MEAM 520

Haptic Rendering

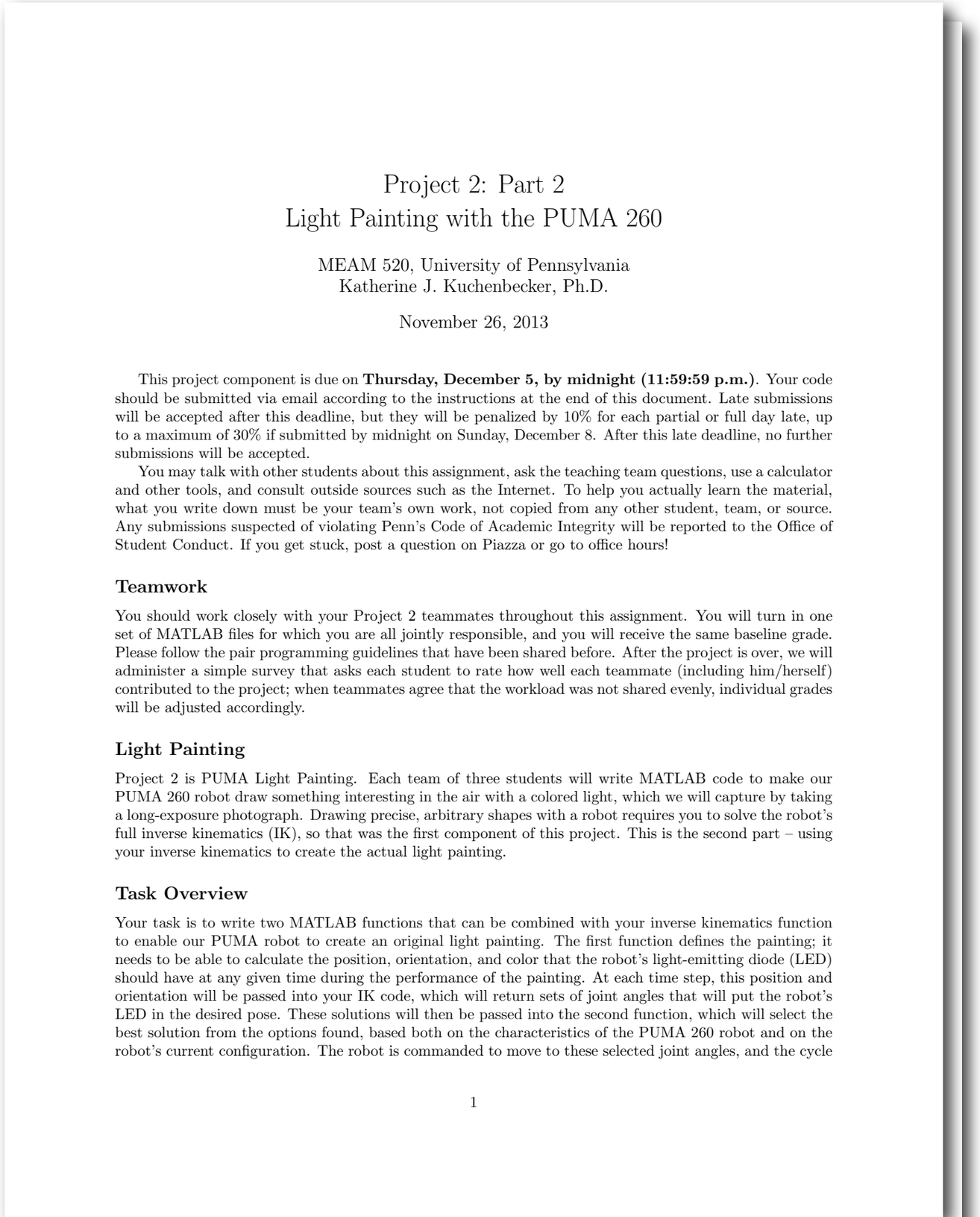
Katherine J. Kuchenbecker, Ph.D.

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

MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 26: December 5, 2013



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

1

Part 2 of Project 2 is due tonight, with the late deadline on Sunday.

Submissions have begun coming in, and the light paintings are looking great!

Many more teams have solved the IK. Good job!

Many thanks to the students who have been helping out!

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

Still need help?

I have office hours today from 2:30 to 4:00 p.m. (extended).

Your IK doesn't need eight perfect solutions to make a good light painting.

Tentative MEAM 520 Calendar

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

Tuesday 12/10 – Teleoperation & 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.

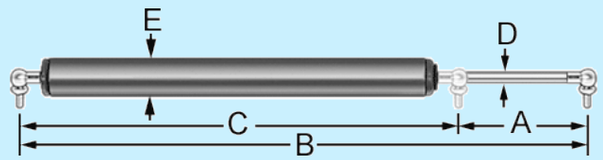
Motion Control via a Cartesian PD Controller

$$\vec{x}_{h,\mathrm{des}} = \Lambda(\theta_{1,\mathrm{des}}, \theta_{2,\mathrm{des}}, \theta_{3,\mathrm{des}})$$
$$\vec{x}_h = \Lambda(\theta_1, \theta_2, \theta_3)$$

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

virtual spring
ties positions together
causes oscillations

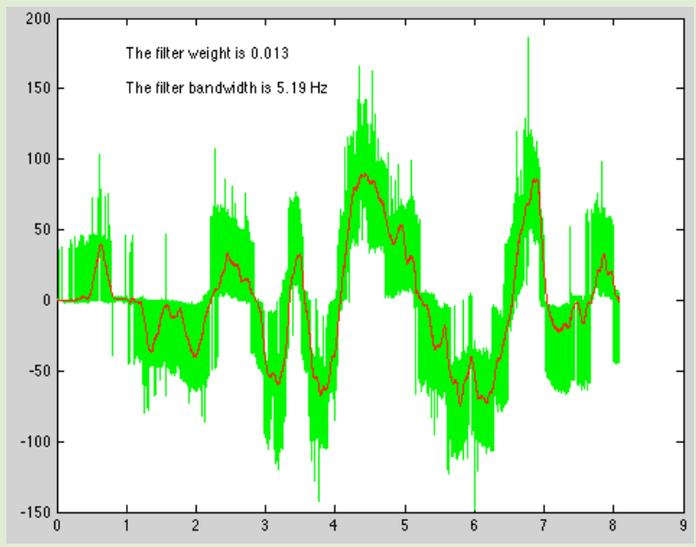
virtual damper
ties velocities together
calms oscillations



$$\frac{k}{m} = \omega_n^2 \quad \textit{natural frequency}$$

$$\frac{b}{m} = 2\zeta\omega_n \quad \textit{damping ratio}$$

We must calculate velocity from the position signal

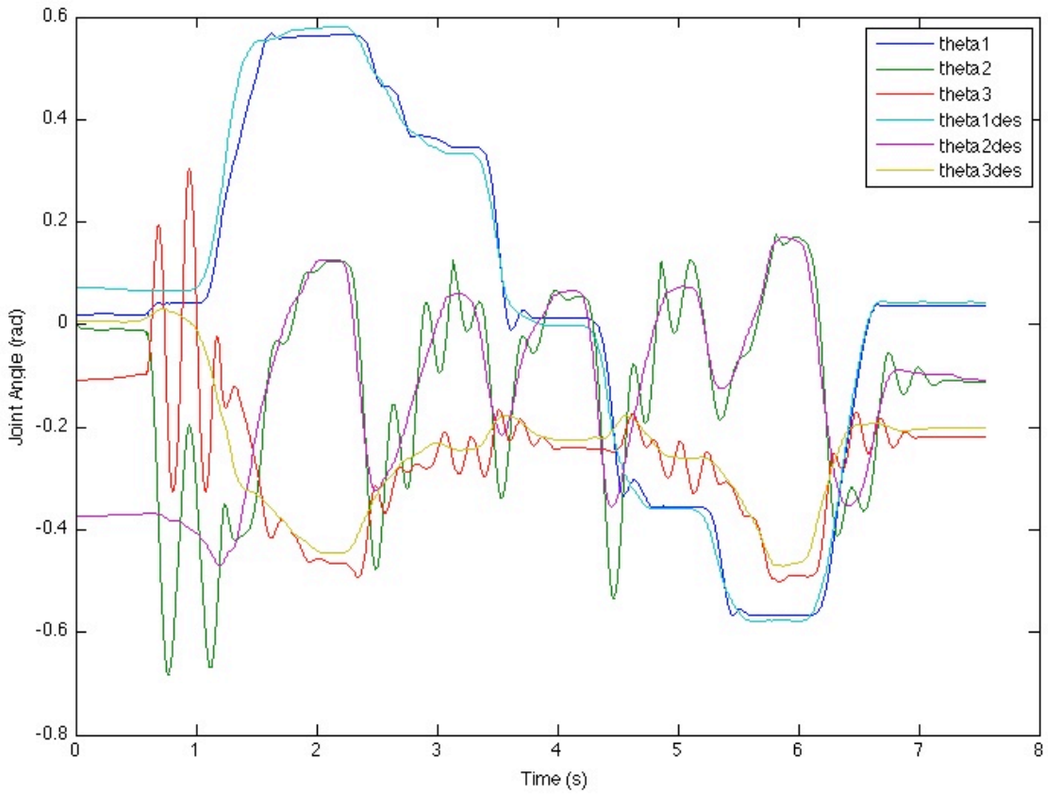


$$\hat{v} = \frac{x(t_2) - x(t_1)}{t_2 - t_1}$$

will be noisy because the position signal is quantized, so use a digital low-pass filter

$$v_{\mathrm{smooth},k} = w \cdot \hat{v}_k + (1 - w) \cdot v_{\mathrm{smooth},k-1}$$
$$0 < w < 1$$

Replaying Taps with Only Proportional Feedback (Virtual Spring)

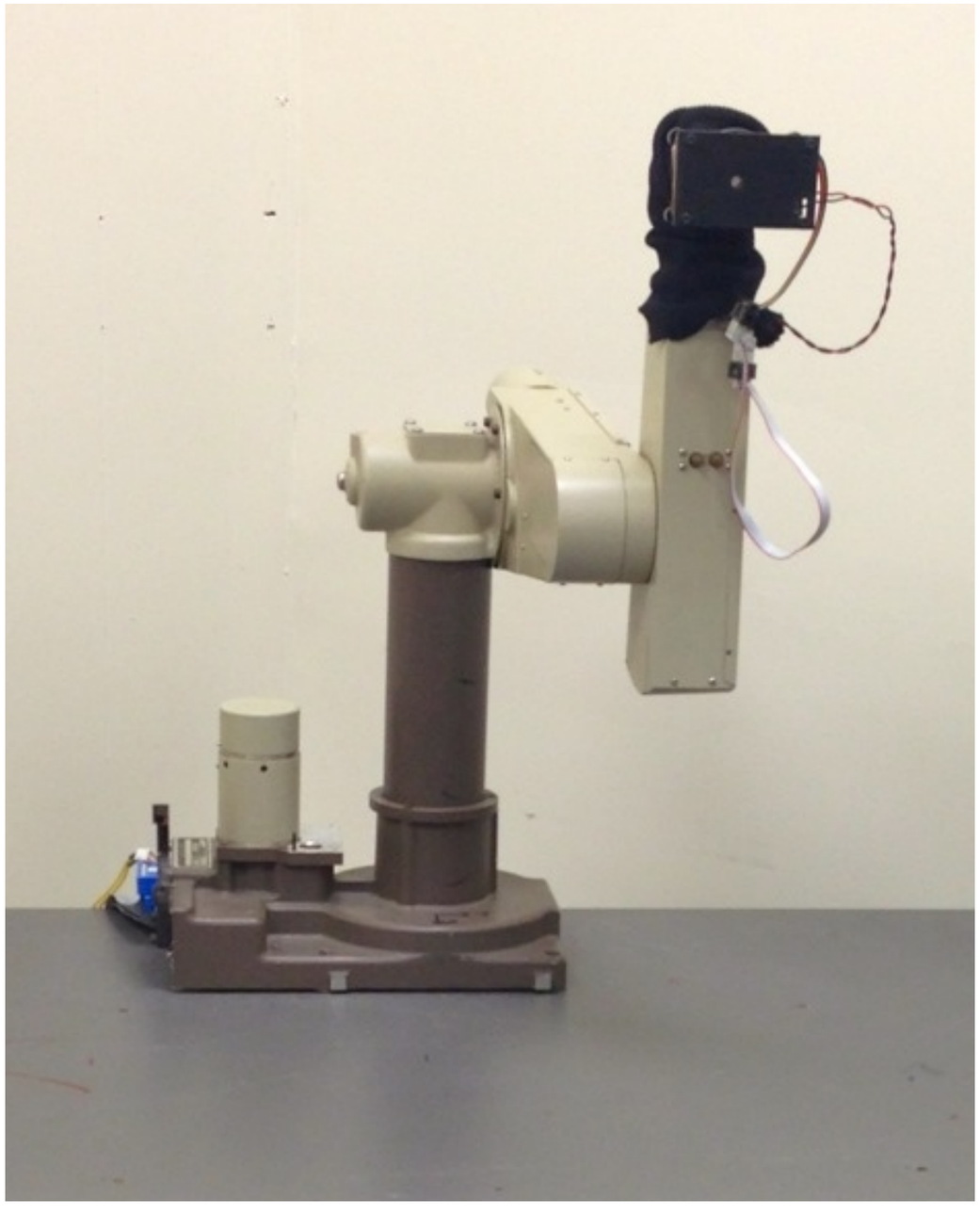


Second and third joints are highly oscillatory.

Replaying Taps with Proportional and Derivative Feedback (Virtual Spring plus Virtual Damper)

Now the second and third joints track pretty well.

Compare and Contrast:
PUMA Controller vs. Phantom Controller.



How are they similar? How do they differ?
Why? Talk with a partner or two.

Compare and Contrast:
PUMA Controller vs. Phantom Controller.

The two robots have different dynamics. The PUMA has much more friction than the Phantom. Why? Because of the gears; sliding surfaces always waste energy. The gears give the PUMA good mechanical advantage, but they cause a lot of friction. The Phantom is much more backdrivable than the PUMA. The joints of the Phantom are also less coupled than those of the PUMA, due to the parallel vs. serial linkage. The PUMA has six joints, so if we're thinking about only position output, it's redundant, so the linear velocity Jacobian isn't square. Could think only about the position of the wrist or the tip with the wrist rigid.

While the Phantom controller is a Cartesian PD controller, the PUMA has a separate PD controller on each joint. If we tried to run the Phantom's controller on the PUMA, the friction in the joints would probably make the robot move the wrong way. And we'd have no control over which IK solution it chose.

Replaying Taps with Proportional and Derivative Feedback (Virtual Spring plus Virtual Damper)

Now the second and third joints track pretty well.

The inertia, weight, and friction of the robot all interfere with tracking.

The inertia, weight, and friction of the robot all interfere with tracking.

Weight is easy to anticipate and cancel!

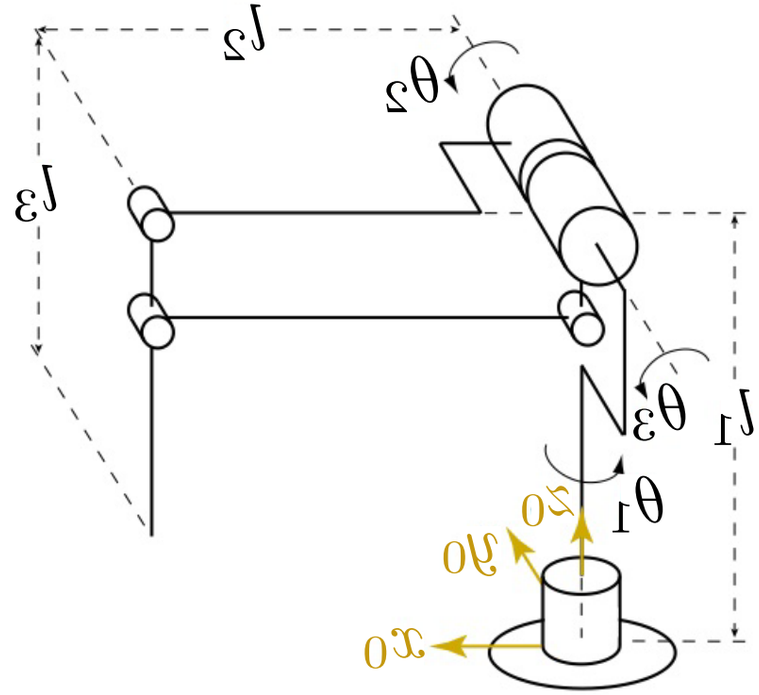
Robot designers generally try to **minimize the inertia (mass & mass distribution)** of the robot so it can accelerate more quickly and be less affected by gravity.

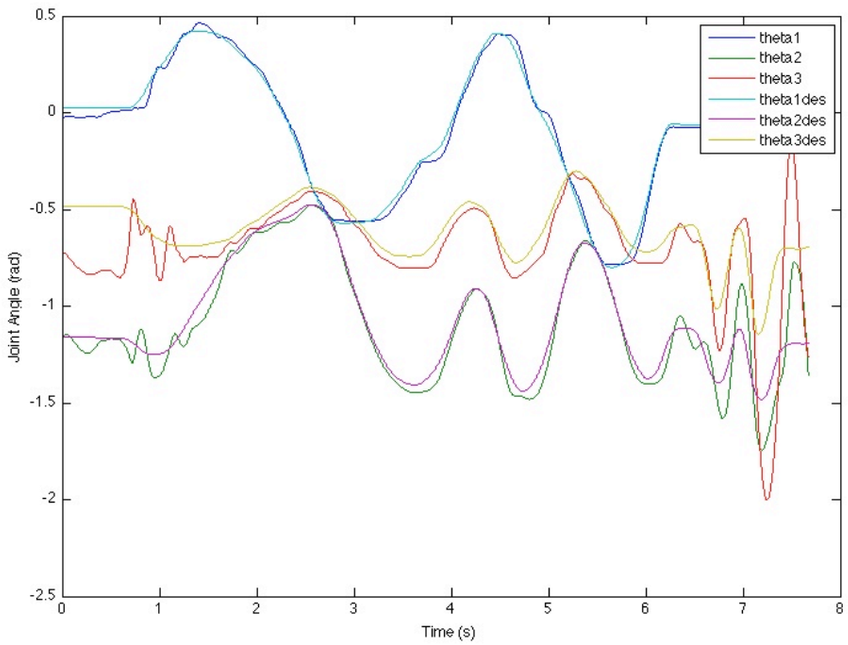
Similarly, robot designers try to **minimize the friction** of the robot so that the start of motion is smooth and sustained motion doesn't require much torque.

It is generally difficult to use feed-forward control to anticipate and cancel out inertia and friction.

How to implement gravity compensation in software?

Joints 2 and 3 are affected by gravity.





