ABSTRACT

A STUDY OF KINEMATICS AND KINETICS IN TIME-CONSTRAINED ARM MOVEMENTS

by
Oyindamola Owoeye

Several studies such as the equilibrium point hypothesis (EPH) purport that the motor signals that descend from the brain instead of encoding muscle torques, influence an existing relationship between muscle torque and body configuration.

In the present study, the possibility of torque depending explicitly on position was tested using a task in which subjects (N=5) moved a simulated weightless frictionless mass through a small (<8 degree) elbow extension in order to move a cursor on a screen to a target location. Each subject completed 720 trials. On ~10% of trials the simulated mass was increased unknown to the subject. The relationship between the cursor's position and the torque applied to the system was held constant even when the simulated mass was increased. Thus, any change in torque produced was neither due the subjects' perception of the mass nor due to their perception of the cursor. The time at which the subjects torque changed direction was seen to be significantly different ($p<0.005$) during trials which the mass changed. This change in torque is concluded to be position-dependent. However the possibility of this being a merely mechanical effect could not be ruled out by due to poor EMG collection.

A post-hoc analysis of different position-dependent motor control models, was done. Particularly, an exponential spring model, a linear spring model, and a linear spring with relative damping model were each tested to see how well they could predict a change

in produced output torque from a change in position. Only the linear spring and relative damping models were able to do so.

This experiment is not enough to prove that descending torque produced is systematically position-dependent but the methodology for testing models is promising and additional studies should be done along similar lines.

A STUDY OF KINEMATICS AND KINETICS IN TIME-CONSTRAINED ARM MOVEMENTS

by
Oyindamola Owoeye

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Masters if Science in Biomedical Engineering

Department of Biomedical Engineering

January 2016

APPROVAL PAGE

A STUDY OF KINEMATICS AND KINETICS IN TIME-CONSTRAINED ARM MOVEMENTS

Oyindamola Owoeye

Dr. Richard Foulds, Thesis Advisor　　　　Date
Associate Professor of Biomedical Engineering, NJIT

Dr. Sergei Adamovich, Committee Member　　　　Date
Associate Professor of Biomedical Engineering, NJIT

Dr. Tara Alvarez, Committee Member　　　　Date
Professor of Biomedical Engineering, NJIT

BIOGRAPHICAL SKETCH

Author:	Oyindamola Owoeye
Degree:	Master of Science
Date:	August 2015

Undergraduate and Graduate Education:

- Master of Science in Biomedical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2015

- Bachelor of Science in Mathematics,
  University of Pittsburgh, Pittsburgh, Pennsylvania, 2012

Major:	Biomedical Engineering

Since a man without limbs is no less loveable than I, and since I cannot conjure sentience, and since a man with malfunctioning organs is no less loveable than I, and since my laptop is less loveable than a dog, I must thank the God that created my lovability and sentience. Before I had either I did not covet them and yet now here they are.

ACKNOWLEDGMENT

The coincidence of my time at NJIT as a student with Dr. Richard Foulds time as a faculty was an extremely fortunate occurrence. While I can’t reasonably expect every individual I come across in academia to be as encouraging, supportive, and inspiring as Dr. Foulds has been in divers of my endeavors over the past two and a half years, I hope that if I ever find myself in such a position, I will attempt to be the same for future students. I’d like to thank Dr. Sergei Adamovich for taking a vested interest in my research and providing many resources to further my insight in the field. I would also like to thank Dr. Tara Alvarez for both her support and her superb teaching in the classroom.

A special thanks goes out to John Hoinowski without who was always of great assistance both during this study and in general. A special thanks also to Kevin and Kiran who helped me familiarize with the HapticMASTER and endured countless hours of pilot testing. They along with Madeline, Naphtali, and Peter were incredibly available to bounce ideas with.

No matter what I do or what I feel like I ever accomplish, I will always remember and acknowledge that without my parents, I wouldn’t know how to use a spoon.

TABLE OF CONTENTS

Chapter **Page**

1 INTRODUCTION……..........................……………..………………………… 1

1.1 Objective……………………………………………………………………….. 1

1.2 Past Animal Studies………………………………………………………. 2

1.3 Equilibrium-Point Models…………………………………………………… 3

2 METHODS……………………………………………………………………. 11

2.1 Admittance Control……………………………………………………….. 11

2.2 Experimental Set-up……………………………………………………… 13

2.3 Parameter Determination…………………………………………………. 17

3 RESULTS……………………………………………………………………... 19

3.1 Torque Sign Change……………………………………………………… 19

3.2 Cross-Determination……………………………………………………… 23

4 DISCUSSION…………………………………………………………………. 27

4.1 Discussion of Torque Sign Change………………………….…………. 27

4.2 Discussion of Cross-Determination……………………………………. 28

4.3 Future Directions………………………………………………………….. 29

A APPENDIX…………………………………………………………………. 30

A.1 Matlab Code for Task…………………………………………………….. 30

A.2 C# code for Task……………………………………………………………….. 36

A.3 Matlab Code for Cross-Determination…………………………………. 41

LIST OF TABLES

Table **Page**

2.1 Torque Sign Change Points………………………………………………………. 20

LIST OF FIGURES

Figure		Page
1.1	Apparatus used in classic animal motion studies………………………………	3
1.2	Convergent Force Fields………………………………………………….	4
1.3	EMG over grid……………………………………………………….…	5
1.4	Invariant characteristics…………………………………………………..	6
1.5	Coriolis reaching trajectories…………………………………….……….	8
2.1	Haptic Master Control Loop……………………………………………....	11
2.2	Screenshots of virtual environment from task…………………………..……	13
3.1	Example cursor trajectories……………………………………………….	19
3.2	Example cursor trajectory for excluded trial………………………………....	20
3.3	Trial average cursor trajectory, torque, and arm angle……………………...	21-22
3.4	Cross determinations……………………………………………………....	24-26

CHAPTER 1

BACKGROUND

1.1 Objective

The aim of the present study is to investigate the possibility that the activations of muscles in human arm movements, given a signal descending from the brain into the spinal cord, are position dependent. Mathematically, this could be written as

$$a(t) = f\big(s(t), x(t)\big) \tag{1.1}$$

where $a(t)$ is the activation of a muscle at time t, $s(t)$ is motor signal that is sent from the brain to the spinal chord, and $x(t)$ is a variable that is related to the positions and/or velocity of the (parts of the) arm. What this Equation implies is that even if $s(t)$ is fixed, $a(t)$ can be different as a result of a different $x(t)$. Alternatively it is possible that the central signal affects the muscle activation in a generally position-independent manner as in

$$a(t) = g\big(s(t)\big) \tag{1.2}$$

where g is a monotonic function.

Equation 1.1 can be considered a generalization of theories from the past several decades. Despite much research, position-dependent control has neither been proven nor disproven.

The following sections review these theories and discuss some reasons they haven't been completely accepted or rejected.

