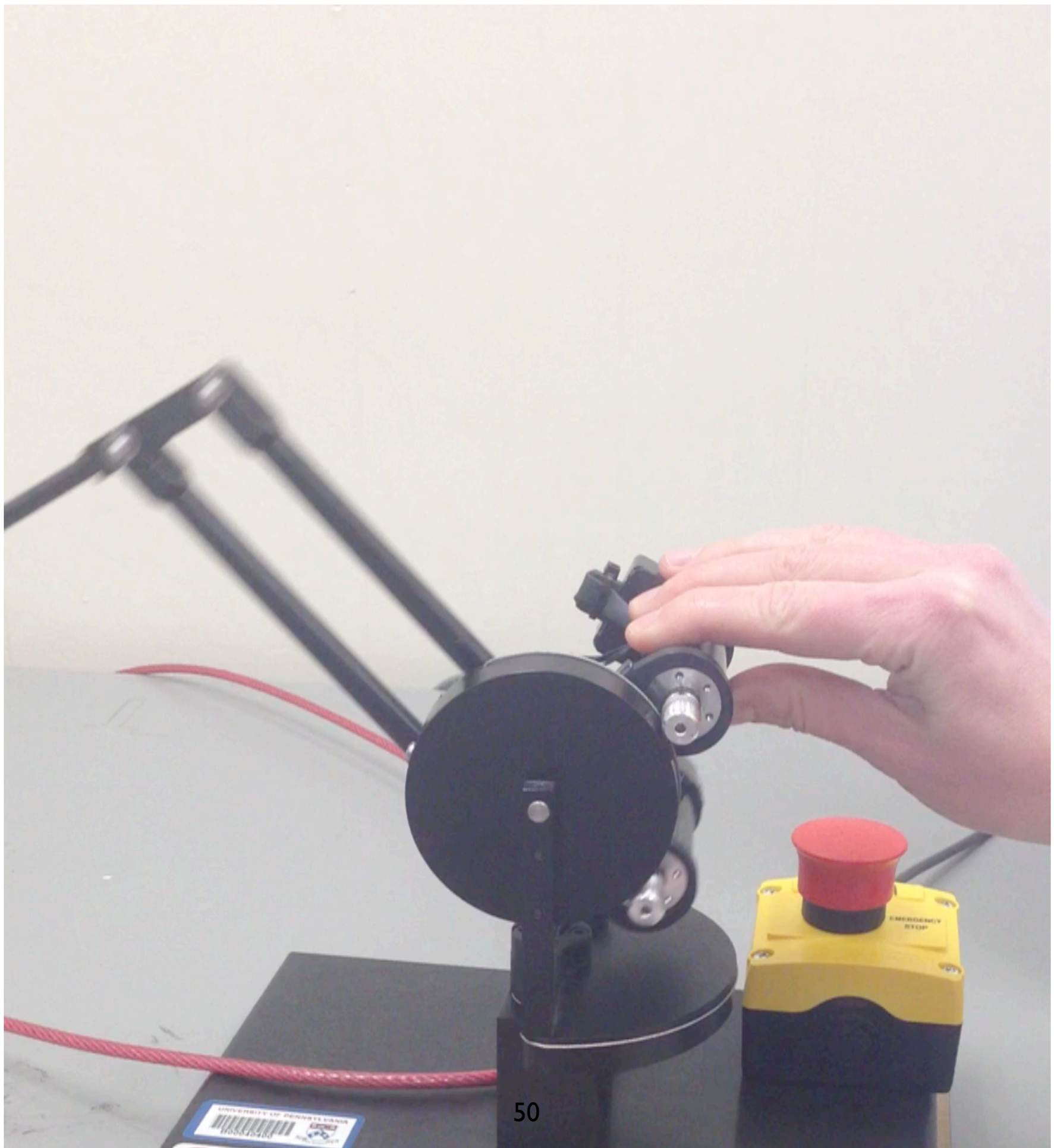
Very nice!



Is it perfect?

How can we improve the controller's tracking?

Add gravity compensation!


EMERGENCY
STOP

How could we compensate for gravity?

mechanically (add weight near the tip)
mechanically (add carefully tuned springs)
in software with integral feedback (use the motors)
in software with feedforward (use the motors)

Which option is best?

All approaches to gravity compensation are useful.

For haptics and robotics, a small amount of feedforward software gravity compensation is useful to avoid having to increase the inertia of the system. Springs are hard to calibrate and install.

For Phantom, first focus on joint 3, because its inherent gravity balance is worse than joint 2.

Gravity compensation is a form of feedforward control (SHV 6.4)

with updated gravity compensation on joints 2 and 3