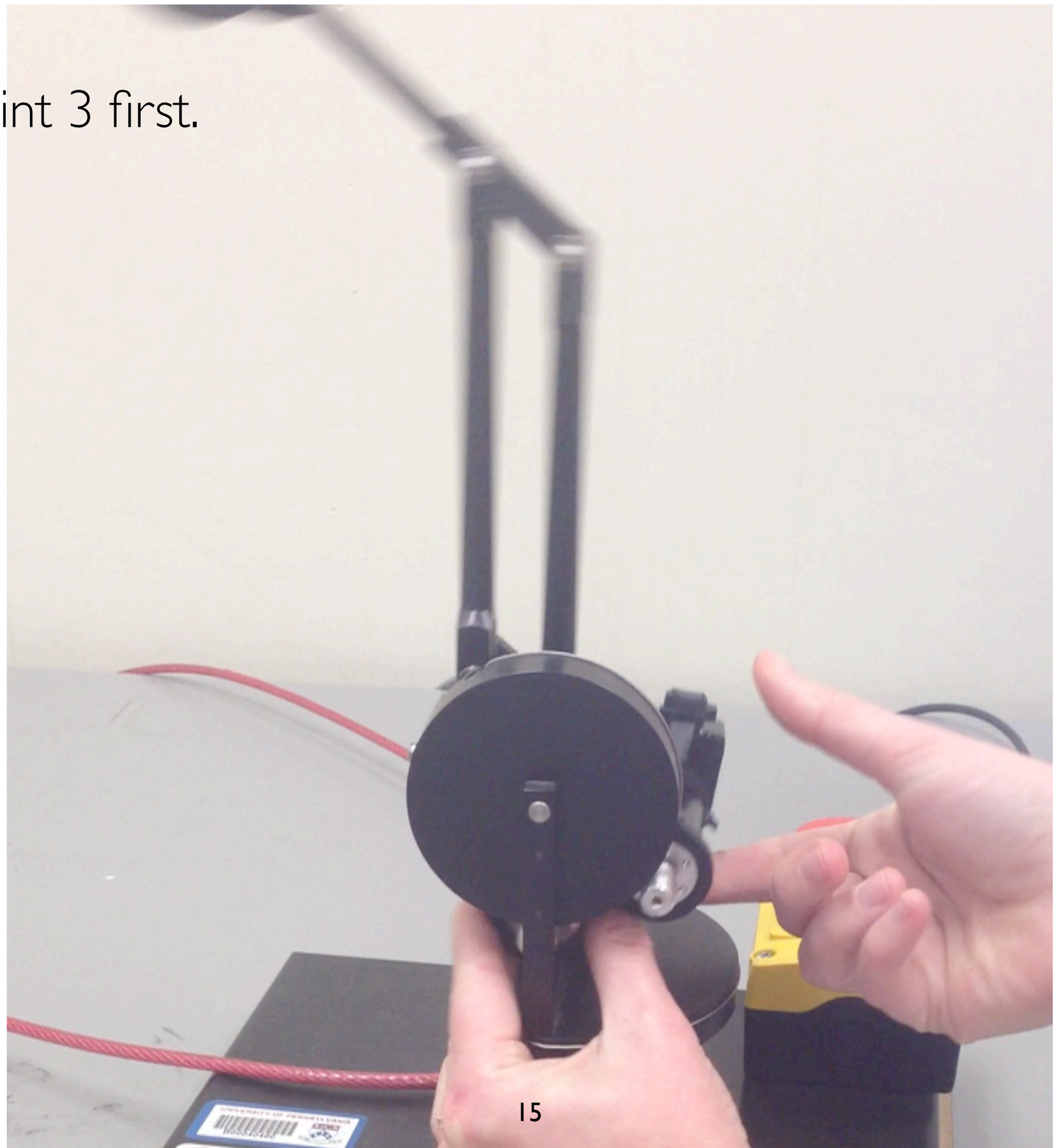
How do we calibrate gravity compensation?

Option 1: Inspect the robot to determine the form we expect for the gravity torque, then tune the strength through experiments.

Option 2: Move the robot slowly through a trajectory and record the torque needed to hold up the weight of the robot.

(Very common to use a feedback controller to calibrate a feedforward controller.)

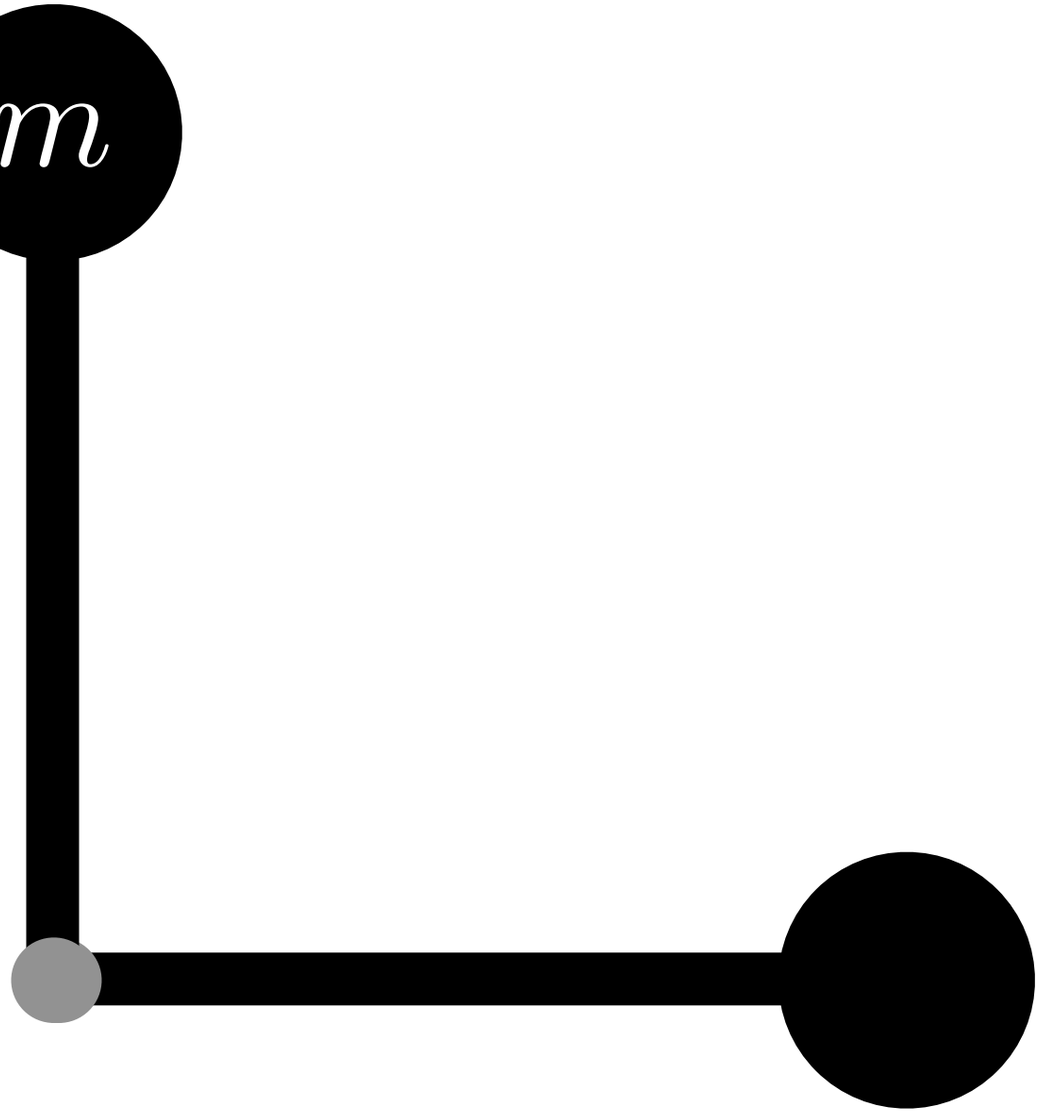
Option 1
Look at Joint 3 first.

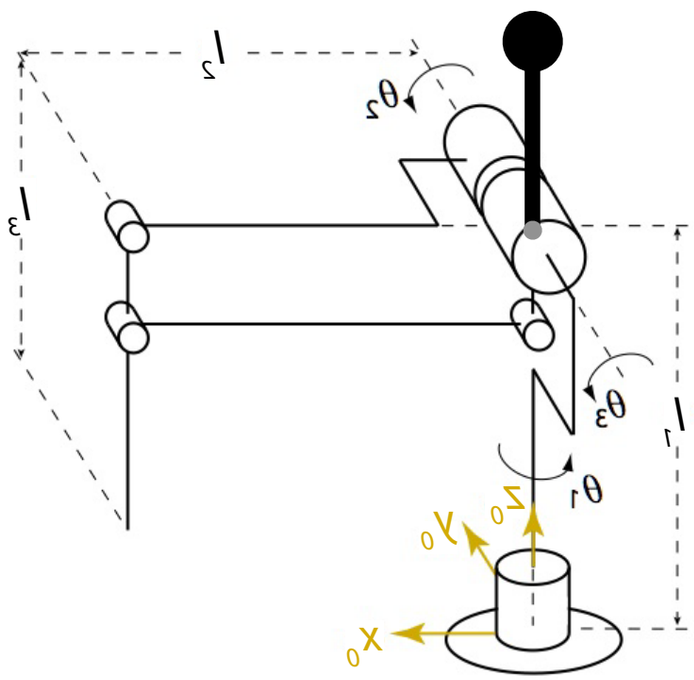


$\theta_3 = 0$

m

g



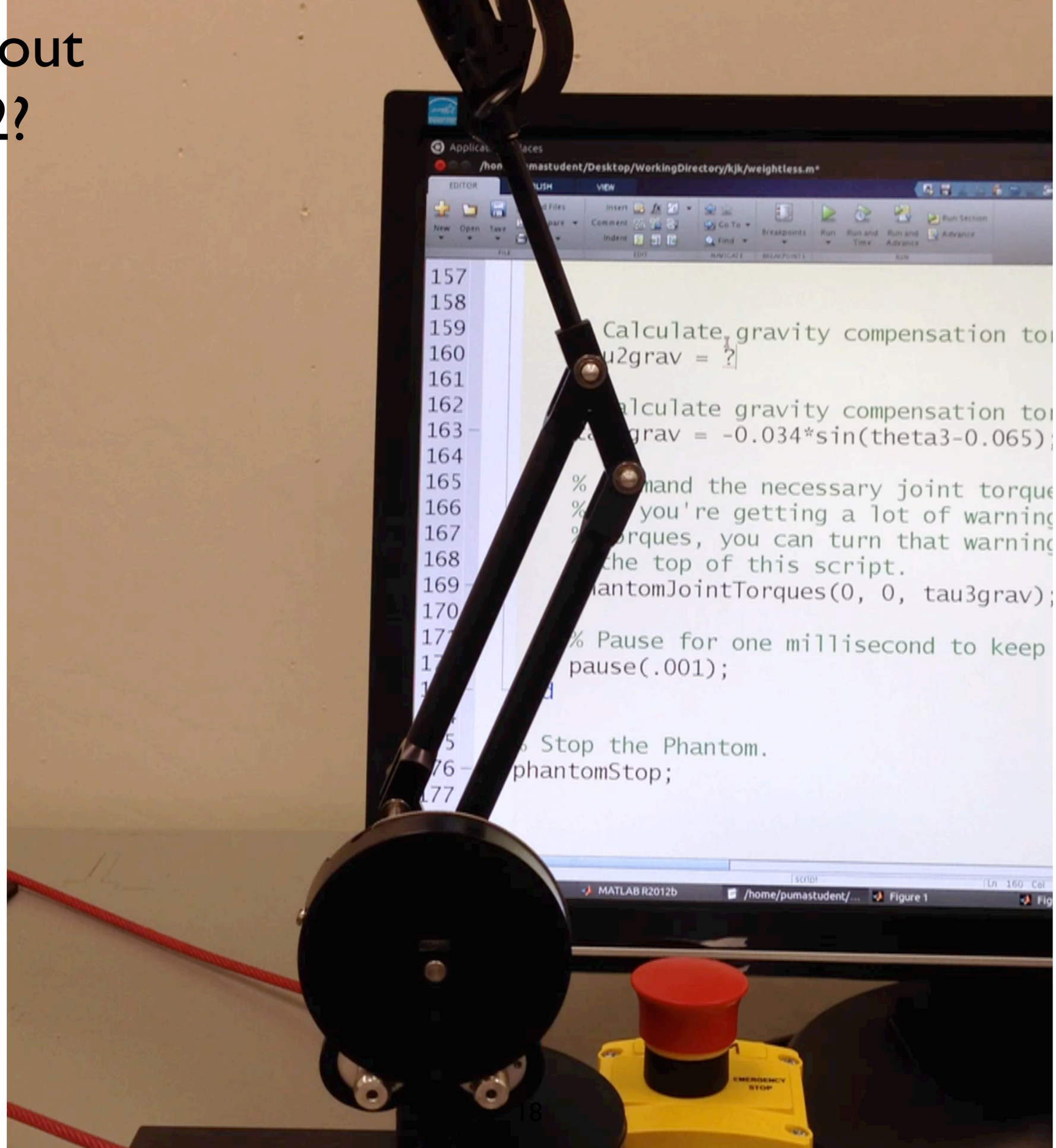


$$\theta_3 = -\pi/2 = -1.57$$

Expect gravity torque to be positive in value and involve -sin(theta3)

with gravity compensation on joint 3

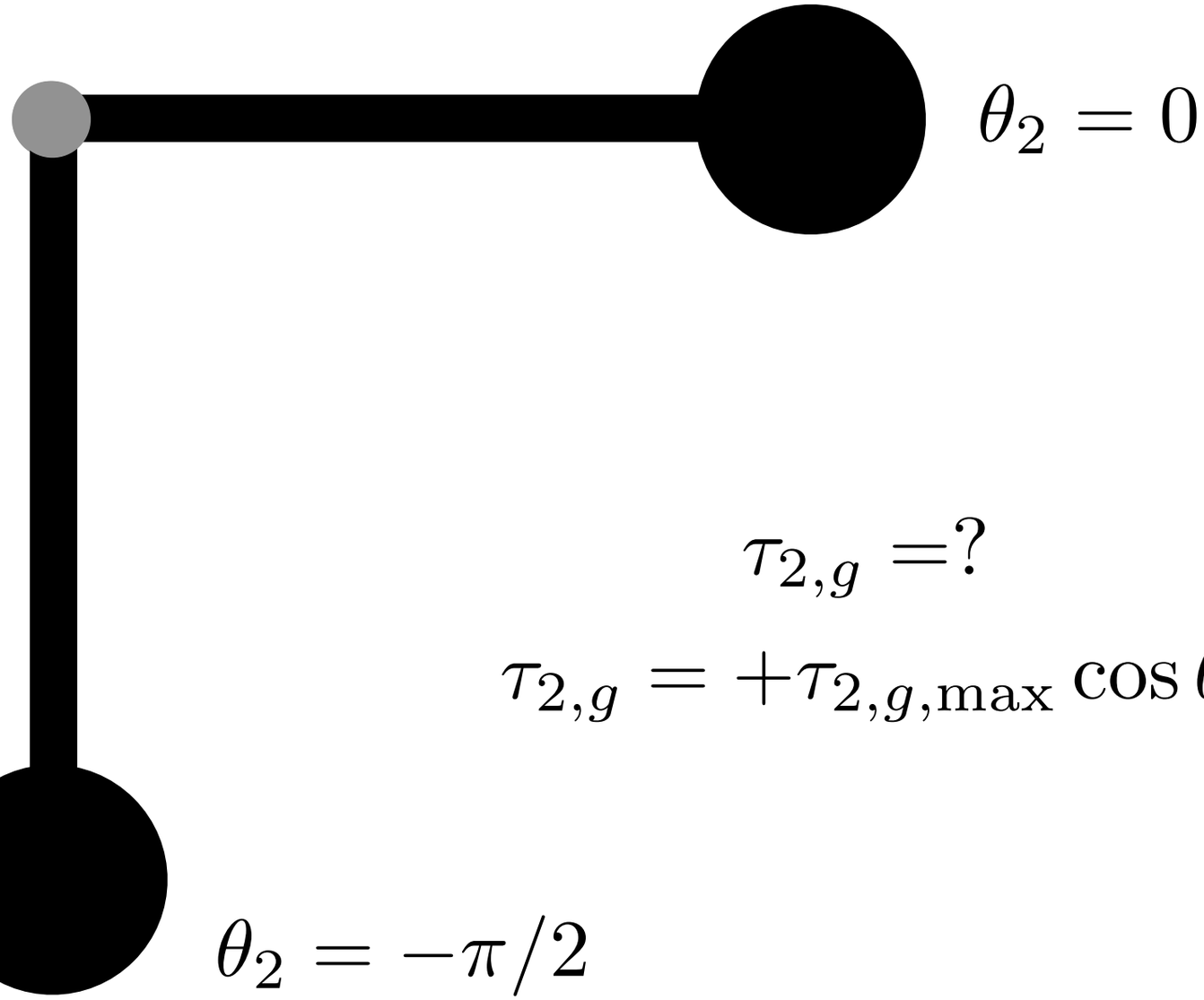
What about joint 2?



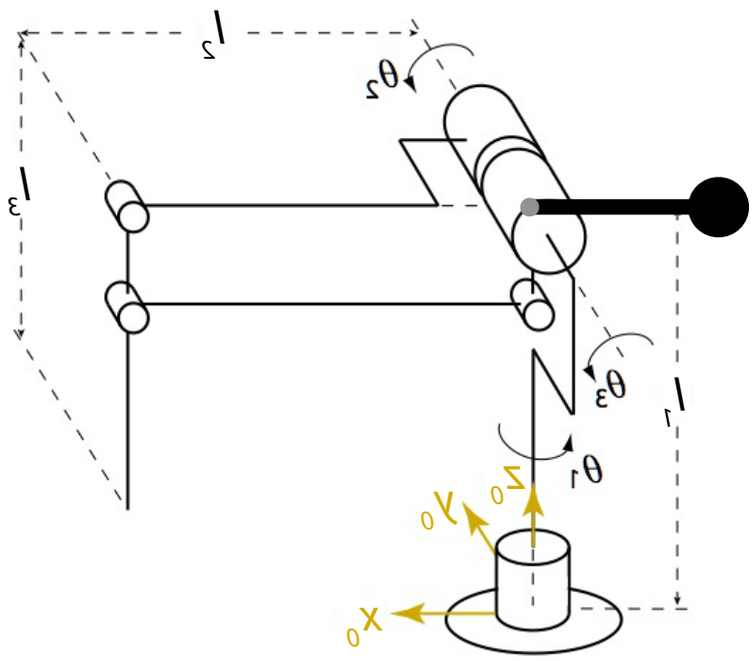
What form do you expect the gravity compensation for joint 2 to take?

Expect gravity torque to be positive in value and involve cos(theta2)

g



$\theta_2 = 0$



$\tau_{2,g} = ?$

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

$\theta_2 = -\pi/2$

Tuning gravity compensation on joint 3

How do we calibrate gravity compensation?

Option 1: Inspect the robot to determine the form we expect for the gravity torque, then tune the strength through experiments.

Option 2: Move the robot slowly through a trajectory and record the torque needed to hold up the weight of the robot.

(Very common to use a feedback controller to calibrate a feedforward controller.)