1.2 Past Animal Studies

The key to verifying position dependent spinal control is to fix $s(t)$, vary $x(t)$, and measure $a(t)$. Fixing $s(t)$ has been achieved in animal studies via direct microstimulation of the spinal column. Gizster, Mussa-Ivaldi, and Bizzi performed such an experiment on bullfrogs [4]. In their study, the bullfrog's spinal column was transected at the calamus scriptorius and microelectrodes were used to stimulate the frog's lumbar spinal cord. The stimulation elicited activation of the frog's leg muscles resulting in a force produced at the frog's ankle. The frog's ankle was restrained by a force transducer which also served the purpose of measuring the force elicited by the stimulation (Figure 1.1). Multiple stimulation sites were used and for each stimulation site, stimulation was repeated with the frog's ankle placed at different locations. Gizster et al. found that the evoked forces for a given stimulation site depended on the location of the ankle. Usually, the plot of forces against ankle position resulted in a "convergent force field", that is, the forces tended to be directed toward a certain location at which the force field was 0 (Figure 1.2). This point was termed the equilibrium point. The equilibrium point and force field was shown to be dependent on the site of spinal stimulation but not on the strength of stimulation. It should be noted that the evoked forces would be slightly position dependent even if Equation 1.1 does not hold simply due to mechanical properties of the limb; passive elastic forces will tend to bring the limb towards a certain position and the torque produced by muscles is dependent on

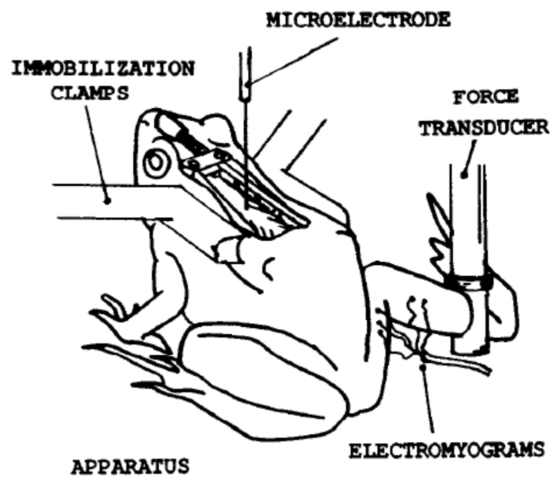

Figure 1.1 Apparatus used by Giszter, Mussa-Ivaldi, and Bizzi.
Source: Giszter, S. F., Mussa-Ivaldi, F. A., & Bizzi, E. (1993).

muscle length. Gizster et al. did compensate for these forces when finding the convergent force fields. They also collected electromyography (EMG) from the frog's legs. The amplitude of EMG can be considered related to the activation of muscles, thus, a dependency of EMG amplitude on position lends further evidence to position-dependent control. Indeed Gizster et al. found that the amplitude of EMG varied with position similar to how force varied (Figure 1.3).

The conclusions drawn from the frog study cannot be assumed to hold true for humans as well. However, similar experiments done with cats [10] and rats [7] have also shown convergent force fields.

1.3 Equilibrium-Point Models

One of the earliest motor control theories that involved position-dependent control was Merton's "servo hypothesis" [6]. In Merton's hypothesis, the length of a muscle is specified by the descending signal similar to how the angle of a servo motor is specified. This mode of control is achieved through modulation of the activity of γ-motoneurons which controls the sensitivity of the spindles to muscle-length. This change elicits the tonic stretch-reflex to change the activity of the α-motoneurons thus, bringing the muscle to a specified position. The servo hypothesis implied that a change in activity of α-motoneurons should be delayed relative to a change in the activity of γ-motoneurons. However, experiments shows that changes in α-motoneuron activity and γ-motoneuron activity are simultaneous. The servo hypothesis had two more fatal flaws in the fact that it required a very high gain of the stretch reflex and it implied a large delay due to its feedback loop. While the servo hypothesis did not survive, it was the first theory to unite the control of movement and posture into a single mechanism. It also provided a simple mode of control that simplifies the inverse kinematics the brain would need to compute to achieve a goal motion. The equilibrium-point hypothesis (EPH) shares these merits with the servo hypothesis but does not have the same crucial flaws.

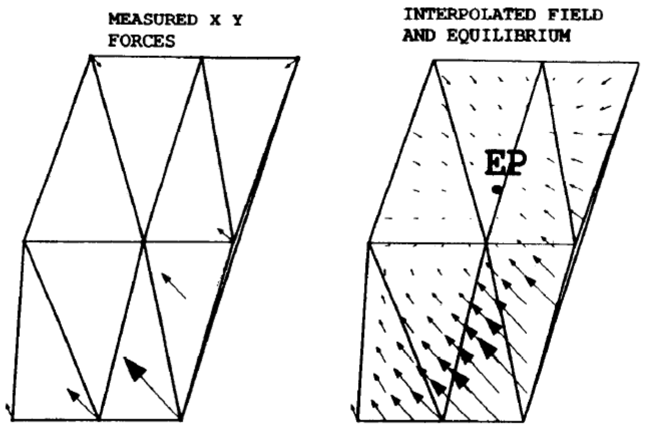

Figure 1.2 Force fields measured by Giszter, Mussa-Ivaldi, and Bizzi.
Source: Giszter, S. F., Mussa-Ivaldi, F. A., & Bizzi, E. (1993).

The original formulation of the equilibrium-point hypothesis was developed by Anatol Feldman in 1966. The basis of the EPH was experiments involving spinal stimulation of decerebrated cats [8]. Such experiments revealed that stimulation did not correspond to muscle activation but instead to a change in the force-length characteristic of the muscle. In general, the amount of force a muscle produces increases nonlinearly with its length. The muscle will change length unless the load and the muscle force balance out.

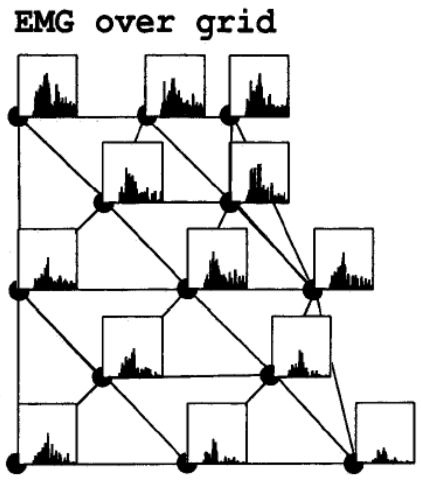

Figure 1.3 EMG from frog semitendinosus muscle collected by Gizster et al. The position of the each EMG trace corresponds with the position of the frog's hindlimb during stimulation.
source: Giszter, S. F., Mussa-Ivaldi, F. A., & Bizzi, E. (1993).

The length at which the load and muscle force balance is termed the equilibrium point (EP). The EPH is the hypothesis that the central signal changes the force-length characteristic and thus, influences the EP (Figure 1.4). It should be noted that the EP is not determined entirely by the central signal but the combination of the central signal and the external load. Feldman hypothesized that voluntary movements are produced by shifting EPs from an initial point to a desired point. In this way, posture and movement are controlled by the same mechanism.

There are several versions of the EPH. The λ hypothesis postulates that the force-length characteristic is modulated by changes in the threshold for the stretch reflex. This hypothesis avoids the high gain required by a servo hypothesis since the central signal does not directly specify the resulting muscle-length. It also implies a much lower delay than the servo hypothesis because the spinal cord's interpretation of the spindle activity is modulated as opposed to the spindle activity itself being modulated through the gamma-motor neurons. This results in a much shorter feedback loop.

The λ hypothesis can be formulated as

$$a(t) = c\left[e^{b(x(t)-\lambda(t))} - 1\right] \tag{1.3}$$

where t, represents time, b and c are constant parameters, a represents the muscle activation, x the muscle length, and λ the threshold for the stretch reflex. Here, λ is replacing s from Equation 1.1 and has a specific physiological meaning.