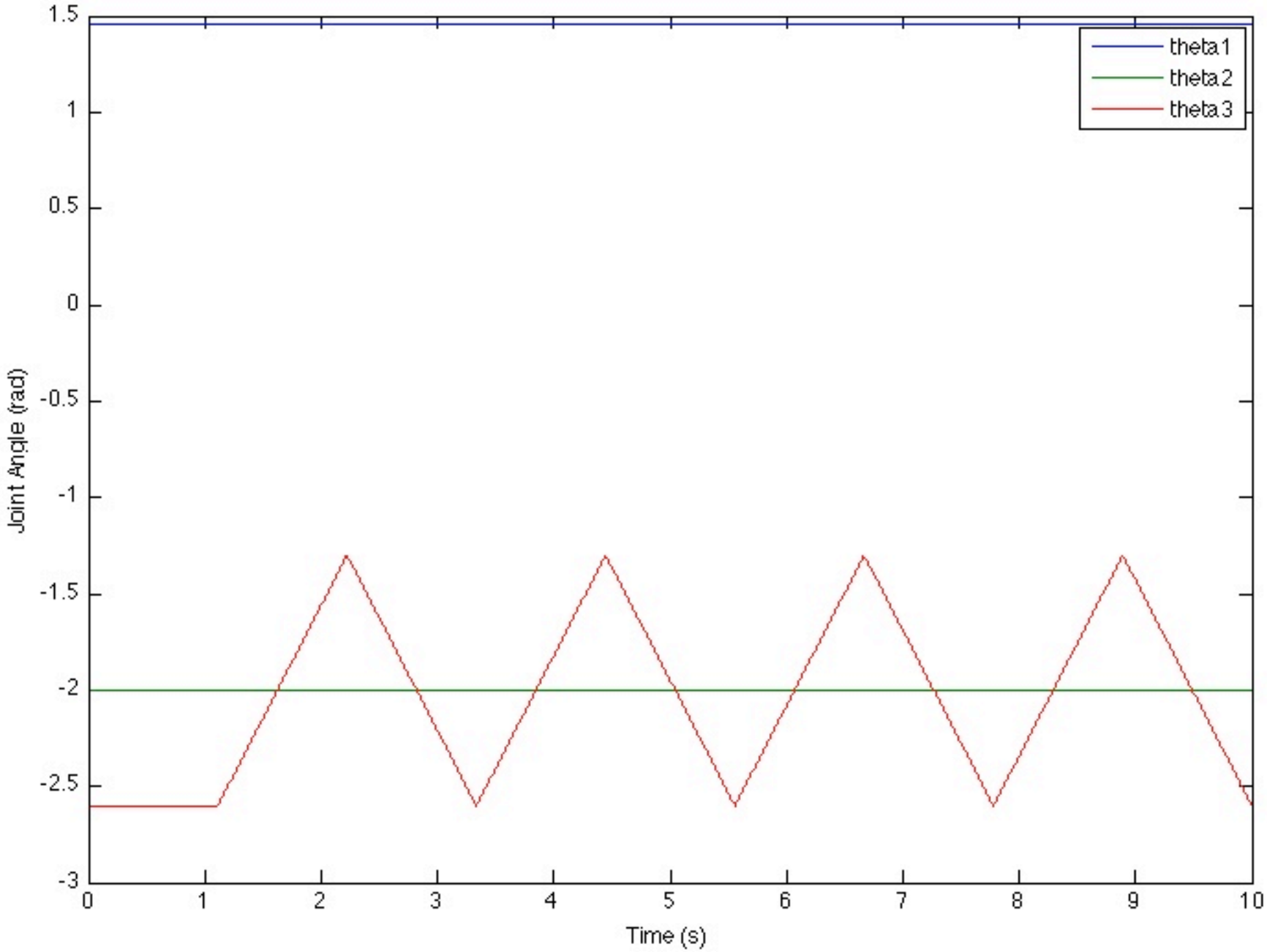
Editor - /Users/kuchenbe/Desktop/kjk/make_theta3_v1.m

File Edit Text Go Cell Tools Debug Desktop Window Help

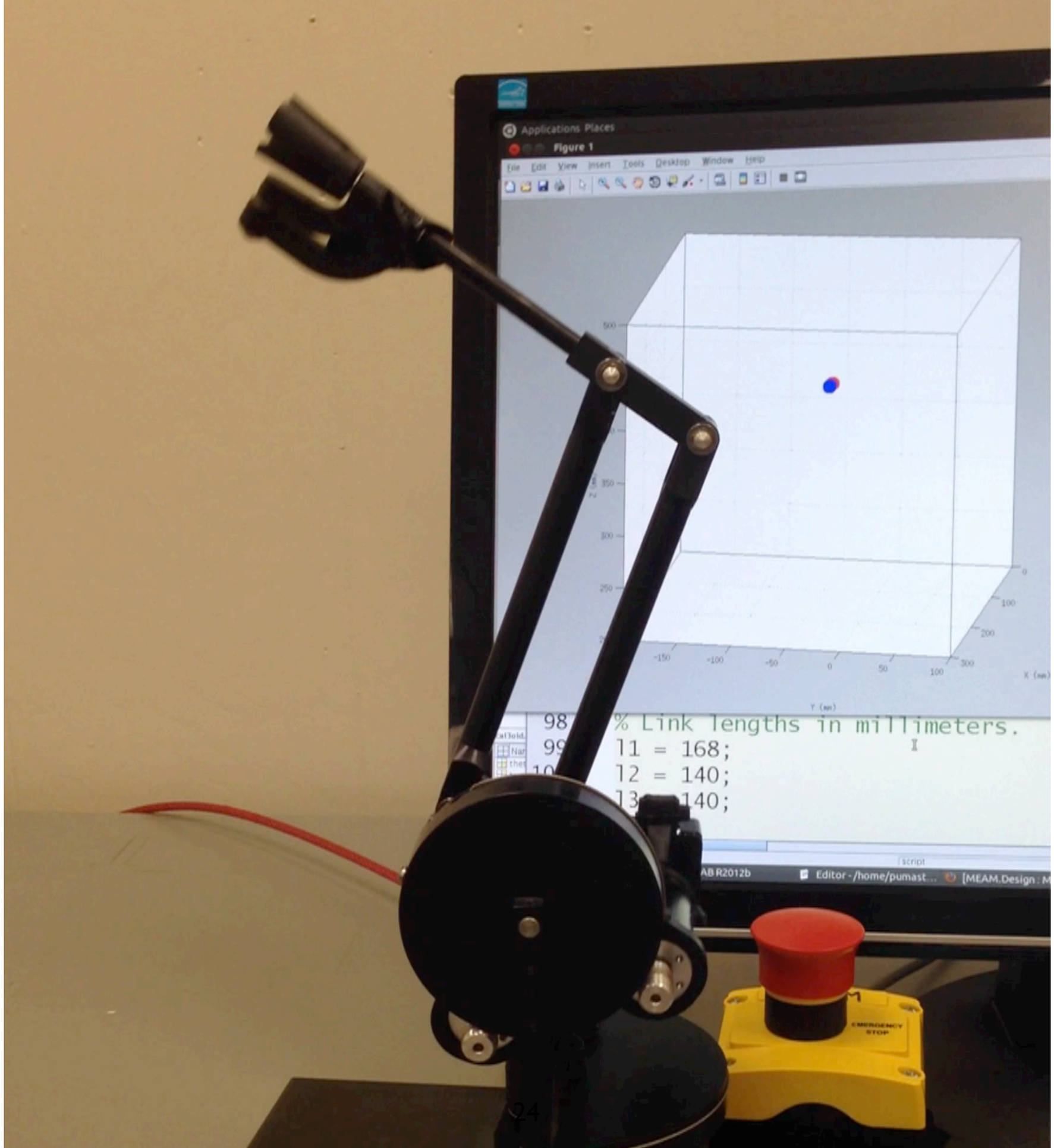
```
 1      % Make t.
 2 -    t = linspace(0,10,9000)';
 3
 4      % Make all thetas.
 5 -    thetas = zeros(9000,3);
 6 -    thetas(:,1) = 1.46;
 7 -    thetas(:,2) = -2;
 8 -    thetas(:,3) = -2.6;
 9 -    thetas(1001:2000,3) = linspace(-2.6,-1.3,1000)';
10 -    thetas(2001:3000,3) = linspace(-1.3,-2.6,1000)';
11 -    thetas(3001:4000,3) = linspace(-2.6,-1.3,1000)';
12 -    thetas(4001:5000,3) = linspace(-1.3,-2.6,1000)';
13 -    thetas(5001:6000,3) = linspace(-2.6,-1.3,1000)';
14 -    thetas(6001:7000,3) = linspace(-1.3,-2.6,1000)';
15 -    thetas(7001:8000,3) = linspace(-2.6,-1.3,1000)';
16 -    thetas(8001:9000,3) = linspace(-1.3,-2.6,1000)';
17
18     % Plot thetas.
19 -    figure(6);
20 -    plot(t,thetas)
21
```

ox_demo.m | haptic_ball_team50.m | haptic_damping_team50.m | cmdhist.m | make_theta3_v1.m

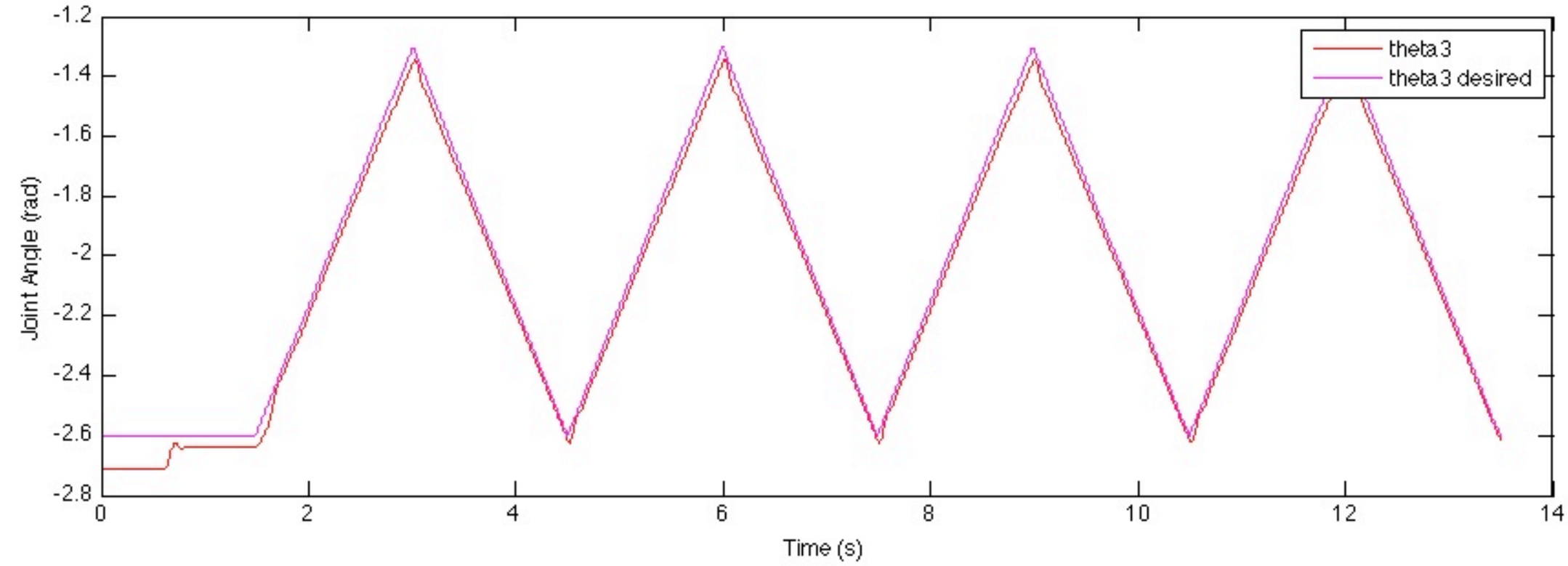
script Ln 17 Col 1

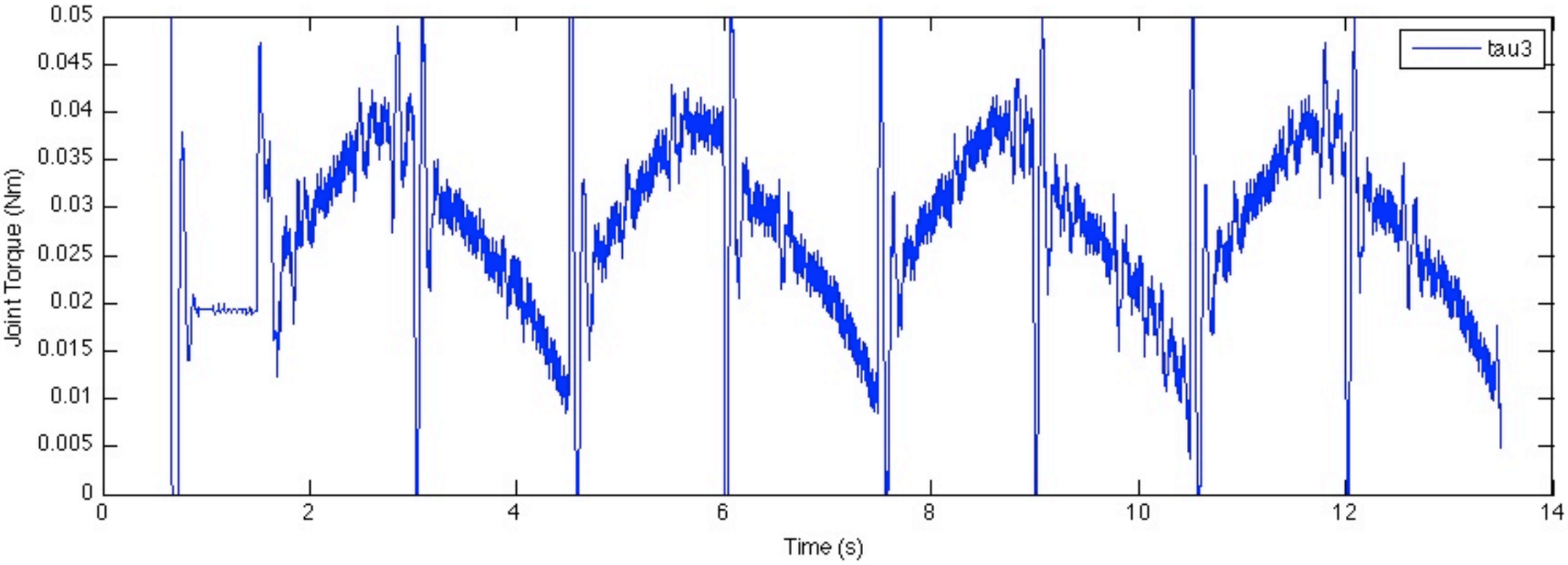

theta1
theta2
theta3
Joint Angle (rad)
Time (s)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
1
2
3
4
5
6
7
8
9
10

cal3


Applications Places
Figure 1
File Edit View Insert Tools Desktop Window Help
% Link lengths in millimeters.
l1 = 168;
l2 = 140;
l3 = 140;
script
AB R2012b
Editor - /home/pumast...
[MEAM.Design : M
EMERGENCY STOP

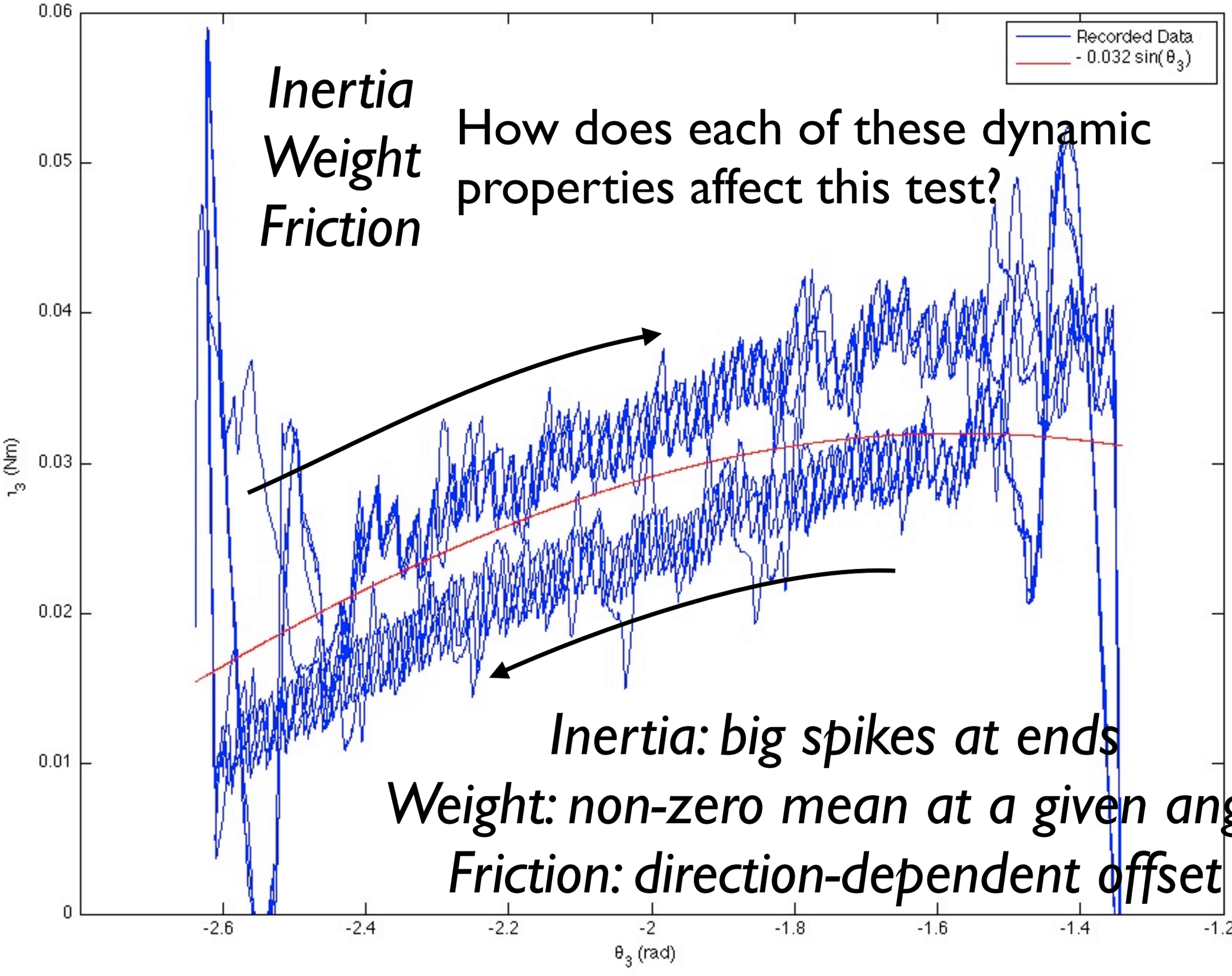
cal3


theta3
theta3 desired
Joint Angle (rad)
-1.2
-1.4
-1.6
-1.8
-2
-2.2
-2.4
-2.6
-2.8
0
2
4
6
8
10
12
14
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
0
2
4
6
8
10
12
14
Time (s)

cal3

How can we make this better?



Inertia
Weight
Friction

How does each of these dynamic properties affect this test?

Inertia: big spikes at ends

Weight: non-zero mean at a given angle

Friction: direction-dependent offset

Trajectory Generation

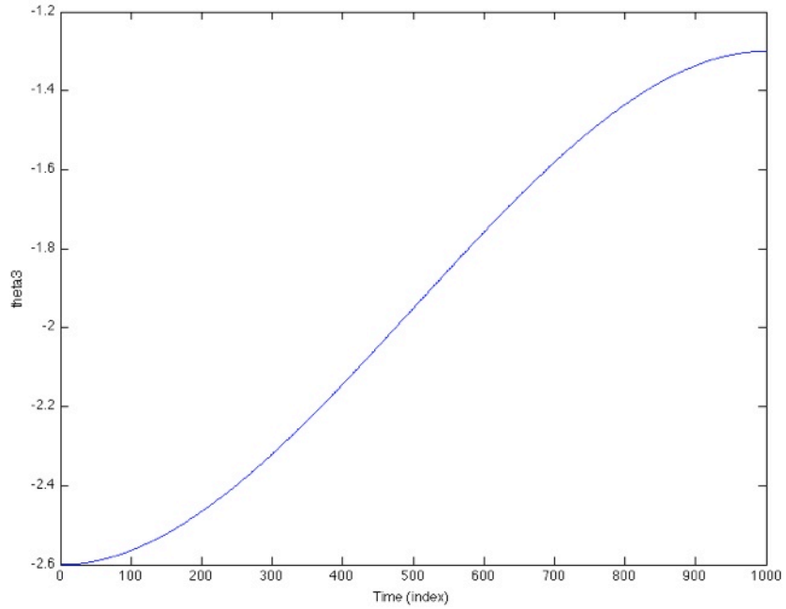
start → end

$$q(t_0) = q_0 \longrightarrow q(t_f) = q_f$$

$$\dot{q}(t_0) = v_0 \longrightarrow \dot{q}(t_f) = v_f$$

cubic polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$



$$\begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 \\ 0 & 1 & 2t_0 & 3{t_0}^2 \\ 1 & t_f & {t_f}^2 & {t_f}^3 \\ 0 & 1 & 2t_f & 3{t_f}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Editor - /Users/kuchenbe/Desktop/kjk/make_theta3_v3.m

File Edit Text Go Cell Tools Debug Desktop Window Help

```
% Make an abbreviated time vector with 1000 elements.
t = 1:1000;

% Define initial conditions.
q0 = -2.6;
v0 = 0;

% Define final conditions.
qf = -1.3;
vf = 0;

% Put initial and final conditions into a vector.
conditions = [q0; v0; qf; vf];

% Put time elements into matrix.
mat = [1    t(1)     t(1)^2       t(1)^3;
       0       1     2*t(1)     3*t(1)^2;
       1  t(end)   t(end)^2     t(end)^3;
       0       1   2*t(end)   3*t(end)^2];

% Solve for coefficients.
as = mat \ conditions;

% Pull individual coefficients out.
a0 = as(1);
a1 = as(2);
a2 = as(3);
a3 = as(4);
```

ox_demo.m | haptic_ball_team50.m | haptic_damping_team50.m | cmdhist.m | make_theta3_v3.m

script Ln 23 Col 1

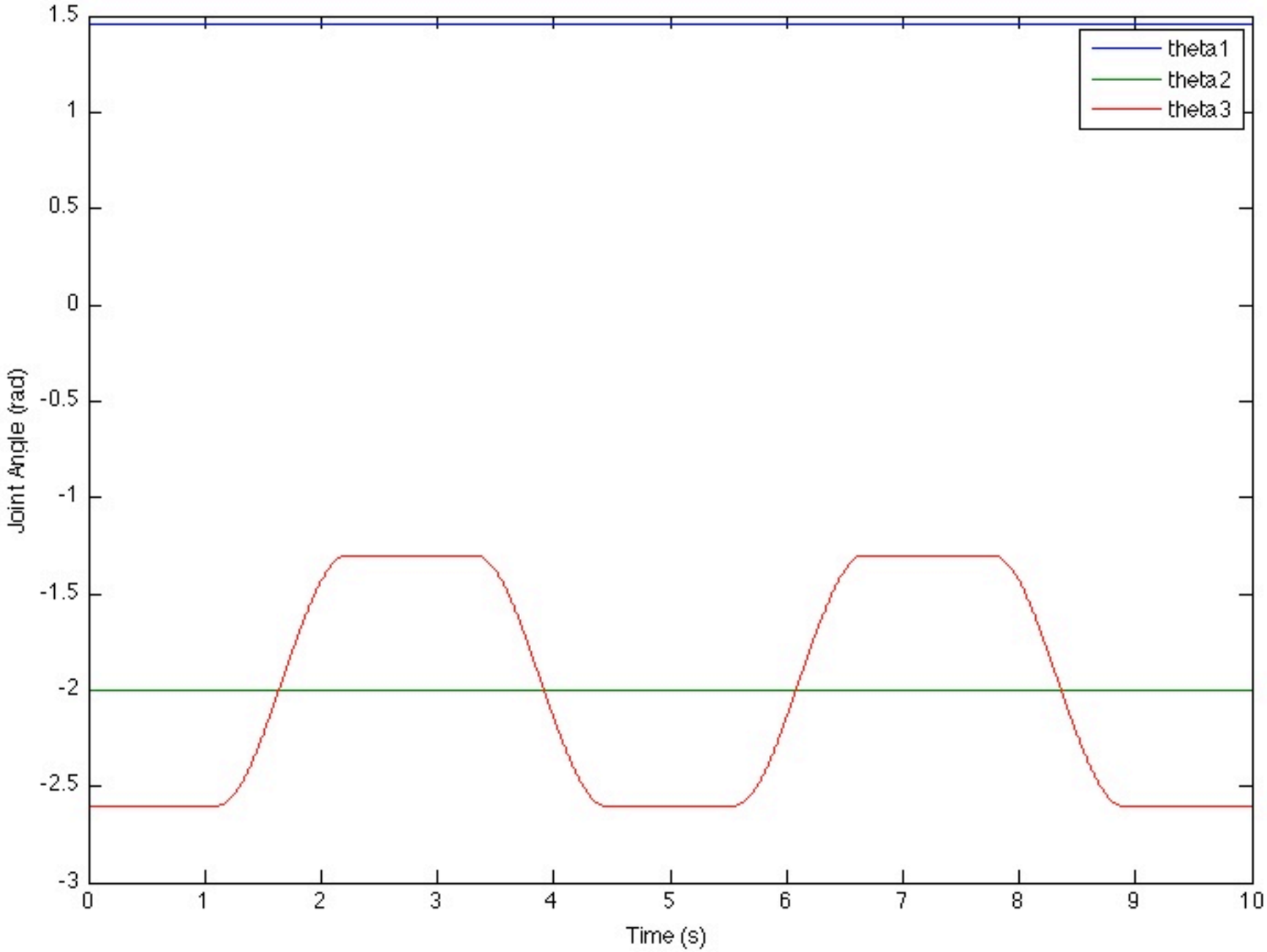
Editor - /Users/kuchenbe/Desktop/kjk/make_theta3_v3.m

File Edit Text Go Cell Tools Debug Desktop Window Help

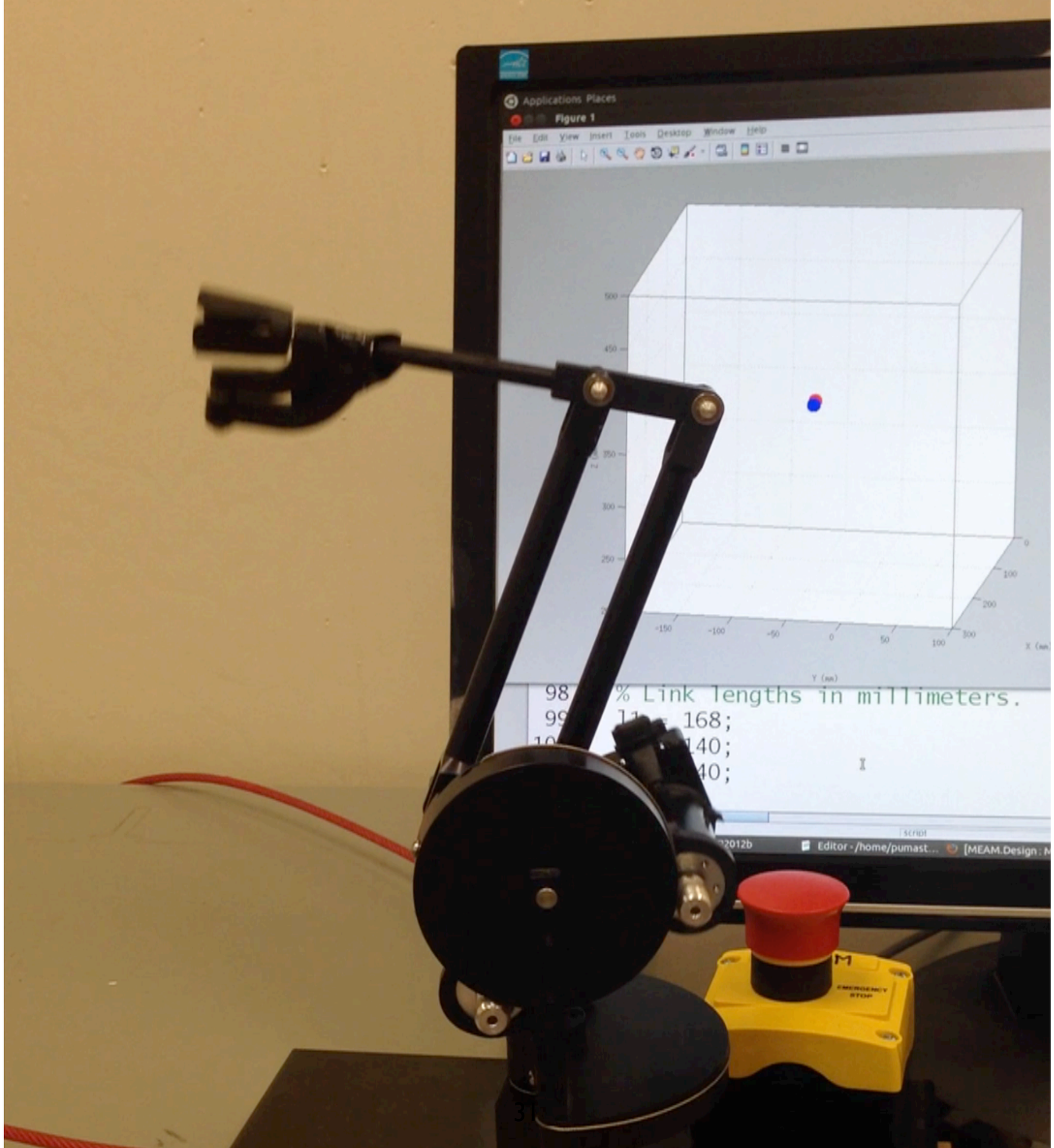
```
34 -    figure(2)
35 -    plot(t,q)
36 -    xlabel('Time (index)')
37 -    ylabel('theta3')
38
39
40      % Make t.
41 -    t = linspace(0,10,9000)';
42
43      % Make all thetas.
44 -    thetas = zeros(9000,3);
45 -    thetas(:,1) = 1.46;
46 -    thetas(:,2) = -2;
47 -    thetas(:,3) = -2.6;
48 -    thetas(1001:2000,3) = q';
49 -    thetas(2001:3000,3) = -1.3;
50 -    thetas(3001:4000,3) = flipud(q');
51 -    thetas(4001:5000,3) = -2.6;
52 -    thetas(5001:6000,3) = q';
53 -    thetas(6001:7000,3) = -1.3;
54 -    thetas(7001:8000,3) = flipud(q');
55 -    thetas(8001:9000,3) = -2.6;
56
57      % Plot thetas.
58 -    figure(6);
59 -    plot(t,thetas)
60 -    xlabel('Time (s)')
61 -    ylabel('Joint Angle (rad)')
62 -    legend('theta1', 'theta2', 'theta3')
```

ic_box_demo.m | haptic_ball_team50.m | haptic_damping_team50.m | cmdhist.m | make_theta3_v3.m

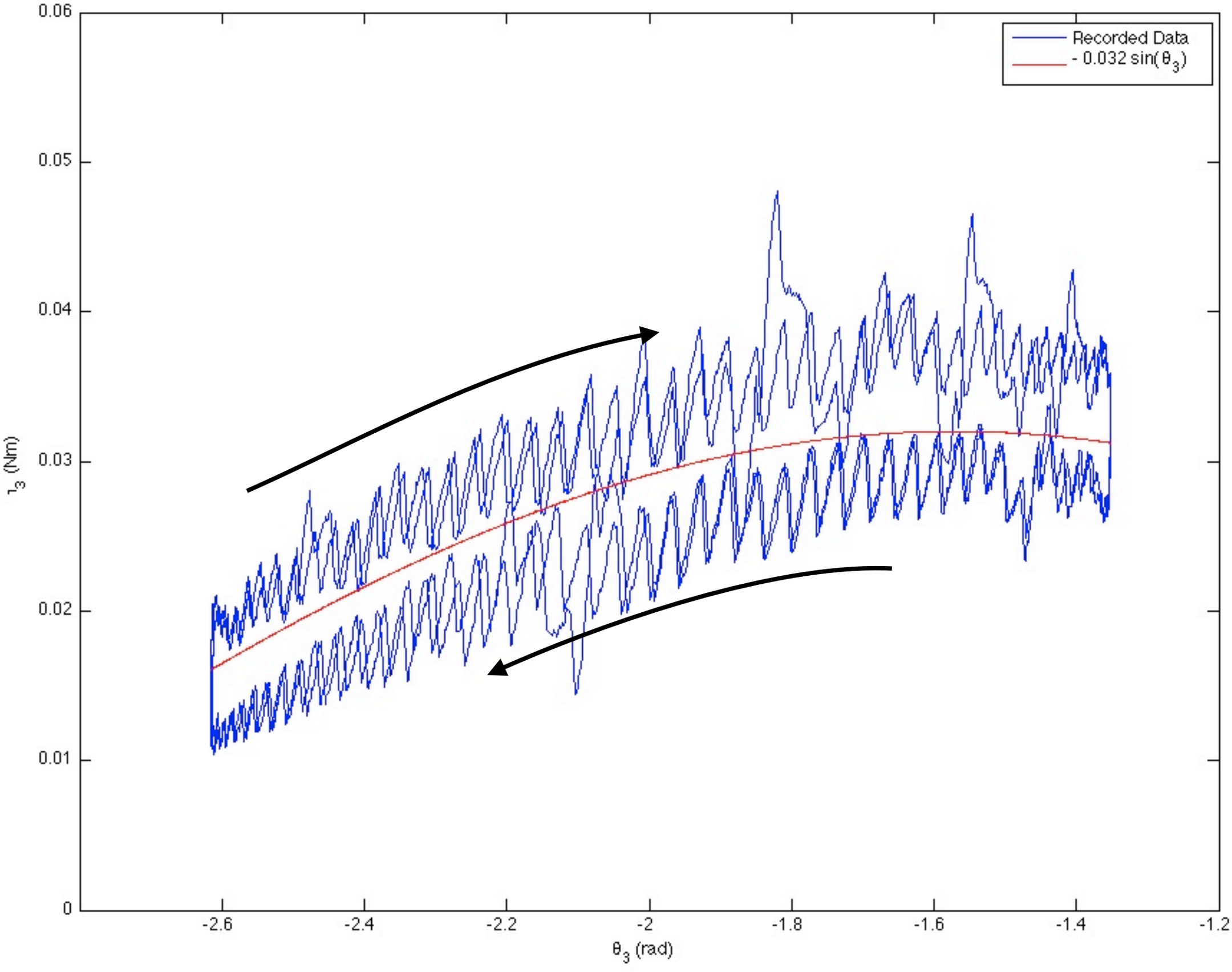
script Ln 46 Col 18


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

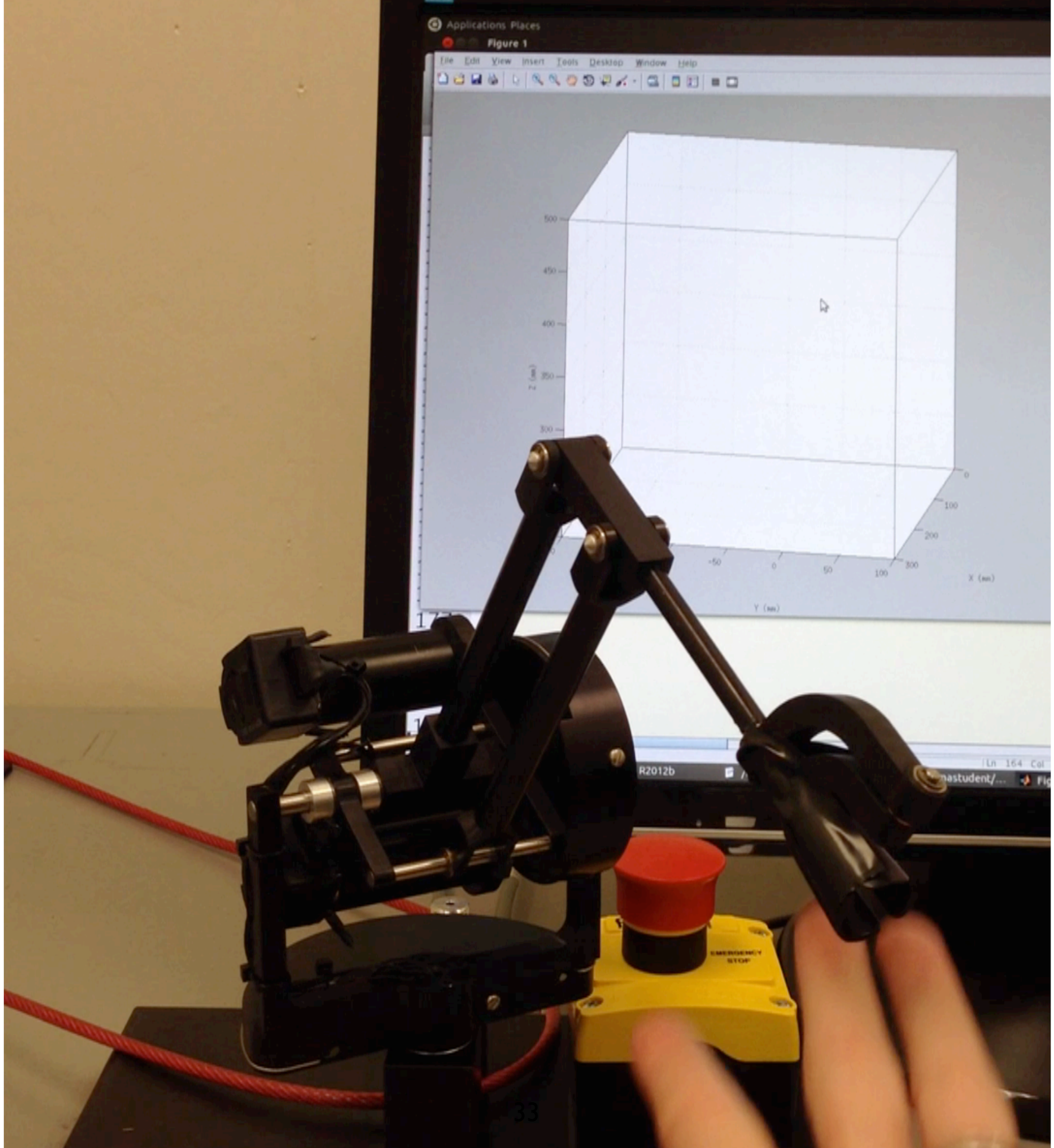
cal3newnew


Applications Places
Figure 1
File Edit View Insert Tools Desktop Window Help
98 % Link lengths in millimeters.
168;
140;
40;
Editor - /home/pumast...
[MEAM.Design : M
EMERGENCY STOP