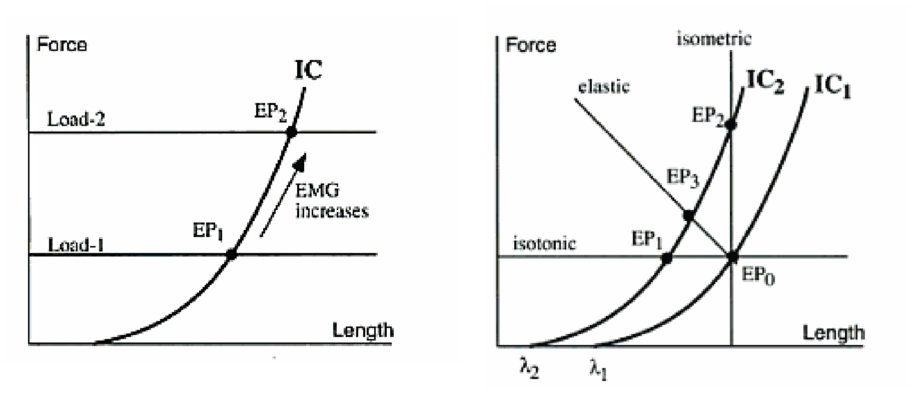

Figure 1.4 Left: invariant characteristic. Right: two different ICs are shown representing two different central commands in the EPH. The muscle length can be changed under a constant load by shifting the IC as is seen by comparing EP_1 to EP_0.
Source: Latash, M. L. Neurophysiological Basis of Movement.

Numerous studies have been conducted which purportedly disprove the EPH. The main paradigm for such studies involves subjects learning a movement with a certain external load and then subject repeating the movement with the load changing unexpectedly. Under the internal dynamics model (IDM), the brain develops a physical model of the system and through inverse kinematics computes the necessary muscle activations to complete the task. The central signal then directly specifies those muscle activations. The IDM predicts that when the load is unexpectedly changed, unless the central signal also changes, the muscle activations should remain the same. The EPH predicts that changes in the external load should lead to changes in the muscle length which should lead to changes in the muscle activation. A special situation is when the change in the external load is only transient in which case the movement should still terminate at the equilibrium point as the equilibrium point is determined by the final load and the central signal. The property of the movement to terminate in the same position regardless of transient changes in the load is termed equifinality. Thus, studies in which said conditions are met but equifinality is not observed serve as disproof of the EPH.

In one such study conducted by Hinder and Milner, subjects were tasked with moving a cursor on a screen to a target location via wrist flexion [5]. Their wrist flexion was assisted by a motor that produced assisting torque proportional to the angular velocity of their wrist. The action of the motor can be thought of as a negative damping force; since normally damping increases the stability of a system, the motor destabilized their movement resulting in oscillations around the endpoint. After many trials, subjects learned to reach the target but oscillations persisted. In later trials, occasionally the assisting torque of the motor was either reduced or completely eliminated. Hinder and Milner observed that without the assisting torque, subjects undershot the target without oscillation and the cumulative EMG was not significantly different (up until just before the oscillations would typically start) than it was with the assisting torque. This study shows a scenario in which it is evident that muscle activation is position independent and specified directly by the brain. However this study is curious in the fact that the oscillations were never eliminated despite subjects performing hundreds of trials. The oscillations are more expected in an EPH style control due to the fact that the dynamics of the combination of EPH and the assisting torque is akin to an underdamped nonlinear spring. If the brain directly specifies the muscle activations after building a model, one would expect the brain could simply compensate for the assisting torque by reducing the torque produced by the wrist and thus, eliminate the oscillations.

In another study conducted by Dizio and Lackner in 1994, Subjects were tasked

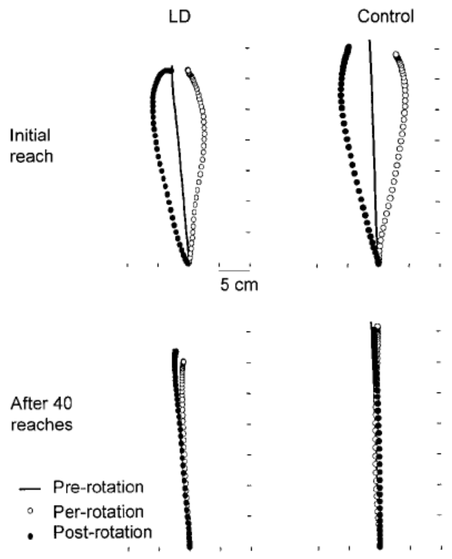

Figure 1.5 Average reaching movement paths for labyrinthine-defective (LD) and control subjects before(pre) during (per) and after (post) rotation.
Source: DiZio, P., & Lackner, J. R. 2001

with touching a target which was visible only before the onset of movement [2]. After performing this task 40 times, the subjects repeated the task again but while sitting in a dark room on a rotating platform. The rotation induced Coriolis forces on the arm which are proportional to the speed of the arms movement and thus, presented a transient perturbation. After 40 trials of the rotation condition, the subjects performed the task again without rotation. The subjects showed a lack of equifinality both when initially encountering the coriolis force, and immediately after the coriolis force was removed. Their trajectories are akin to what one might expect if the subjects were producing a strictly time-dependent force that was added to the externl forces independent of position (Figure 1.5).

In response to the Hinder and Milner study as well as the Dizio and Lackner study, Feldman, Ostry, and Levin, stated that the subjects central commands may have unintentionally changed due to the subjects detecting the changes in the dynamics [3]. Such a theory led Dizio and Lackner to include subjects in their experiments with labyrinthine defects that prevented them from feeling the rotation. Feldman, Ostry and Levin also postulate that the central signal simply has to change in these situations because of the fact that they are experiencing negative damping due to the environment. Indeed, it is difficult to know whether or not the central signal changes since it cannot at this time be directly measured. For this reason, experiments often involve the subject being instructed not to react to changes in dynamics, and also involve quirks such as removing a visual target just as a subject starts reaching for it. The idea here is that if the subject cannot see the target, there will be no online corrections made during the movement.

It is possible that there are multiple modes of control that can operate in parallel. While such a notion lacks the allure of simplicity that models such as the EPH have, it allows for the reconciliation of experimental results that seem to be at odds. The implication of such a possibility is that more needs to be done to identify when one mode of control is used over another so studies are not confounded by false premises.

In the present study, a single joint arm flexion with a slightly damped inertial load is tested. The goal of the task is to use a physical interface to move an object displayed on a screen to a target also on the screen. Since the goal object is virtual, it is theoretically possible to change the dynamics of the physical interface, while holding fixed the dynamics of the virtual interface. Instead of commanding subjects to “not react to a change”, subjects are not made aware of the change; instead of removing visual feedback, the visual feedback is manipulated to reinforce subjects’ lack of awareness of the change. The hope is that since the perturbation is not destabilizing, and since the subject is only concerned with the virtual object, their central signal should not change as long as the virtual object’s dynamics remain fixed. Thus, once the task is learned, the time-course of the subject’s central signal should remain the same but the location of their arm should be altered. If the time-course of the subject’s muscle activation is also altered, we can conclude that this is a paradigm in which muscle activations are position-dependent.

CHAPTER 2

METHODOLOGY

2.1 Admittance Control

Admittance control is a mode of human-robot interface in which a robotic end-effector simulates a relationship between the force applied by the human, $\boldsymbol{f_a}$, and the desired position of the end-effector $\hat{\boldsymbol{x}}$. Admittance controllers generally consist of three parts: a force sensor to measure the applied force, a computer that calculates the desired position of the end-effector at any time, and a lower level follower, e.g. a PID controller, to bring the actual position, $\boldsymbol{x}$, of the end-effector to the desired position. While there is technically a lag between the desired position and the actual position, if the follower is fast enough, this lag is negligent.

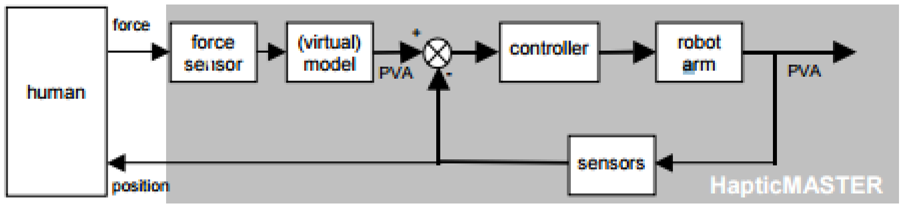

Figure 2.1 Haptic Master Control loop.
Source: Van der Linde, R. Q., et al.

A simple example of what can be achieved with an admittance controller is simulation of a frictionless weightless mass. To this end the force-position relationship used would be

$$\ddot{\hat{\boldsymbol{x}}} = \frac{\boldsymbol{f_a}}{m} \tag{2.1}$$

where $\ddot{\boldsymbol{x}}$ represents the second time derivative of $\boldsymbol{x}$, and m the mass being simulated. This can be modified to include a constant "gravitational" field by incorporating another term:

$$\ddot{\hat{\boldsymbol{x}}} = \frac{\boldsymbol{f_a}}{m} + \boldsymbol{a_g} \tag{2.2}$$

where $\boldsymbol{a_g}$ is the acceleration due to gravity.

Using admittance control devices allows the simulation of different physical scenarios that may never be encountered in real life. Such a paradigm is very valuable to motor control research for two reasons. Firstly, because any model of human motor production is in a sense "fit" to our observations of usual physical scenarios, but a good model should be able to make accurate predictions in a great variety of scenarios. Secondly, some studies, such as the current one take advantage of the physical scenario to shape the relationship between possible control variables and the motor goal.

The HapticMASTER is an admittance control robotic device that runs its admittance control at a rate of 2500 Hz. The force sensor of the HapticMASTER is located at the linkage of its end-effector allowing for any attachment to effectively control it.

2.2 Experimental Set-up

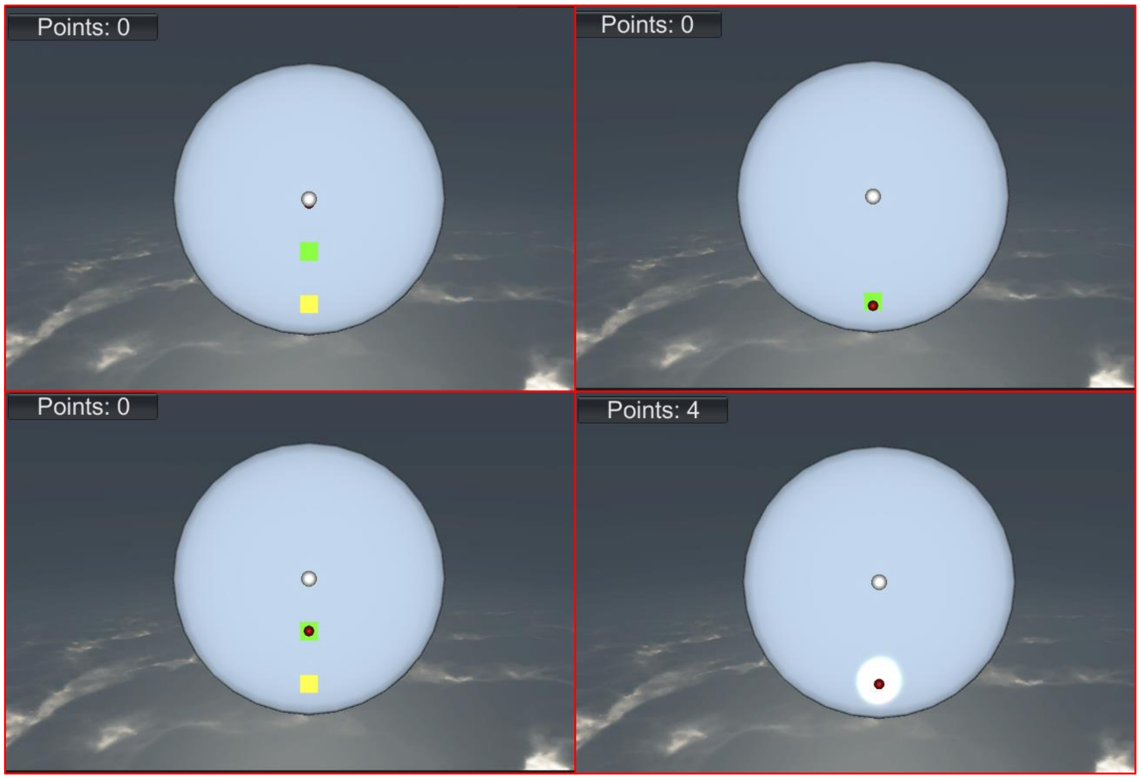

Figure 2.2 Screenshots of the visual environment.

The visual environment consisted of 4 entities on a screen: a round cursor, a round target in the center of the screen, a square target directly below the round target henceforth referred to as the "near target", and another square target directly below the near target henceforth referred to as the "far target". The three targets were all evenly spaced.

Subjects performed a task in which they moved the cursor from one target to the next in sequence. Each trial was initialized by the subjects bringing the cursor to the round target. This triggered the appearance of the near target in green as well as the far target in yellow. Subjects then moved the cursor to the near target. Once the target had come to a stop within the near target, following a short delay, the near target disappeared and the far target turned green. The far target color changed cued the subject to move the cursor to the far target. A programmatic stopwatch was also started at this time, the stopwatch was paused whenever the cursor was within the far target. If the cursor was moved into the full target and brought to a stop before the stopwatch reached 700ms, the trial was a success, this was indicated to the subject by the explosion of the target as well as the incrementing of a score in the top left of the screen. If the subject failed to reach the far target in time, the trial was unsuccessful as indicated by the mundane disappearance of the far target.

To control the cursor, subjects moved the end-effector of the HapticMASTER in the vertical direction using only flexion of their elbow. Subjects sat in a chair with their elbow placed on an elbow rest and grasped the HapticMASTER end-effector with their dominant hand. The angle of the subject's forearm was inferred from the position of the HapticMASTER end-effector. To this end, a calibration was done for each subject in which they performed several flexion and extension movements. The positions of the HapticMASTER during these movements was fit to a circle yielding the vertical location of the subjects elbow in the HapticMASTER's coordinates, as well as the length of the subject's lever arm which was the distance from their elbow to their proximal phalanges.