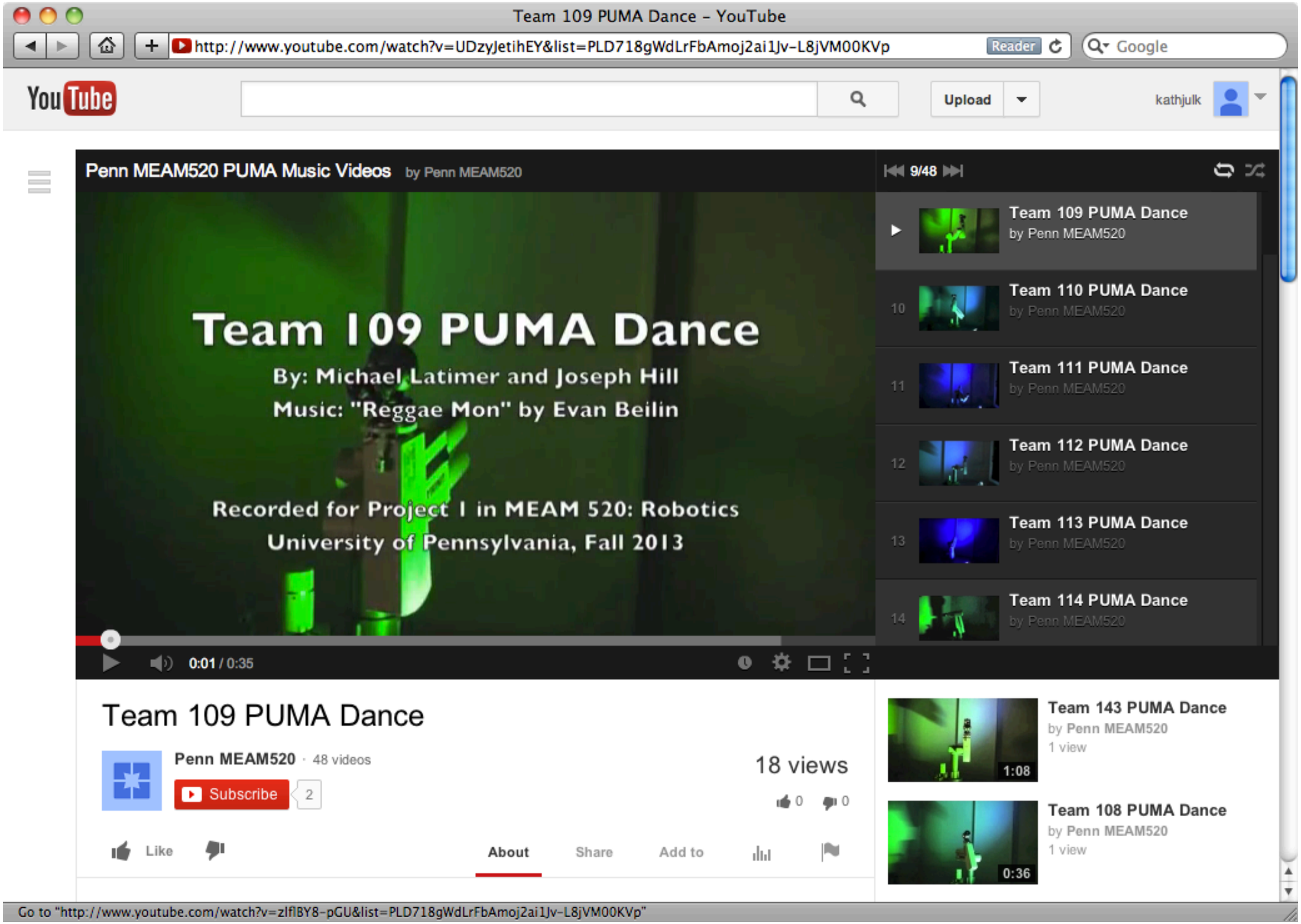
cal3newnew


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)

with updated gravity compensation on joints 2 and 3



Let's watch Teams 109–116.



Haptics

My definition of a haptic interface

Senses a physical quantity from the user, such as motion or force

Physically acts on the user via a variable actuator

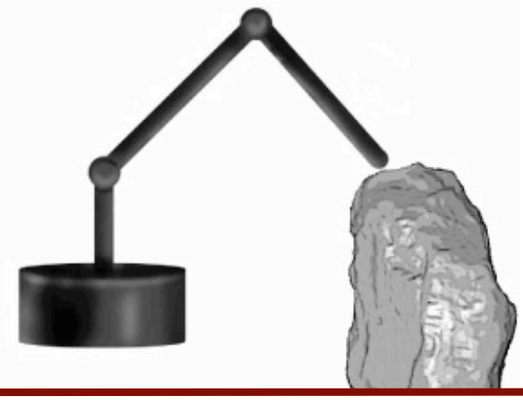
Connects sensing to acting with fast processing

(A lot like a robot, except a person is holding on to the tip.)

Environment

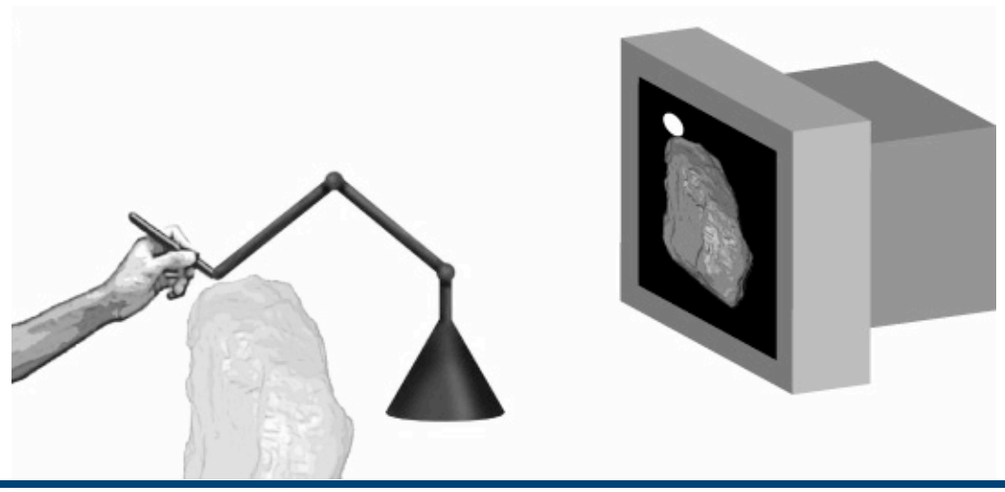
Real

Assistive Interaction: augments human sensing and/or motion capabilities in real physical environments



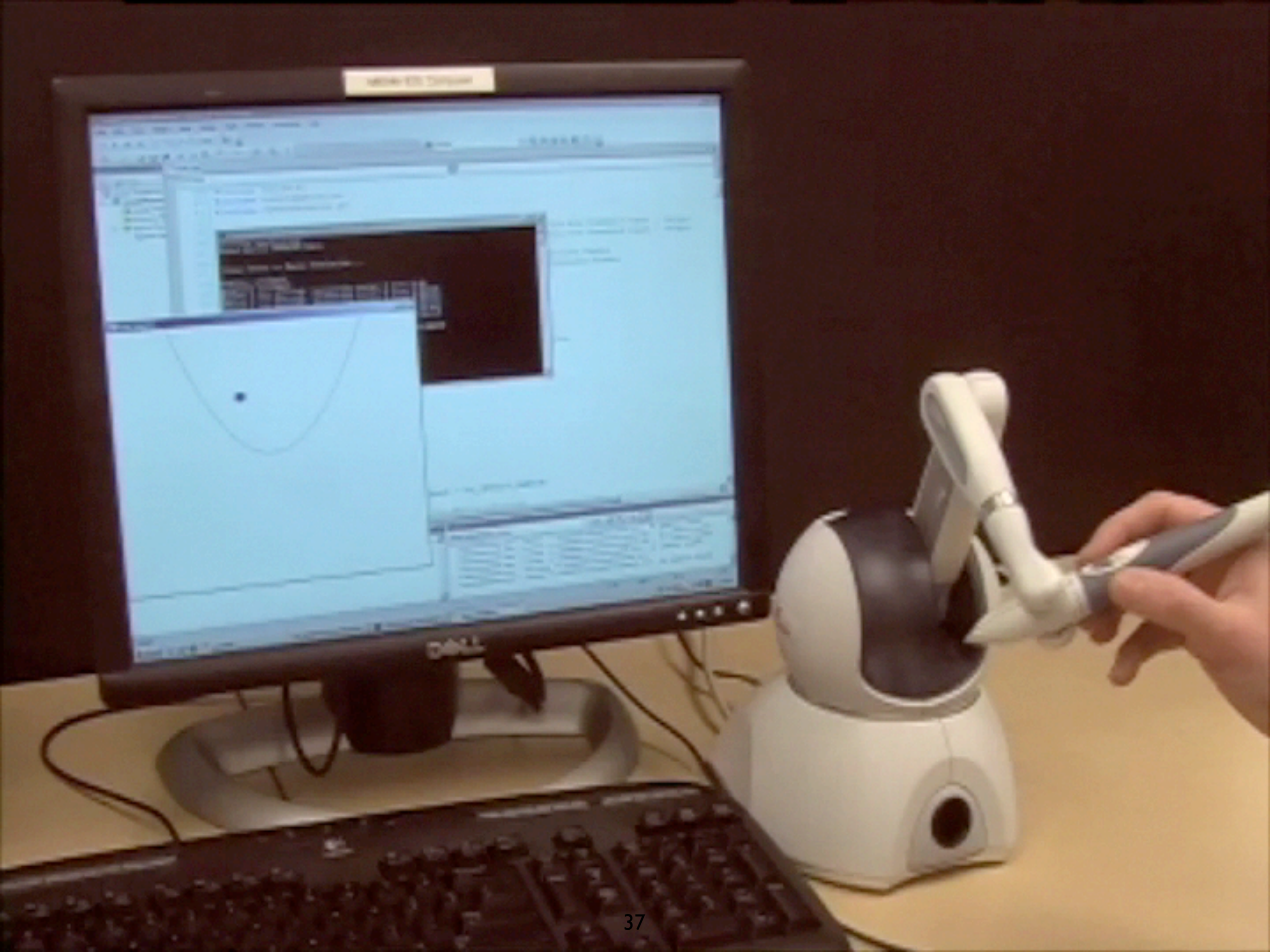
Remote

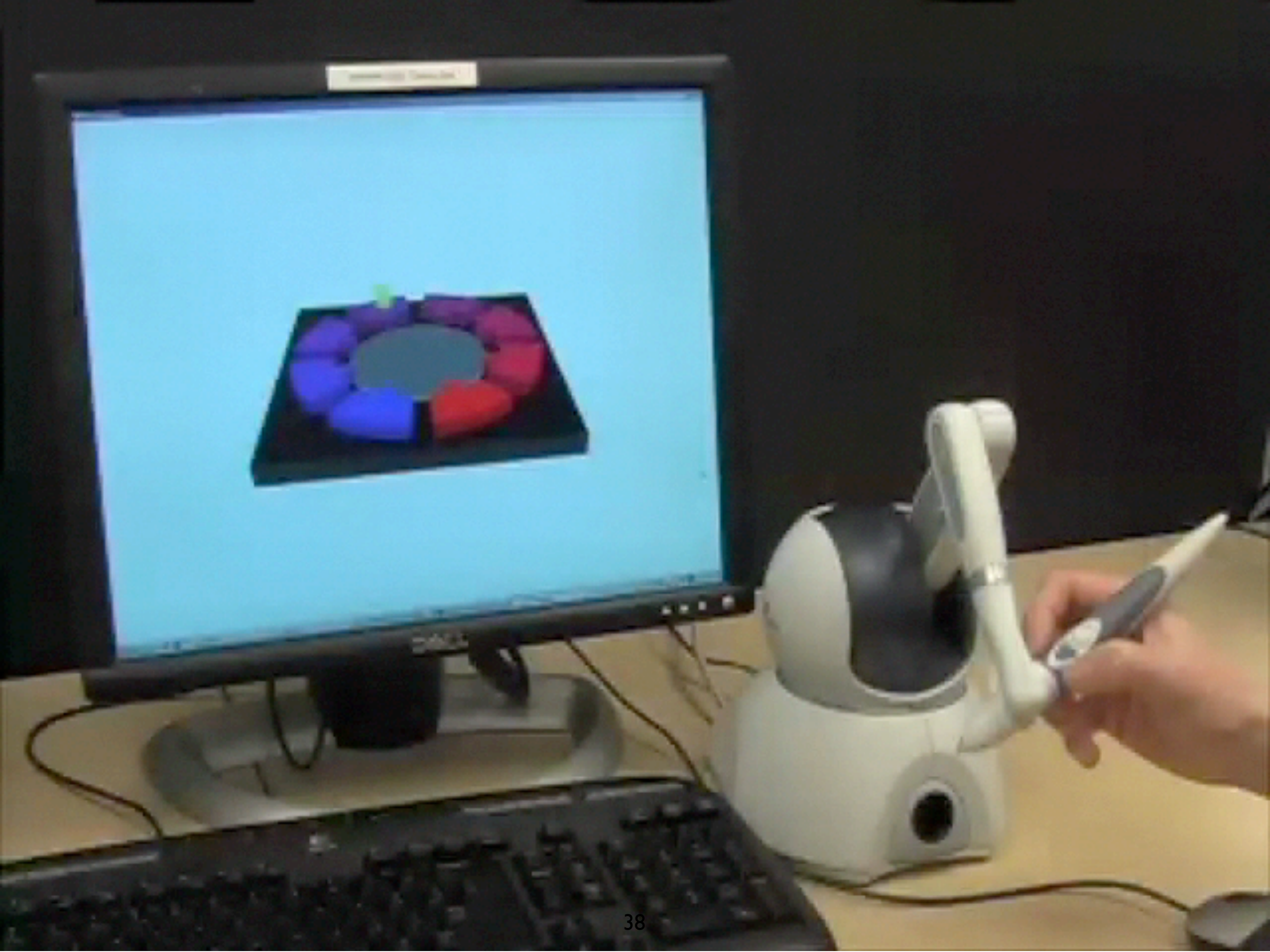
Teleoperation: extends the reach of the human hand to remote, hazardous, unreachable environments



Virtual

Simulation: enables humans to touch geometric and dynamic computer-based data and models






Without Acceleration Matching
With Acceleration Matching
world HAPTICS 2009

Commercial Kinesthetic Haptic Interfaces

SensAble Phantom Premiums

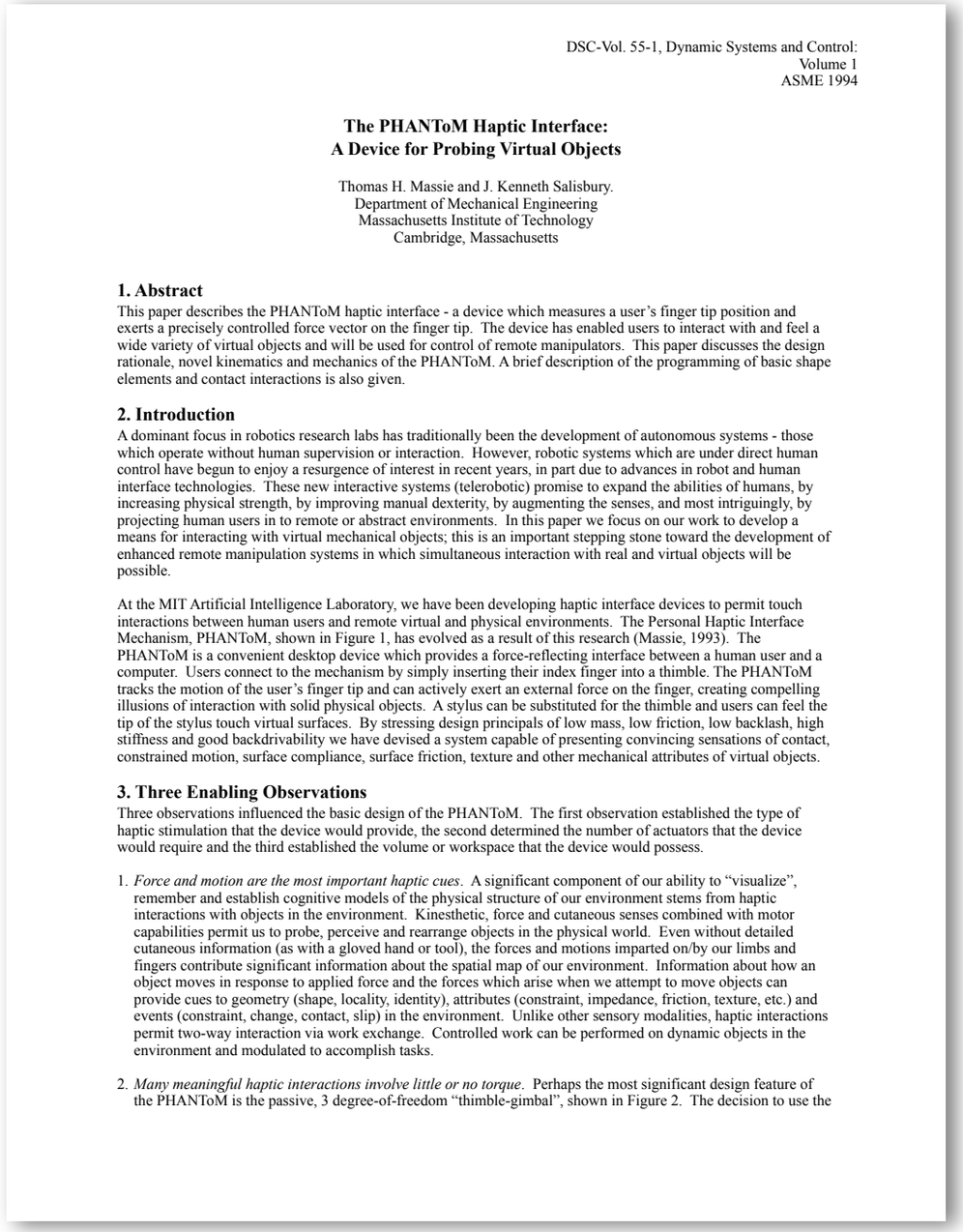
Force Dimension Omega

Immersion Impulse Engine

SensAble Omni

MPB Freedom6S

Novint Falcon



DSC-Vol. 55-1, Dynamic Systems and Control:
Volume 1
ASME 1994

**The PHANToM Haptic Interface:
A Device for Probing Virtual Objects**

Thomas H. Massie and J. Kenneth Salisbury.
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts

1. Abstract

This paper describes the PHANToM haptic interface - a device which measures a user's finger tip position and exerts a precisely controlled force vector on the finger tip. The device has enabled users to interact with and feel a wide variety of virtual objects and will be used for control of remote manipulators. This paper discusses the design rationale, novel kinematics and mechanics of the PHANToM. A brief description of the programming of basic shape elements and contact interactions is also given.