A linear relationship between the angle of the subject's forearm and the relative position of the cursor was maintained:

$$y = k\theta + \theta_0 \tag{2.3}$$

where θ was the elbow angle, y the relative position of the cursor, and k and θ_0 the parameters that defined their relationship, termed the scale and, offset respectively. There was a kinetic relationship between the cursor's position and the forces applied to the rotational system comprised of the subject's forearm and the HapticMaster's end effector imposed by the laws of rotational motion:

$$\ddot{y} = k\ddot{\theta} = \frac{k\tau}{I} \tag{2.4}$$

where τ is the total torque applied to the system, and I is the inertia of the system. The moments of inertia of subjects' arms, I_a were inferred from measurements and anthropometric ratios. The moment of inertia added by the HapticMASTER was given by

$$I_h = mr^2 \tag{2.5}$$

where m is still the virtual mass simulated by the HapticMASTER and r the subject's lever arm. Since the virtual mass could be controlled, the total inertia of the system could be manipulated as it was the sum of the two inertias:

$$I = I_a + I_h. \tag{2.6}$$

Unknown to the subject, on ~10% of trials, the virtual mass was increased by a factor of 1.25. This lead to a change in the moment of inertia by a factor of α which depended slightly on the subject but was always close to 1.18. Simultaneously, the cursor's scale was increased by the same factor. This yielded the modified Equations of motion:

$$\ddot{\theta} = \frac{\tau}{\alpha I} \tag{2.7}$$

and

$$\ddot{y} = \frac{\alpha k \tau}{\alpha I} = \frac{k\tau}{I}. \tag{2.8}$$

This change occurred at the same time as the color change of the far target. Since subjects were at rest at this point, there was no way for them to feel the change prior to the motion. Such trial are termed “loaded trials”.

The task was completed in sets of 20 trials each. Subjects each completed five sessions consisting of six sets each. To prevent muscle fatigue, subjects had a 30 second rest period in between each set. Loaded trials were absent until the second set or the set following the subject’s first set with at least 70% success. Loaded trials were chosen at random from the last 11 trials of each set which the stipulation that any two loaded trials had at least two standard trials in between them.

Two channels of EMG were recorded using a Delsys Bagnoli™ EMG system. The first channel was used to record EMG activity from the subject’s biceps and the other channel was used to record EMG activity from the subject’s triceps.

All subjects signed a consent form approved by the NJIT Institutional Review Board.

2.3 Parameter Determination

The amount of upwards force that needed to be applied at subjects' hands to counteract the torque due to gravity needed to be determined. The torque due to gravity depends on the angle of the elbow according to

$$\tau_g = m_a g r_a \cos(\theta) \tag{2.9}$$

where m_a is the mass of the subject's forearm, g, the acceleration due to gravity, r_a the effective radius of gyration of the subject's arm, and θ the angle between the subject's elbow and the horizontal plane supporting the elbow. The force needed to balance the torque is given by

$$f_{antigrav} = \frac{\tau_g}{\cos(\theta)\, r} = m_a g \frac{r_a}{r}. \tag{2.10}$$

This force was measured over the 15 degree range of motion used in the task. The force found was generally not constant but tended to be higher for greater angles. Applying the average force for some subjects proved to be adequate only for a small part of this range; at some portion of the range the force would either undercompensate or over compensate for gravity. Instead a force depending on the vertical displacement of the subject's hand was used. This position dependent force was able to completely eliminate the effect of gravity. The average force required to balance the arm was used to compute the average gravitational torque being affecting their arm. The radius of gyration of the subjects arms was computed as being located 82.7% of the distance between their elbow and ulnar styloid as given in anthropometric tables [12]. From the gravitational torque and the radius of gyration, the moment of inertia of the subject's arm was computed.

A damping force of $8Kg \cdot s$ was applied during standard trials. This force was scaled up by α during loaded trials.

CHAPTER 3

RESULTS

3.1 Torque Sign Change

Five naive subjects (4 right-handed 1 left-handed) were tested in this study. Subjects learned to anticipate the move cue which led to their movements typically having a smooth

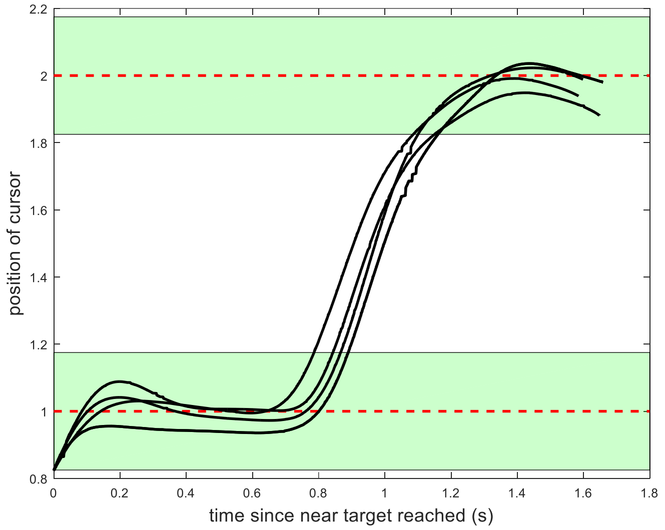

Figure 3.1 Example trajectories for subject 1. On the abscissa is the elapsed time since the cursor was brought inside of the target. On the ordinate is the position of the cursor. The targets are represented by the green boxes with their centers marked by the red dashed lines.

profile as in figure 3.1. On some trials, subjects began moving prior to the cue, this is evident in the cursor trajectory as in figure 3.2. Such trials were excluded from analysis. The first two trials of each set were also excluded. A total of 1368 out of 3600 trials were excluded.

The angular acceleration was computed for each trial and filtered with a low pass Butterworth filter with a cutoff frequency of 25 Hz. The time relative to the far target color change that the angular acceleration reached 0 was computed for each trial. This elapsed

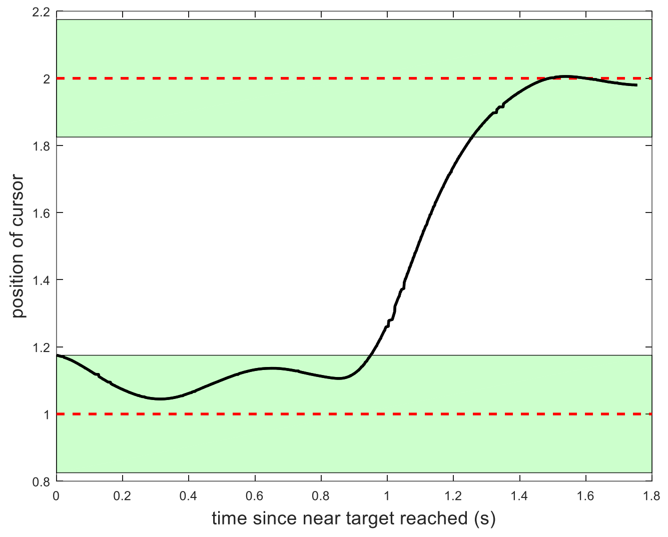

Figure 3.2 Example trajectory for subject 3. The trajectory near target is approached from below (above in the plot) instead of from below. This represents a reversal of movement direction after the subject moved to early.

time, henceforth referred to as the torque sign-change point (TSCP) was averaged for each subject over all trials. The mean TSCP was computed independently for loaded trials and standard trials. The difference between the mean TSCP for loaded trials and the mean TSCMP for standard trials was computed for each subject and is summarized in Table 2.1 A paired t test was performed on the mean TSCP revealing that there was a significant (p<0.005) increase in TSCP during loaded trials. Figure 3.3 shows the torques of the trial averages of the movements. For the preceding analysis trial averaging was not used prior to the computation of the TCSP but it is evident in figure 3.3 that the TCSP for the trial averaged loaded trial is delayed with respect to the TCSP trial averaged standard trial

Table 2.1 Torque Sign Change Points

Subject	Mean TSCP (±stdv) in seconds		Difference in mean TCSP (Loaded – standard)
	Standard trials	Loaded trials	
1	0.9322 (±0.0765)	0.9704 (±0.0904)	0.0382
2	1.0592 (±0.0901)	1.0789 (±0.0858)	0.0197
3	1.0156 (±0.0840)	1.0358 (±0.0848)	0.0203
4	1.0001 (±0.0835)	1.0118 (±0.0903)	0.0117
5	1.0979 (±0.0621)	1.1137 (±0.0451)	0.0158

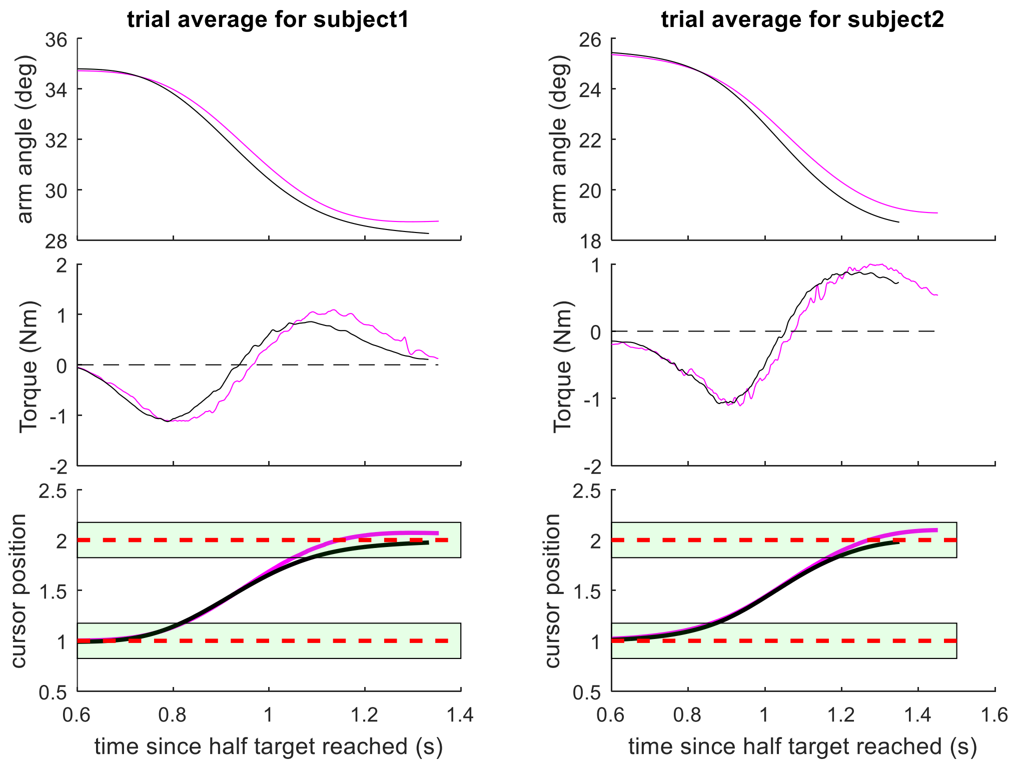

Figure 3.3a-b Trial averages of arm angle, torque, and cursor position. The trial average for standard trials is plotted in black while the trial average for loaded trials is plotted in magenta. It is evident that the torque crosses 0 later in the loaded trials.

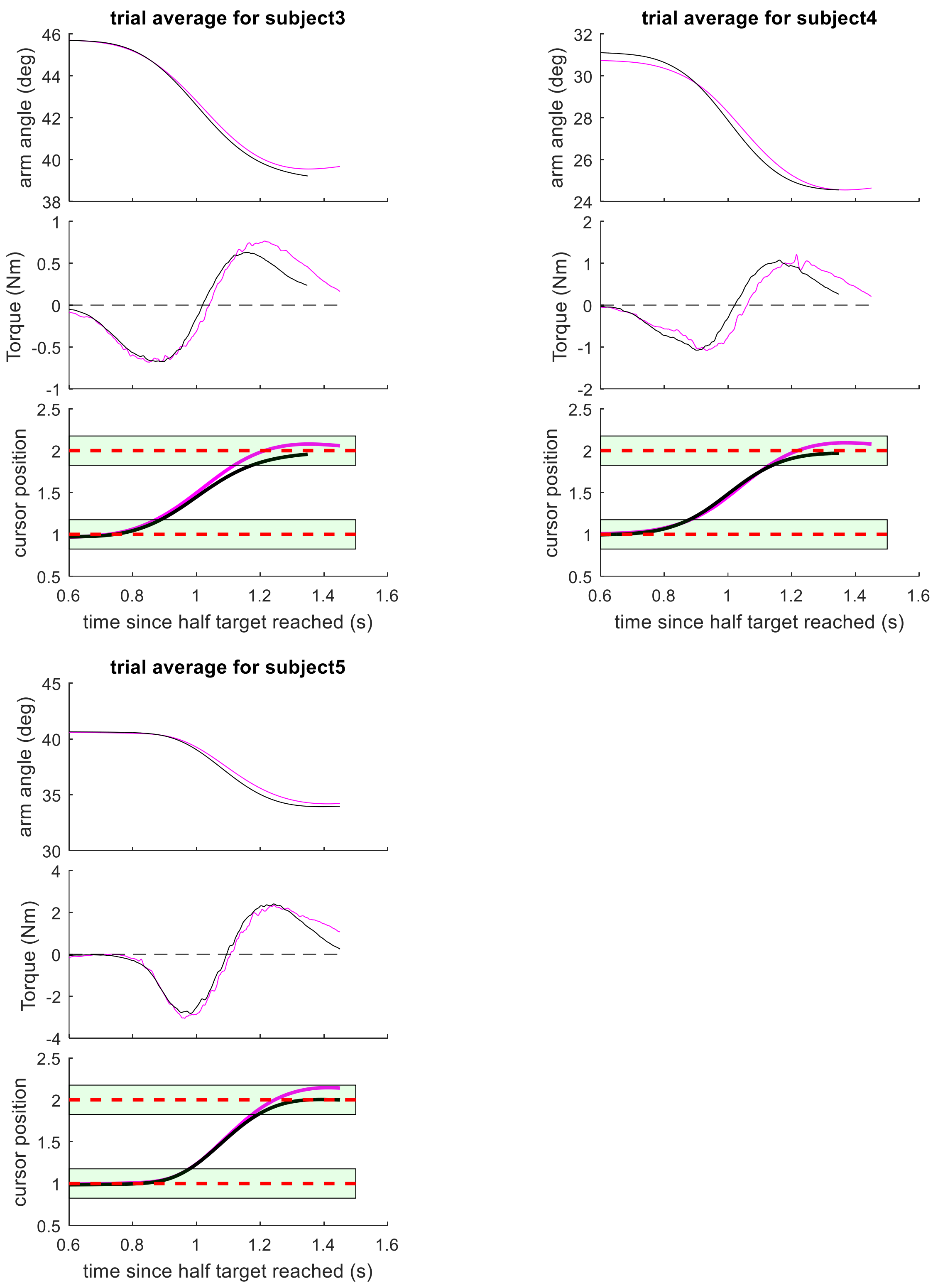

Figure 3.3c-e Trial averages of arm angle, torque, and cursor position. The trial average for standard trials is plotted in black while the trial average for loaded trials is plotted in magenta. It is evident that the torque crosses 0 later in the loaded trials.