2. Introduction

A dominant focus in robotics research labs has traditionally been the development of autonomous systems - those which operate without human supervision or interaction. However, robotic systems which are under direct human control have begun to enjoy a resurgence of interest in recent years, in part due to advances in robot and human interface technologies. These new interactive systems (telerobotic) promise to expand the abilities of humans, by increasing physical strength, by improving manual dexterity, by augmenting the senses, and most intriguingly, by projecting human users in to remote or abstract environments. In this paper we focus on our work to develop a means for interacting with virtual mechanical objects; this is an important stepping stone toward the development of enhanced remote manipulation systems in which simultaneous interaction with real and virtual objects will be possible.

At the MIT Artificial Intelligence Laboratory, we have been developing haptic interface devices to permit touch interactions between human users and remote virtual and physical environments. The Personal Haptic Interface Mechanism, PHANToM, shown in Figure 1, has evolved as a result of this research (Massie, 1993). The PHANToM is a convenient desktop device which provides a force-reflecting interface between a human user and a computer. Users connect to the mechanism by simply inserting their index finger into a thimble. The PHANToM tracks the motion of the user's finger tip and can actively exert an external force on the finger, creating compelling illusions of interaction with solid physical objects. A stylus can be substituted for the thimble and users can feel the tip of the stylus touch virtual surfaces. By stressing design principals of low mass, low friction, low backlash, high stiffness and good backdrivability we have devised a system capable of presenting convincing sensations of contact, constrained motion, surface compliance, surface friction, texture and other mechanical attributes of virtual objects.

3. Three Enabling Observations

Three observations influenced the basic design of the PHANToM. The first observation established the type of haptic stimulation that the device would provide, the second determined the number of actuators that the device would require and the third established the volume or workspace that the device would possess.

1. *Force and motion are the most important haptic cues*. A significant component of our ability to "visualize", remember and establish cognitive models of the physical structure of our environment stems from haptic interactions with objects in the environment. Kinesthetic, force and cutaneous senses combined with motor capabilities permit us to probe, perceive and rearrange objects in the physical world. Even without detailed cutaneous information (as with a gloved hand or tool), the forces and motions imparted on/by our limbs and fingers contribute significant information about the spatial map of our environment. Information about how an object moves in response to applied force and the forces which arise when we attempt to move objects can provide cues to geometry (shape, locality, identity), attributes (constraint, impedance, friction, texture, etc.) and events (constraint, change, contact, slip) in the environment. Unlike other sensory modalities, haptic interactions permit two-way interaction via work exchange. Controlled work can be performed on dynamic objects in the environment and modulated to accomplish tasks.

2. *Many meaningful haptic interactions involve little or no torque*. Perhaps the most significant design feature of the PHANToM is the passive, 3 degree-of-freedom "thimble-gimbal", shown in Figure 2. The decision to use the

T. H. Massie and J. K. Salisbury. *The PHANToM haptic interface: A device for probing virtual objects*. In Proceedings of the Third International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, ASME Dynamic Systems and Control, Volume 55(1), pages 295-300. American Society of Mechanical Engineers, 1994.

This is the most highly cited paper in the field of haptics. It describes the design objectives of the Phantom haptic interface, which was developed in Ken Salisbury's lab at MIT. The paper also anecdotally describes the reactions of users of the device. The described device is very similar to the SensAble Phantom Premium 1.0 you are using in this class.

PHANToM means
"Personal HAptic iNTerface Mechanism"

Where did the "o" come from?

The student inventor's name was
Thomas "Tom" Massie

Lesson: Name your robot after yourself.

thomas massie - Google Search

http://www.google.com/#q=thomas+massie

Google

thomas massie

+Katherine 3 Share

Web Images Maps Shopping News More Search tools

About 2,690,000 results (0.18 seconds)

Thomas Massie for US Congress - 4th District Kentucky: Home
www.**thomasmassie**.com/
"Industrial hemp is a sustainable crop and could be a great economic opportunity for Kentucky farmers" WASHINGTON – Congressman **Thomas Massie** (R-KY) ...

Thomas Massie - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Thomas_Massie**
Thomas Harold **Massie**' (born January 13, 1971) is an American politician who has been the United States Representative for Kentucky's 4th congressional ...
Early life, education, and ... - Judge Executive of Lewis County

Congressman **Thomas Massie** - House of Representatives
massie.house.gov/
***ADVISORY: Rep. Massie Announces Two-Day Listening Tour in Western Region of 4th District ... **Thomas Massie's** new office in Cannon. We looking forward ...

News for **thomas massie**

Kentucky Congressman **Thomas Massie** Objects to Plastic Gun Ban
WFPL - 1 day ago
Kentucky Fourth District Congressman **Thomas Massie** says he was the sole voice vote against a 25-year ban on firearms that can evade metal ...

House extends Undetectable Firearms Act as Rep. **Massie** claims sole opposi...
Examiner.com - 1 day ago

Thomas Massie

United States Representative

Thomas Harold Massie' is an American politician who has been the United States Representative for Kentucky's 4th congressional district since 2012. Previously he was Judge-Executive of Lewis County, Kentucky from 2011 to 2012. Wikipedia

Born: January 13, 1971 (age 42), Vanceburg, KY

Office: Representative (R-KY 4th District) since 2013

Spouse: Rhonda Massie

Go to "http://www.google.com/imgres?imgurl=http://upload.wikimedia.org/wikipedia/commons/thumb/f/fd/Thomas_Massie,_official_portrait,_112th_Congress.jpg/220px-Thomas_Massie,_official_port

thomas massie - Google Search

http://www.google.com/#q=thomas+massie

Google

Thomas Massie for US Congress - 4th District Kentucky: Home
www.**thomasmassie**.com/
"Industrial hemp is a sustainable crop and could be a great economic opportunity for Kentucky farmers" WASHINGTON – Congressman **Thomas Massie** (R-KY) ...

Thomas Massie - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Thomas_Massie**
Thomas Harold **Massie**' (born January 13, 1971) is an American politician who has been the United States Representative for Kentucky's 4th congressional ...
Early life, education, and ... - Judge Executive of Lewis County

Congressman **Thomas Massie** - House of Representatives
massie.house.gov/
***ADVISORY: Rep. Massie Announces Two-Day Listening Tour in Western Region of 4th District ... **Thomas Massie's** new office in Cannon. We looking forward ...

News for **thomas massie**

Kentucky Congressman **Thomas Massie** Objects to Plastic Gun Ban
WFPL - 1 day ago
Kentucky Fourth District Congressman **Thomas Massie** says he was the sole voice vote against a 25-year ban on firearms that can evade metal ...

House extends Undetectable Firearms Act as Rep. **Massie** claims sole opposi...
Examiner.com - 1 day ago

Congressman **Thomas Massie** | Facebook
https://www.facebook.com/Rep**ThomasMassie**
Congressman **Thomas Massie**, Washington, DC. 25134 likes · 2396 talking about this. Here's a little bit of information regarding my life prior to taking office: For ...

Thomas Massie (RepThomasMassie) on Twitter
https://twitter.com/Rep**ThomasMassie**
The latest from **Thomas Massie** (@RepThomasMassie). U.S. Representative for Kentucky's 4th District.

Thomas Massie

United States Representative

Thomas Harold Massie' is an American politician who has been the United States Representative for Kentucky's 4th congressional district since 2012. Previously he was Judge-Executive of Lewis County, Kentucky from 2011 to 2012. Wikipedia

Born: January 13, 1971 (age 42), Vanceburg, KY

Office: Representative (R-KY 4th District) since 2013

Spouse: Rhonda Massie

Education: Massachusetts Institute of Technology

Awards: Lemelson-MIT Student Prize

People also search for

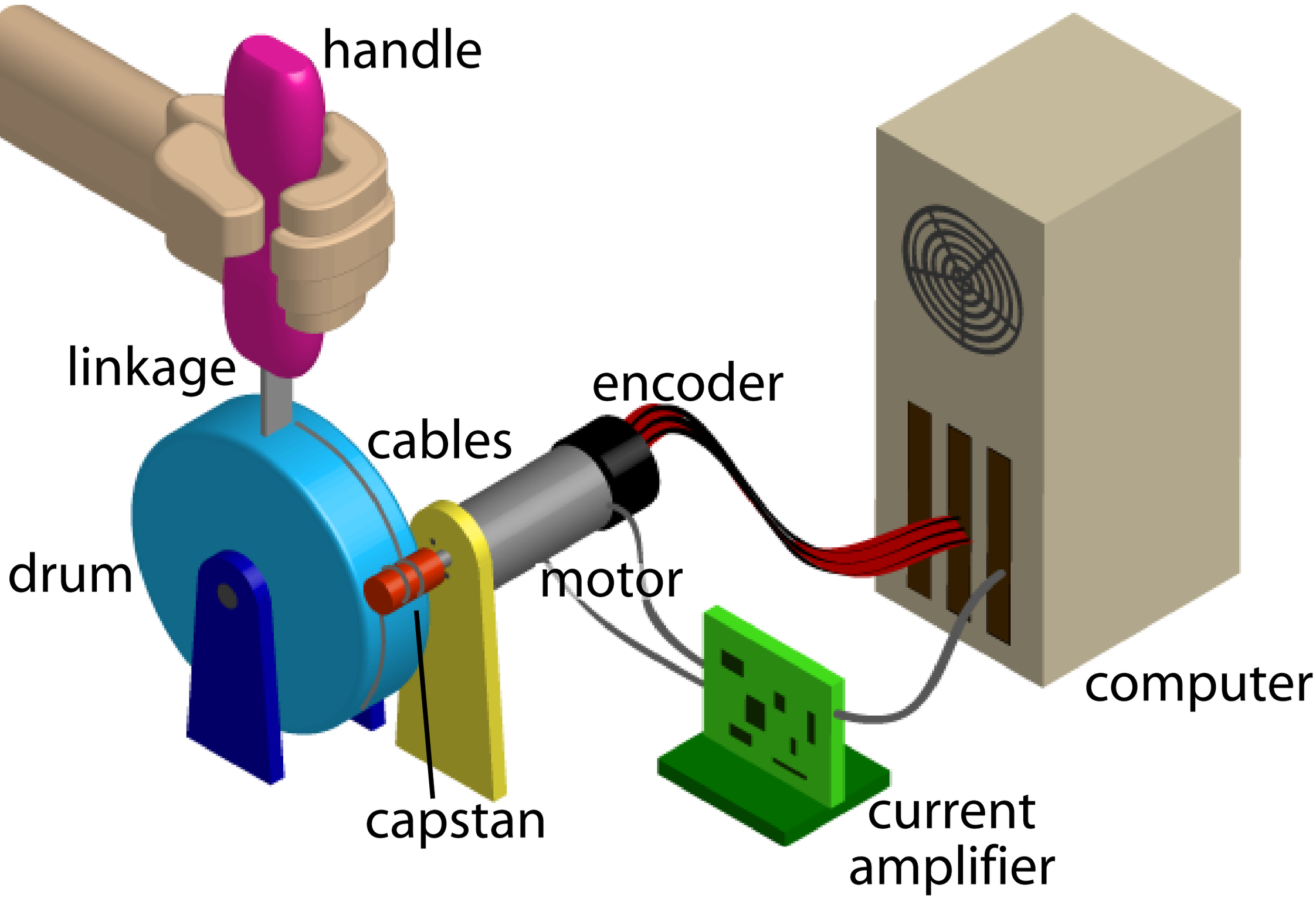
Three Necessary Criteria

1. Free space must feel free.

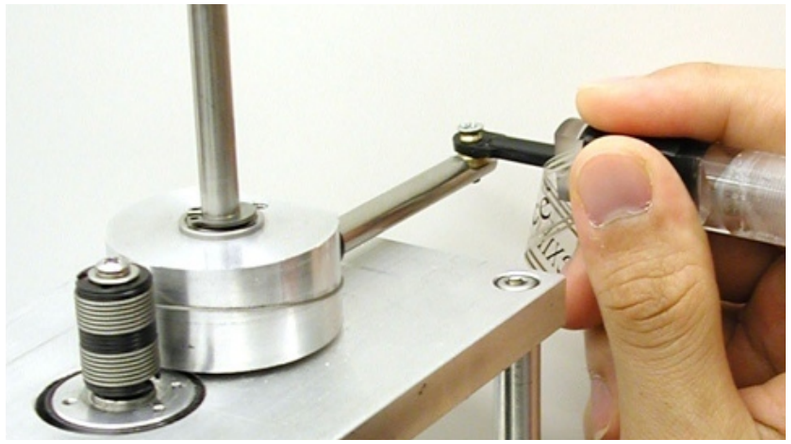
2. Solid virtual objects must feel stiff.

3. Virtual constraints must not be easily saturated.

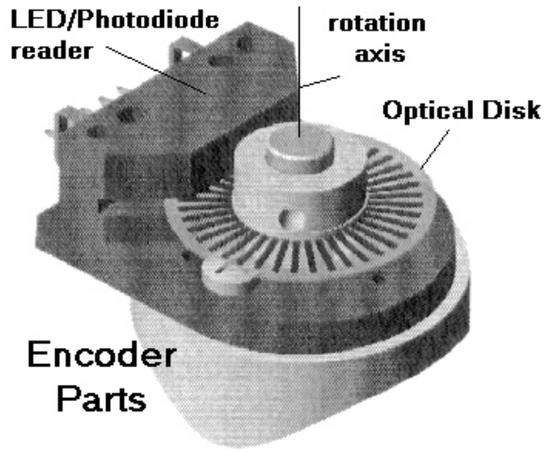
We want a design that can achieve all of these objectives simultaneously.


handle
linkage
encoder
cables
drum
motor
computer
capstan
current
amplifier

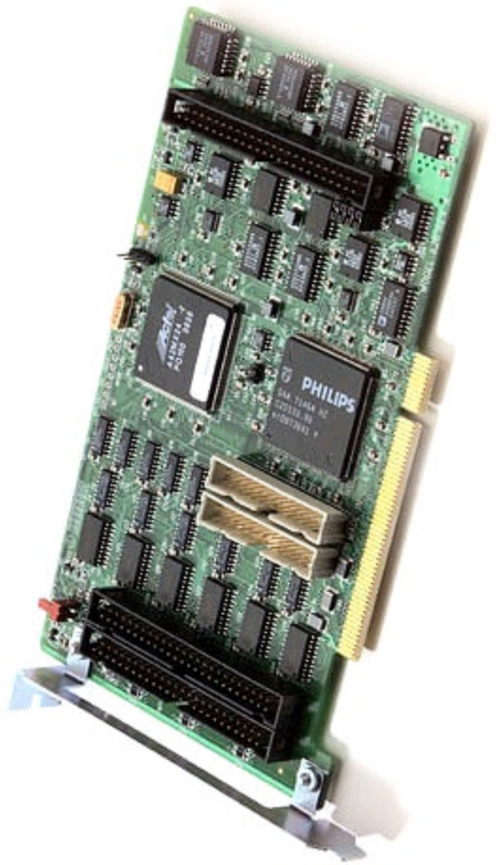
Common Haptic Interface Components



Capstan & Cable Drive
Stiff Metal Linkages

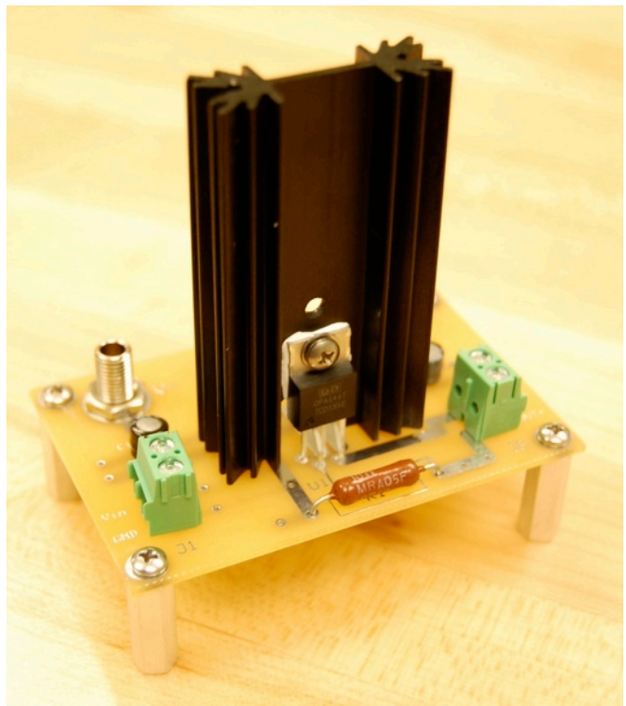


Incremental
Optical Encoder



Computer Interface Card

Brushed Permanent Magnet
Direct Current Motor



Current Amplifier

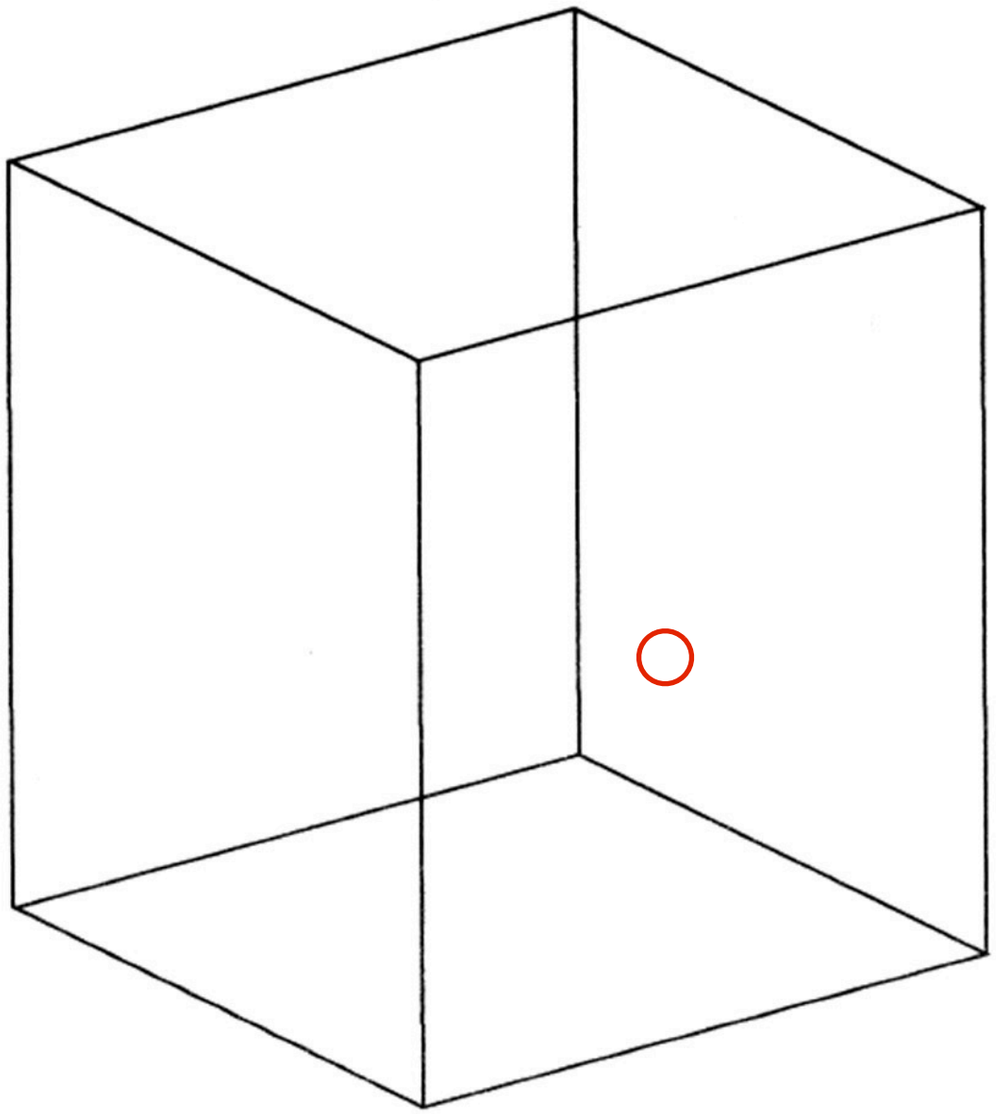
Your General Haptics Programming Task

Specify:
Force vector

As a function of:
Everything that is known, including
position vector, velocity vector, time,
model geometry, model properties,
keyboard inputs, mouse inputs, button inputs,
and randomness

In order to:
Fool the user into thinking they are touching something

How would you make the user believe they are trapped inside a box?