3.2 Cross-Determination

A post-hoc analysis of how well the kinetic and kinematic data fit the prediction of three different position-dependent control models. The first model is an exponential spring model obtained by taking the λ hypothesis (Equation 1.3) and assuming that torque is constantly proportional to activation yielding

$$\tau(t + t_d) = c\left[e^{b\left(\theta(t)-\lambda(t)\right)} - 1\right], \tag{3.1}$$

where t_d represents a time delay. The next model is obtained by assuming that the produced torque is actually more akin to a linear spring due to the combined efforts of many different motor units. This takes the form

$$\tau(t) = c\left(\theta(t) - \lambda(t)\right). \tag{3.2}$$

The third model is a relative damping model similar to the one proposed in [10] given by

$$\tau(t + t_d) = c_s\left(\theta(t) - \lambda(t)\right) - c_d\left(\dot{\theta}(t) - \dot{\lambda}(t)\right). \tag{3.3}$$

It should be noted that all these models normally have absolute damping which is being neglected here. It was assumed that once the motion was learned, from one trial to the next, $\lambda(t)$ was the same. Thus, the parameters of these models could be fit by eliminating λ. This was achieved in the nonlinear λ hypothesis model by manipulating Equation 3.1 to obtain

$$\log\left(\frac{\tau_2(t+t_d)+c}{\tau_1(t+t_d)+c}\right) = b\big(\theta_2(t)-\theta_1(t)\big). \tag{3.4}$$

Likewise the linear spring model implies

$$\tau_2(t+t_d)-\tau_1(t+t_d) = c\big(\theta_2(t)-\theta_1(t)\big) \tag{3.5}$$

And the relative damping model implies

$$\tau_2(t+t_D)-\tau_1(t+t_d) = c_s\big(\theta_2(t)-\theta_1(t)\big) + c_d\big(\theta_2(t)-\theta_1(t)\big). \tag{3.6}$$

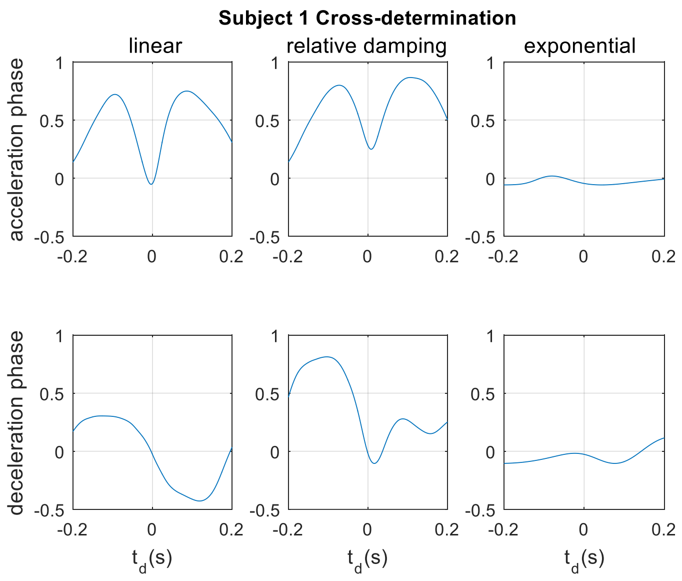

Figure 3.4a Cross-Determination for subject 1. Each axes shows the coefficient of determination plotted versus the time delay for the three position-dependent activation models. The acceleration and Deceleration phase of the movement is analyzed separately.

The parameters in Equations 3.4 - 3.6 can all be determined via least squares regression. The goodness of the fit can be determined by the coefficient of determination. However, spurious fits can be obtained and the time delay, t_d is unknown. For each subject and each model, the coefficient of determination was determined for a range of possible time delays. The coefficients of determinations for different time delays will be referred to as the cross-determination as it is an extension of a cross-correlation.

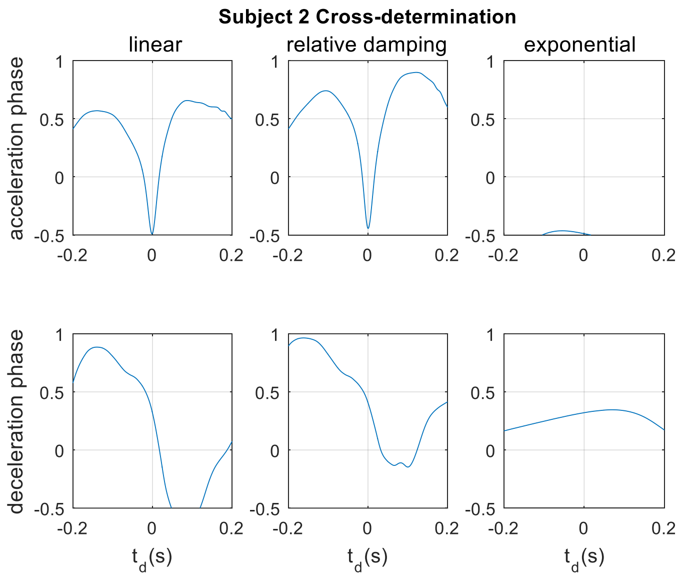

Figure 3.4b Cross-Determination for subject 2.

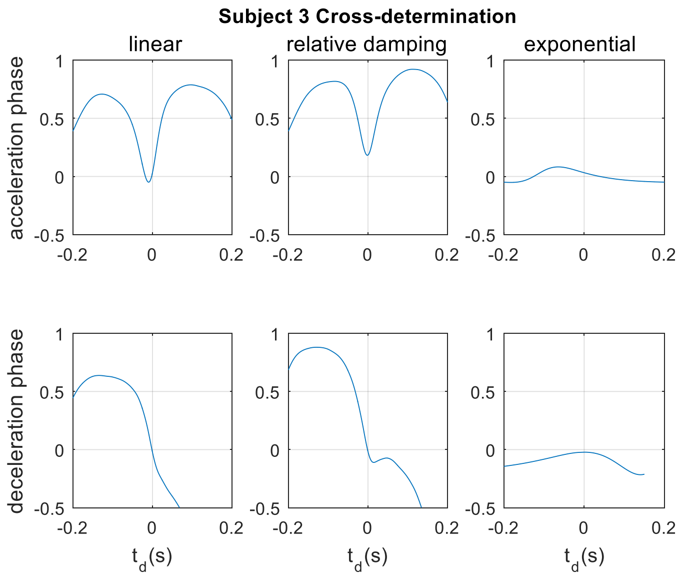

Figure 3.4c Cross-Determination for subject 3.

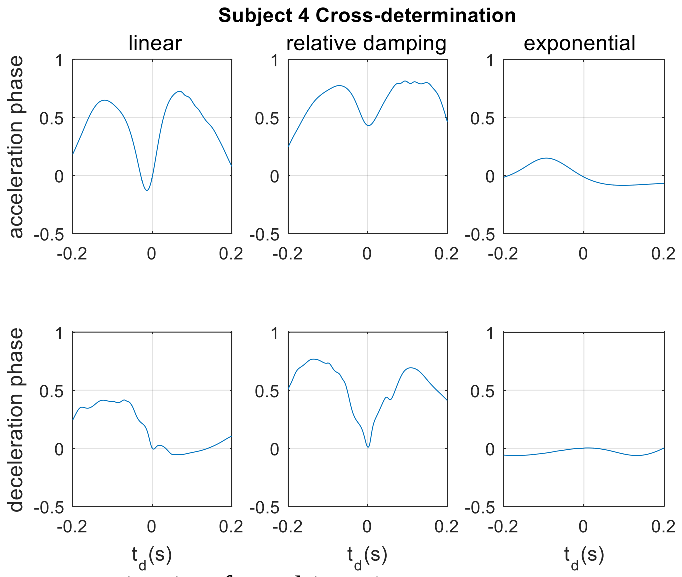

Figure 3.4d Cross-Determination for subject 4.