High-Level Approach For Rendering Shapes

Used by Massie and Salisbury in 1994

Now known as the *Vector-Field Approach*

- Force is a function of only the present position

$$\vec{F}_h = f(\vec{x}_h)$$

- Divide space into volumes that represent objects
- The rest of the volume is free space.
- Object contact is rendered by keeping track of a geometric proxy. It stays on the surface of the object, and we use a virtual spring to pull the user toward its location, always perpendicular to the surface.

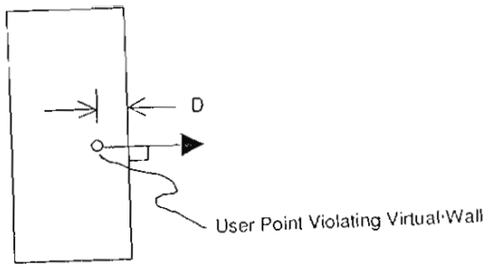


Figure 4: Simulation of virtual planes. Force, $F = Kx$, is exerted normal to plane when the user pushes into virtual surface.

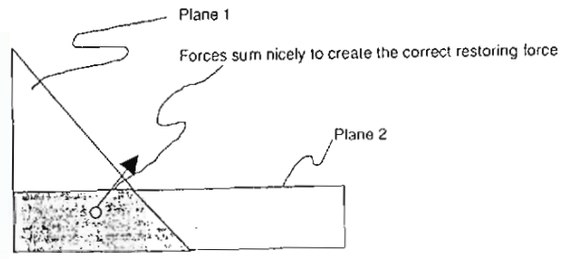


Figure 5: Forces at interior corner defined by 2 planes. Forces sum properly to create correct restoring force when planes meet at obtuse angle.

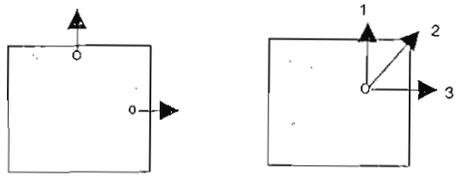


Figure 6: Virtual Cubes. Figure at left illustrates reasonable force vectors for cube with slightly compliant surface. For the point shown in figure at right, it is not clear which of forces 1, 2 and 3 should be exerted. This is path dependent.

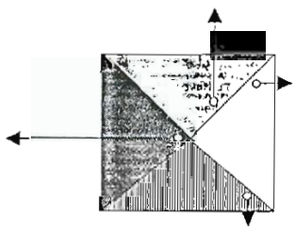


Figure 7: Solution to corner problem with cubes. Dividing cube into regions shown provides simple solution to path ambiguity problem. In 3-D, regions are pyramid in shape and permit stable behavior at edges and corners. If large forces are exerted at corners, probe point may move into the adjacent region and be pushed off object giving the sensation of "plucking" the corner.

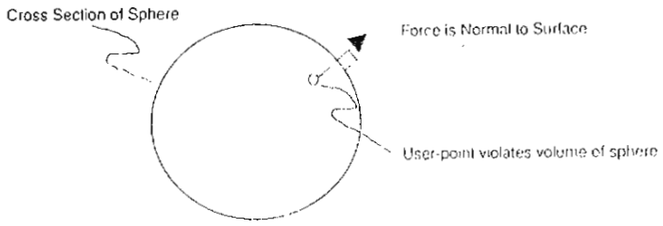


Figure 8: Forces generated by contact with sphere. As with planes, force is proportional to penetration depth and in direction normal to surface. Because force is always normal, spheres feel very slippery.

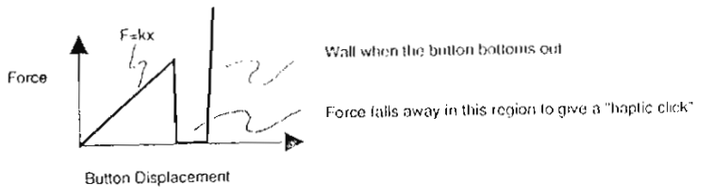
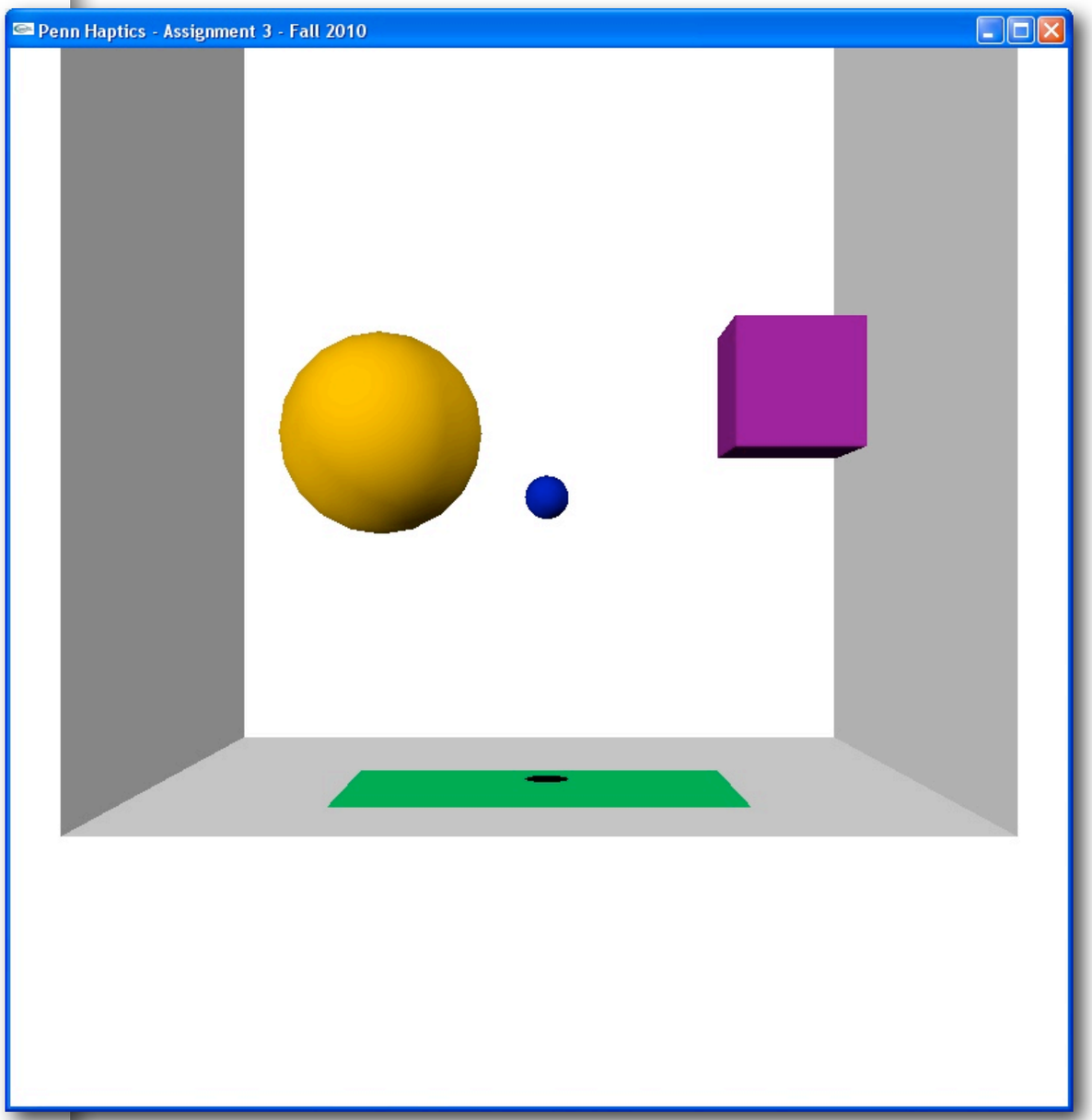
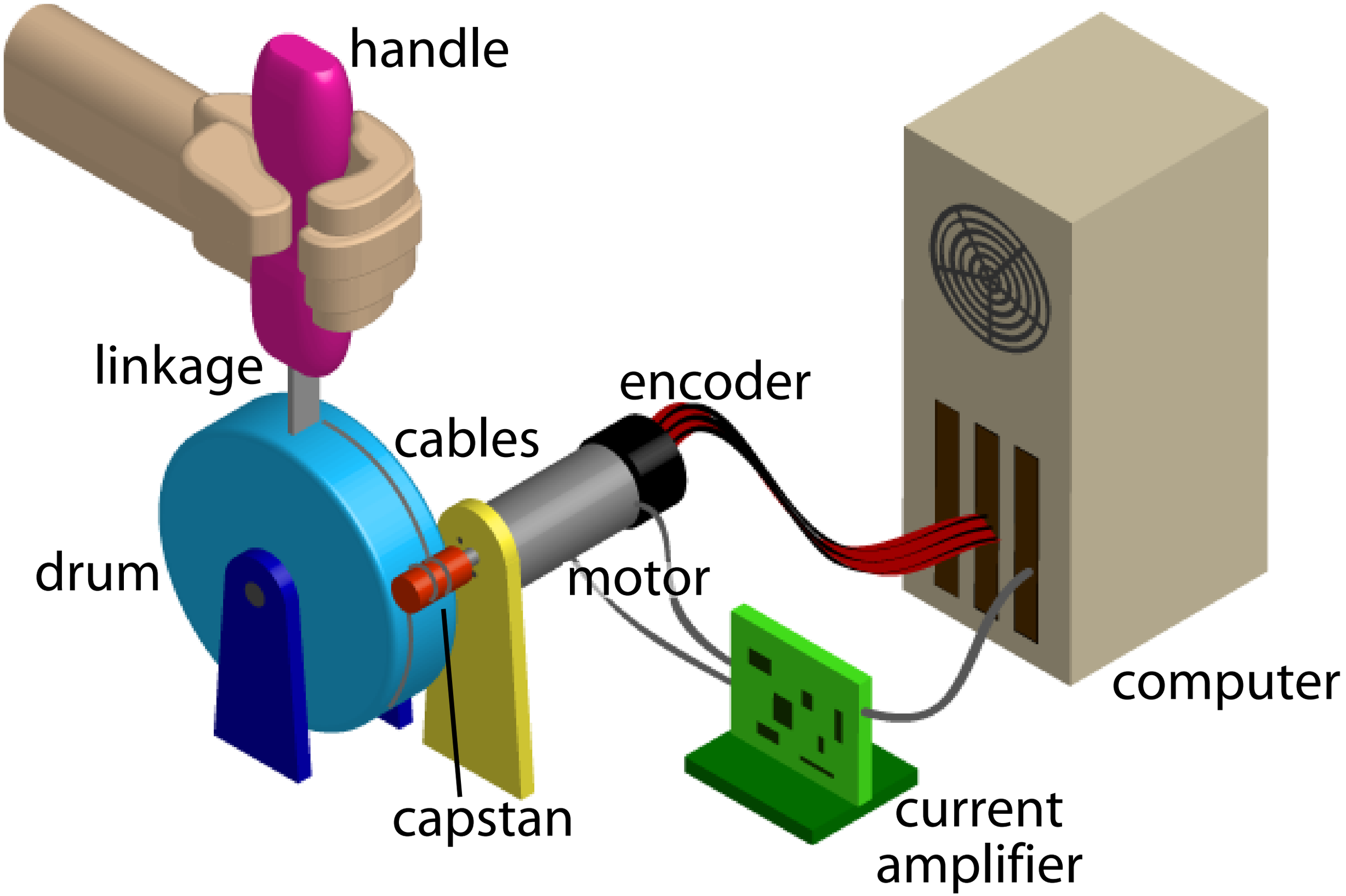


Figure 9: Virtual buttons. Nonlinear force-deflection curve for button demonstration showing "fall-away" force characteristic.

11 Appendix 1: PHANToM Specifications

Force resolution:	12 bit
Nominal spatial resolution:	400 dpi
Peak force:	10 Nt
Continuous force:	1.5 Nt
Backdrive friction:	0.1 Nt
Max/min force (Dyn. range):	100 : 1
Inertia at tip:	100 gm
Workspace:	8 x 17 x 25 cm
Max. object stiffness	35 Nt/cm




handle
linkage
encoder
cables
drum
motor
computer
capstan
current
amplifier

Typical Software Configuration

Input: from sensor signals to counts

Counter or Serial or A2D
(Analog-to-Digital Converter)

Input Processing: from counts to joint values

$$\vec{q}_k = \vec{a} * \vec{Q}_k + \vec{b}$$

Forward Kinematics: from joint values to tip position

$$\vec{x}_k = \Lambda(\vec{q}_k)$$

Force Computation: from tip position to tip force

$$\vec{f}_k = F_i(\vec{x}_k)$$

Torque Computation: from tip force to joint torques

$$\vec{\tau}_k = [J(\vec{q}_k)]^T \vec{f}_k$$

Output Processing: from joint torques to counts

$$\vec{r}_k = \vec{c} * \vec{\tau}_k + \vec{d}$$

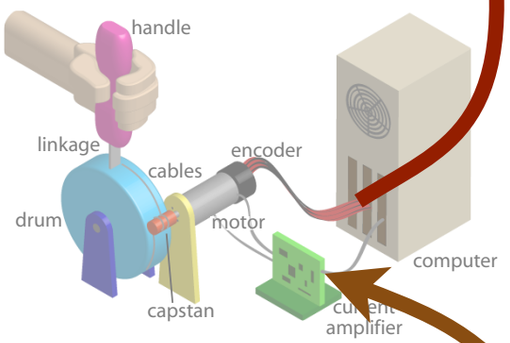
Output: from counts to command signals

Digital or Serial or D2A
(Digital-to-Analog Converter)

Graphics, Remote Robot,
and Other Slower Processes

from the
motion sensors

to the
current amplifiers



Typical Software Configuration

Input: from sensor signals to counts

Counter or Serial or A2D
(Analog-to-Digital Converter)

$$\begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix}$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \text{ or } \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \text{ or } \begin{bmatrix} d_1 \\ \theta_2 \\ d_3 \end{bmatrix} \text{ or } \begin{bmatrix} \vdots \end{bmatrix}$$

Input Processing: from counts to joint values

$$\vec{q}_k = \vec{a} * \vec{Q}_k + \vec{b}$$

Forward Kinematics: from joint values to tip position

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\vec{x}_k = \Lambda(\vec{q}_k)$$

Force Computation: from tip position to tip force

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}$$

$$\vec{f}_k = F_i(\vec{x}_k)$$

Torque Computation: from tip force to joint torques

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$$

$$\vec{\tau}_k = [J(\vec{q}_k)]^T \vec{f}_k$$

Output Processing: from joint torques to counts

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

$$\vec{r}_k = \vec{c} * \vec{\tau}_k + \vec{d}$$

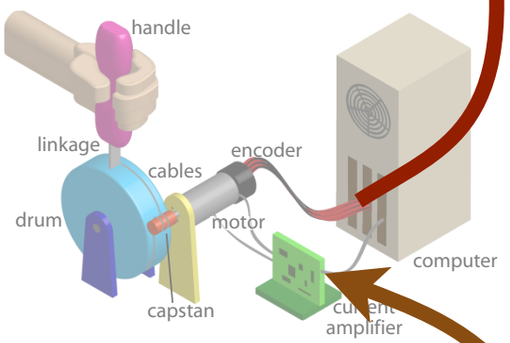
Output: from counts to command signals

Digital or Serial or D2A
(Digital-to-Analog Converter)

Graphics, Remote Robot,
and Other Slower Processes

from the
motion sensors

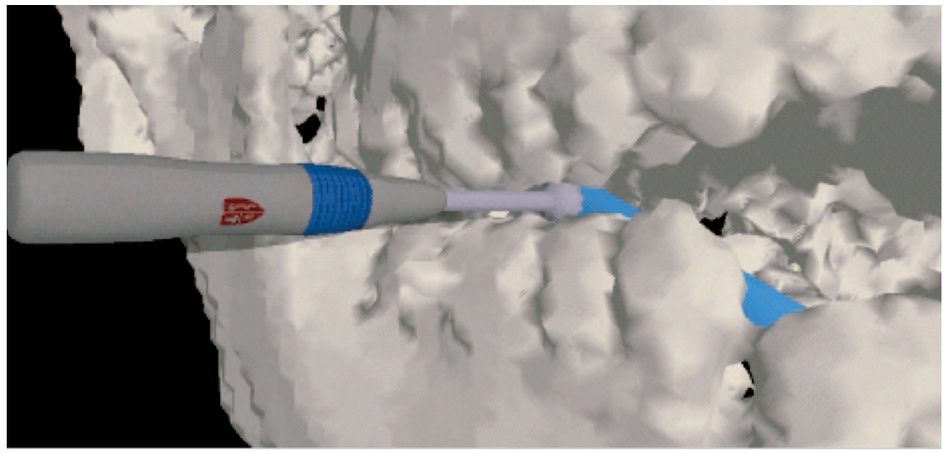
to the
current amplifiers

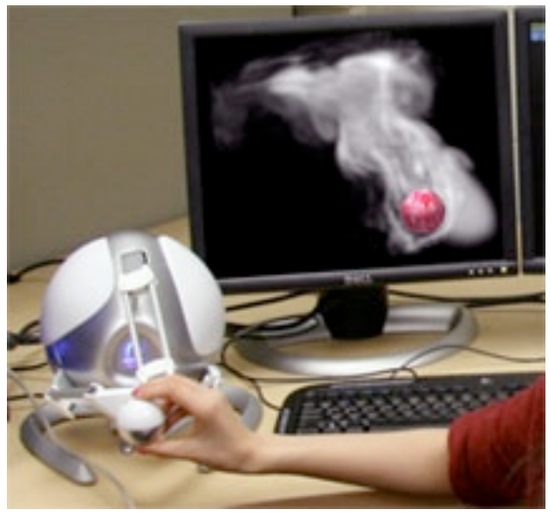


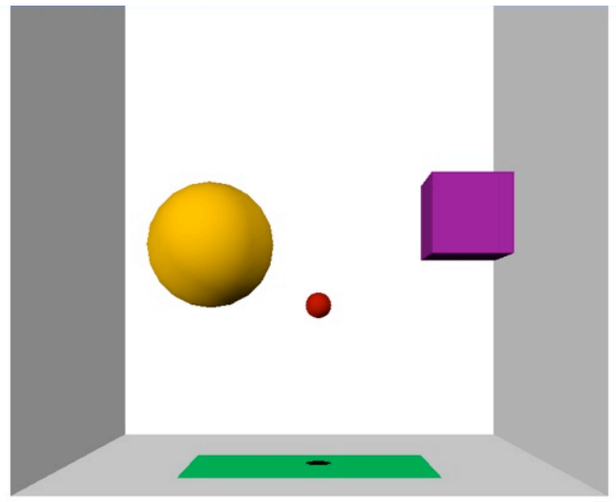
Force Computation

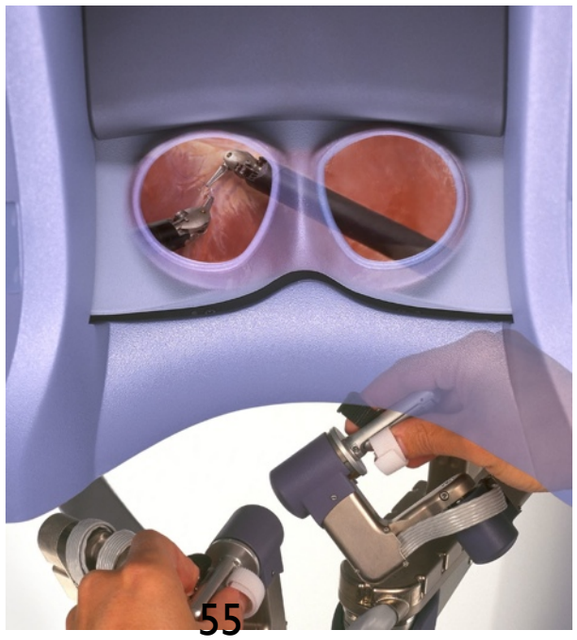
Force Computation: from tip position to tip force

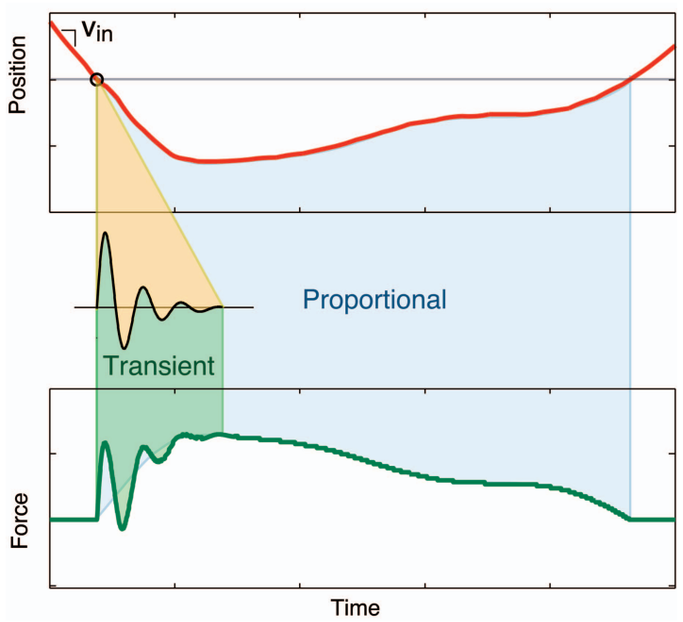
$$\vec{f}_k = F_i(\vec{x}_k)$$











Homework 9:
Haptic Rendering with the Phantom

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

December 5, 2013

This assignment is due on **Tuesday, December 10, by midnight (11:59:59 p.m.)** Your code should be submitted via email according to the instructions at the end of this document. To help you handle the intensity of the end of the semester, **late submissions will be accepted with no penalty until midnight on Saturday, December 14**. Submissions after Saturday will be penalized by 10% for each partial or full day late, up to 20%. After Monday, December 16, 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!

Pair Programming

Because it requires the use of a real robot, you must do this assignment with a partner; individual submissions are not allowed. Working with a partner will help keep you safe, keep the Phantom safe, and ensure everyone has a chance to work with the robot in the time available. You may work with any other student in the class. If you're looking for a partner, consider using the "Search for Teammates!" tool on Piazza.

You should work closely with your partner throughout this assignment, following the paradigm of pair programming. You will turn in one set of MATLAB scripts for which you are both jointly responsible, and you will both receive the same grade. Please follow these pair programming guidelines, which were adapted from "All I really need to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000:

- Start with a good attitude, setting aside any skepticism and expecting to jell with your partner.
- Don't start writing code alone. Arrange a meeting with your partner as soon as you can.
- Use just one computer, and sit side by side; a desktop computer with a large monitor is better for this than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (using the mouse and keyboard or recording design ideas) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every thirty minutes, *even if one partner is much more experienced than the other.* You may want to set a timer to help you remember to switch.
- If you notice a bug in the code your partner is typing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Recognize that pair programming usually takes more effort than programming alone, but it produces better code, deeper learning, and a more positive experience for the participants.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.

Homework 9

Done in pairs.

Nominally due by midnight on Tuesday. No late penalty if in by midnight Saturday.

Should be fun and interesting.

Task Description

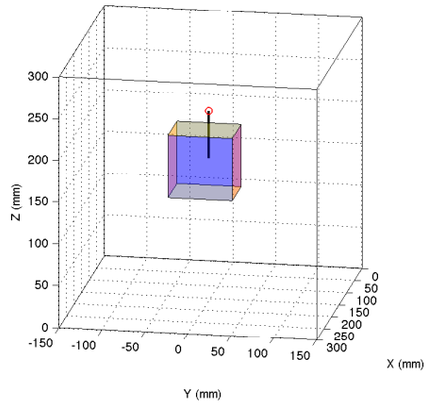
Like Homework 8, this assignment centers on the SensAble Phantom Premium 1.0, an impedance-type haptic interface with three actuated rotational joints. Designed to be lightweight, stiff, smooth, and easily backdrivable, this type of robotic device enables a human user to interact with a virtual environment or control the movement of a remote robot through the movement of their fingertip while simultaneously feeling force feedback.

As described below, there are three parts to this assignment. The first part aims to familiarize you with the functionality of the Phantom and its simulator. The second and third parts ask you to modify provided starter code to achieve a desired behavior of the Phantom. Everything you need to know to accomplish this assignment was covered in lecture. To learn more about haptic rendering, we encourage you to look over "The PHANToM Haptic Interface: A Device for Probing Virtual Objects" by Thomas H. Massie and J. Kenneth Salisbury, which was handed out in class and is also posted on Piazza under Lecture Resources.

Begin by downloading the **starter code** from the assignment resources section on Piazza; it is provided as a zip file named `hw09_starter.zip`. You will first develop your code for this assignment using the simulated version of the Phantom that is included in the starter code. We are providing you with p-coded versions of all the functions you need. The code starts the Phantom by calling `pumaStart(hardware)`, where `hardware` is a Boolean value. Setting `hardware` to `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. You choose the duration of the test by setting the value of `nCycles` at the top of the starter code.

Once your team has written a good draft of the code for each section, you will need to test it on the real Phantom and make any necessary refinements. Detailed instructions for testing on the Phantom are included at the end of this document.

1. Haptic Box Demo

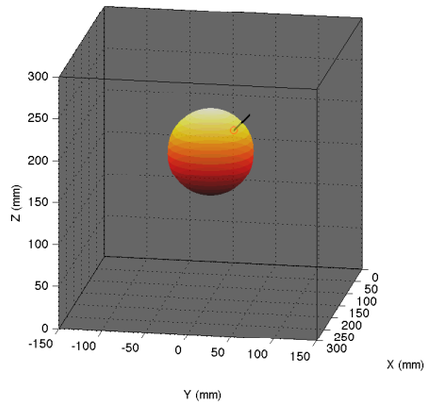


After you download the starter code from Piazza, run `haptic_box_demo.m` and look at how it is written. This demonstration creates a virtual haptic box for the user to feel, as seen in the illustration at right. The user is trapped inside the virtual box and feels a virtual spring force each time they 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 location and size of the box are controlled through variables such as `boxWidthX`; all dimensions are in millimeters.

The provided software 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`, and watch how the force vector changes as the simulated user moves around. This is the type of mapping you will need to create in this assignment.

Play around with this simulation to make sure you understand it. Try changing the location of the box and the stiffness of the walls. Notice that the code uses variables rather than hard-coded values (magic numbers) so it's easier to modify, and it includes comments to explain what's being done at each step of the calculation; your code should do the same. Once you understand how the haptic box demo works, you should move on to the next part. There is no deliverable for this part of the assignment.

2. Haptic Ball



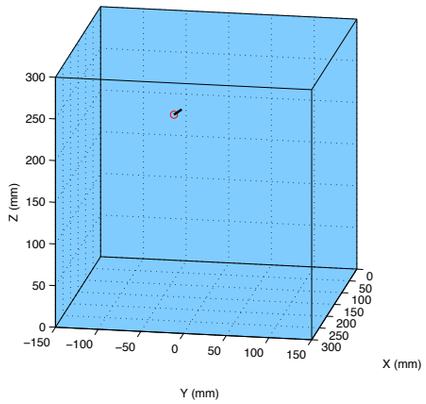
Complete the haptic ball scene that has been started for you in `haptic_ball_starter.m`. Change the filename to include your team members' PennKeys (`_pennkey1_pennkey2`) instead of `_starter`, and enter the names of your team members at the top of the file in the `studentNames` variable.

The graphics for this scene are shown in the image at right. Again, the position of the Phantom tip is shown as a red circle, the ball is shown as a transparent warm-colored sphere, and a scaled version of the force vector is shown as a thick black line. The background is gray like the outer space around the sun.

Modify the code between the two lines of stars to push the user out of the ball whenever they come inside, using a virtual spring mapping. The force should push them straight out of the ball, and the magnitude of the force should be proportional to their penetration depth. The stiffness of your spring should be `k` (already defined for you in the code using units of newtons per millimeter). The ball's surface should not include any friction or other haptic effects. Please comment your code.

Use the graphics to debug your calculations in simulation (with hardware set to false) until both of your team members are convinced the calculations are correct. Only then should you try your code on the real Phantom, following the instructions at the end of this document.

3. Haptic Damping



Complete the haptic damping scene that has been started for you in `haptic_damping_starter.m`. Change the filename to include your team members' PennKeys, and enter the names of your team members at the top of the file in the `studentNames` variable.

The graphics for this scene are shown in the image above. Again, the position of the Phantom tip is shown as a red circle, and a scaled version of the force vector is shown as a thick black line. The background is blue like water.

Modify the code between the two lines of stars to output a viscous damping force that always acts to slow the user down. This force should act in the opposite direction to the user's velocity vector, and its magnitude should scale with the magnitude of their velocity vector. The viscosity of your damper should be `b` (already defined for you in the code using units of newtons per millimeter per second). Please comment your code.

When correctly implemented, viscous damping should feel like you are moving your hand through molasses or honey. There should not be any jittering in the force. This means you will need to low-pass filter your velocity vector to reduce quantization noise. Be careful to implement the velocity filter correctly; examine the graph produced at the end of the simulation to confirm that the filter has the effect you want.

Use the graphics to debug your calculations in simulation (with hardware set to false) until both of your team members are convinced the calculations are correct. Only then should you try your code on the real Phantom, following the instructions at the end of this document.

Testing Your Code on the Phantom

The actual Phantom robot is located in Towne B2 and is attached to the same computer that runs the PUMA robot. All members of this class have been given card-swipe access to Towne B2 for this homework assignment. The Phantom and its computer are the only equipment you'll be using. You should not touch anything else in the room, nor should you attempt to use the PUMA robot.

- You must reserve the Phantom computer in order to test your code. You may reserve only one sixty-minute slot (or two thirty-minute slots) at a time. Respect the next team by finishing on time. The link to the Phantom reservation system will be posted on Piazza.
- Use your Penn ID card to swipe into Towne B2. You may need to swipe your card several times to get the door to unlock.
- Log in to the computer using the "puma student" account. The password is "meam520" without the quotation marks. This is a Linux computer running Ubuntu 12.04. Please do not reboot the computer or modify any of its settings. If you run into computer issues, post a note on Piazza.
- Once logged in, double-click the "Homework 9" folder on the Desktop. Create a folder for your team, naming it with your PennKeys (i.e., `pennkey1_pennkey2`). Do not look at, modify, or run any other code that may be on this computer.
- Copy only your team's main script files (`haptic_box_demo.m`, `haptic_ball_pennkey1_pennkey2.m`, and `haptic_damping_pennkey1_pennkey2.m`) over to the folder you just created; do not copy the rest of the Phantom simulator files. The easiest methods for transferring your files are to use a USB key or to email them to yourself as attachments. You can access the Internet using the Firefox web browser link on the desktop. Open a file browser by clicking "Places" in the upper-left menu bar.
- Once you have your files transferred to this computer, double-click the large MATLAB icon on the Desktop. It will ask, "Do you want to run "MATLAB", or display its contents?" Choose **Run in Terminal**. MATLAB should open. Change the working directory to your folder.
- Before you run any of your code, you **MUST** check the calibration of the Phantom. With the Phantom's emergency stop pressed down, call `phantomStart(true)` from the command line. Then hold the robot in its zero configuration and call `phantomJointAngles` from the command line. If the joints do not all read very close to zero radians at this pose, the robot is not correctly zeroed, so you must run `phantomZero` from the command line before proceeding. Call `pumaStop` when done.
- You must have the Phantom emergency stop down (no forces) the first time you run any code, even the demo code. Watch the graphical output to see if your code is doing what it should. The black line shows a scaled force vector.
- You must firmly grip the Phantom gimbal every time you run any code. If the thimble is too big for your finger, wrap a rubber band around it to make it tighter, but also hold on. Do not let the robot move freely until you are confident your code is doing what it should. The robot can be damaged if it slams into the joint limits.
- Someone **MUST** have their hand on the emergency stop whenever the Phantom is running. Push down the emergency stop as soon as anything unusual happens.
- Start by running `haptic_box_demo.m` to get a feel for how it works. (Make sure you run it first with the emergency stop down. Then make sure to hold firmly onto the thimble and have a hand over the emergency stop the entire time it is running.)
- Then run each of the scripts that you edited for this assignment. Run them first with `hardware` set to `false`, and make sure they are doing what you intend. Then run the code with `hardware` set to `true` but with the emergency stop down, so you can see how the forces change with your motion. Once you're confident the force vector is correct, run it again with the emergency stop up. Update your code to improve it if needed.
- Please immediately post on Piazza if you notice any problems with the Phantom!

Submitting Your Code

Follow these instructions to submit your code:

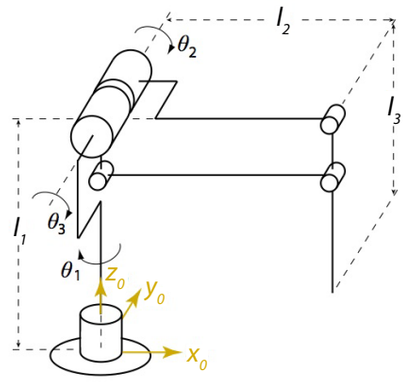
1. Start an email to `meam520@seas.upenn.edu`
2. Make the subject *Homework 9: Your Name and Your Teammate's Name*, replacing *Your Name* and *Your Teammate's Name* with the appropriate full names.
3. Attach your two correctly named MATLAB files to the email as individual attachments; please do not zip them together or include any other attachments. The files should be the following:
   - `haptic_ball_pennkey1_pennkey2.m`
   - `haptic_damping_pennkey1_pennkey2.m`
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 not updated all of them.

Follow the rules for using the Phantom!

PHANToM Rules

1. You must reserve the PHANToM computer in order to test your code. You may reserve only one one-hour slot at a time. Respect the next team by finishing on time.
2. Copy only your haptic_box_demo.m, haptic_ball_teamXX.m, and haptic_damping_teamXX.m scripts, not the rest of the phantom simulator files. Use your team's folder in the working directory on the computer's desktop.
3. Check the calibration of the PHANToM by calling phantomJointAngles from the MATLAB command line. If the joint angles are not very close to zero when the PHANToM is in the zero pose (right), run phantomZero from the command line and follow the prompts.



4. Start by running haptic_box_demo.m to get a feel for how it works.
5. You must have the PHANToM emergency stop down (no forces) the first time you run any code, even the demo code. Watch the graphical output to see if your code is doing what it should. The black line shows a scaled force vector.
6. You must firmly grip the PHANToM gimbal every time you run any code.
7. Someone must have their hand on the emergency stop whenever the PHANToM is running. Depress it right after your code finishes.

Please immediately post on Piazza if you notice any problems with the PHANToM!

Editor - /Users/kuchenbe/Documents/teaching/meam 520/assignments/09 haptics/hw09_starter/haptic_box_demo.m

EDITOR | PUBLISH | VIEW

haptic_box_demo.m | haptic_ball_starter.m | haptic_damping_starter.m

```
1     %% haptic_box_demo.m
2     %
3     % This script enables the user to touch a virtual box using a PHANToM
4     % Premium 1.0 haptic interface.  The user's position is shown as a small
5     % red circle, and the sides of the box are transparent.  The user is
6     % trapped inside the box, and each wall is rendered using a virtual spring.
7     % This script is provided to you in fully functional form so you can see
8     % and feel how haptic rendering works.
9     %
10    % Written by Katherine J. Kuchenbecker for MEAM 520 at the University of Pennsylvania.
11
12
13    %% Clean up
14
15 -  clear all
16 -  close all
17 -  clc
18
19
20    %% Set hardware mode, duration, and warnings
21
22    % Set whether to use the actual PHANToM hardware.  If this variable is set
23    % to false, the software will simulate the presence of a user by reading a
24    % pre-recorded trajectory.  You should use this mode to debug your code
25    % before running anything on the PHANToM computer.  Once you are on the
26    % PHANToM/PUMA computer in Towne B2, you may set this variable to true.
27    % Always first run your code with the emergency stop down to prevent the
28    % application of forces and make sure everything works correctly on the real
29    % robot.  Hold onto the PHANToM tightly and keep a hand on the emergency stop.
30 -  hardware = false;
31
32    % Set how many times we want our servo loop to run.  Each cycle takes about
33    % two milliseconds on the computer in Towne B2.  There are 10000 cycles in
34    % the recorded data file.  If nCycles exceeds 10000, the motion loops.
35 -  nCycles = 2000;
36
```

script | Ln 12 Col 1