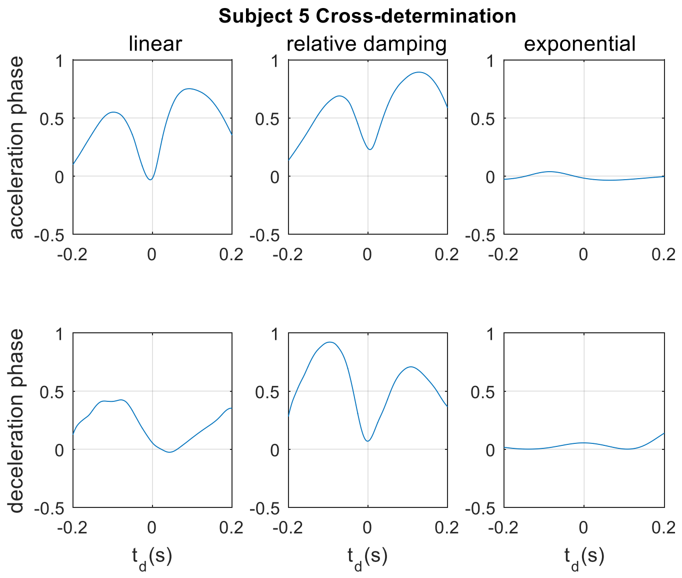

Figure 3.4e Cross-Determination for subject 5.

CHAPTER 4

DISCUSSION

4.1 Discussion of Torque Sign Change

It should be noted that subjects were informed about the loaded trials only after the experiment had concluded and all reported that they hadn't noticed the change in inertia during the experiment. The TSCP increase during loaded trials is in line with what would be expected in position-dependent control. For example, under the EPH, the lag in elbow angle caused by the increase in inertia would lead to the triceps remaining active longer than during standard trials. However, this result is not very strong on its own since the muscles are not the only producer of torque in the system. Particularly, a frictional torque that doesn't scale with the moment of inertia could lead to somewhat similar results.

A more useful measurement would be the duration of the tricep EMG burst during the movements. Unfortunately, the EMG collected proved to be very nonstationary during the course of the experiment and only a small fraction of the trials showed EMG activity that looked related to the task. Initially this was believed to be due to the task eliciting too little muscle activation to yield significant EMG. The parameters of the experiment, namely the starting inertia and the maximum of the movement time were manipulated during pilot trials to achieve a higher EMG amplitude. While these parameter changes showed a boost in EMG signal initially, the quality of the EMG signal tended to fade in and out during the course of many trials. One additional subject was tested with a higher inertia and slightly shorter movement time than previous subjects and unlike with previous subjects, five electrodes were used (two recording from the biceps and three from the triceps). This subject initially reported slight fatiguing. The EMG for this subject exhibited the same lack of stationarity as for the other subjects.

4.2 Discussion of Cross-Determination

In the computation of the coefficients of determination, a positive time delay corresponds to a comparison of kinematic variables (position, velocity, or exponential of position) to a future torque while a negative time delay corresponds to a comparison of kinematic variables to a past torque. Thus, a high coefficient of determination for a positive time delay implies that the kinematic variables linearly influence the torques produced afterwards. A high coefficient of determination for a negative time delay means that the torque has a linear influence on the kinematic variables that result afterwards. A coefficient of determination that is negative means that the model is even less well fit than a model that simply gives the average torque regardless of any other input. The exponential model generally fails to account for variance in future torques. This suggests that this model is invalid. For all five of the subjects, the linear spring model appears to explain 60-70% of the variance in torque around with a time delay of about 90ms. This would suggest that the linear spring model is somewhat valid. However this trend is only seen for the acceleration phase of the movement. The relative damping model explains variance in future torques in both the acceleration and deceleration phase for three of the subjects. This better performance could possibly be due to the model being more complex or due to it being more valid. The relative damping model also explains variance in past torques due to the linear relationship between acceleration and change of velocity.

Overall the study is inconclusive though its results point towards a position-dependent interpretation of motor signals. It should still be noted that it is possible that the motor system uses different strategies for different situations. More similar studies would need to be conducted to achieve a conclusive result.

4.3 Future Directions

The biggest flaw of this study is the lack of usable EMG data. A future project could aim at improving the quality of recorded EMG and re-conducting this same study. One possibility is that there was movement of the ground electrode.

Another change to the current study could be the timing mechanics. In this study, there was a pause between the subject reaching the near target and the far target color change subjects would prepare their motor command as much as possible prior to execution. Once the far target appeared, the timer was started to ensure that subjects' movements were quick enough to both elicit measureable EMG and prevent the viability of online corrections. However, subjects didn't always perfectly anticipate the color change of the far target. It would be better to have the timer start once the subject begins moving instead of once the target appears so that timing anticipation becomes less of a factor.

The use of cross-determination was done post-hoc in this experiment and thus, the results of the cross-determination have no weight unless they are reproduced in a future experiment. It is also possible that the apparent relationships between kinematics and torques were idiosyncratic to this particular movement trajectory. Such an analysis should be done on movements with different velocity profiles.

CHAPTER 5 APPENDIX

This appendix contain the Matlab and C# code used in the running of the experiment as well as part of the code used in the analysis, particularly that used to compute the cross-determinations.

A.1 Matlab Code for Task

```
function OyindaHapticCurl(baseHapticMass,handLength,cForce)
global g_position g_force g_timestamp g_posChannel g_trgtChannel
g_emgChannel g_flag1 g_h g_k g_thetaStart g_r

disp('running Haptic Curl')
%% Create Constants

targetWidth = .175; %proximity to target required to "hit" target
centerWidth = .15; %proximity to center required to "hit" center
inRange = 0; %flag for whether cursor is currently hitting full target
outOfBounds = 2.3;%distance corresponding to out of bounds

nReps = 20; %number of repetitions of same target before switching
targets

meanHold = .65; %mean time cursor needs to be in range of half target
before full target appears
hHoldRange = .25; %range of possible hold times
hHold = meanHold + hHoldRange*(rand-.5); %time cursor needs to be range
of half target
cHold = 0; %time cursor needs to be in range of center before half
target appears
fullHold = .75;%time cursor needs to be range of full target to score
% maxTime =1.75; %time alloted to reach full target to score

speedThreshold = 1;%half target isn't considered reached unless the
cursor speed is below this threshold


% sw is a list of the reps in which the scale should increase
n = 1;
sw = zeros(1,10);

if g_flag1
    while n<10;
        n=1;
        i = 9;
        while i<=60;
            if rand>=.6
                sw(n) = i;
                i = i+2;
                n = n+1;
                if n > 10
                    break
                end
            end
            i = i+1;
        end
    end
end


nScaleSwitch = 1;

antiGrav = hm_constant_force([0,0,cForce]);

%% Create record-keeping variables
targetType= 0; %0 if seeking center, 1 if seeking half target, 2 if
seeking full target

dataMax = 50000;% max number of samples

movementData = cell(1,nReps);%cell holding one data struct per movement

%initialize arrays for each movement
for i=1:nReps
    movementData{i}.theta = zeros(1,dataMax);
    movementData{i}.rawPosData = zeros(3,dataMax);
    movementData{i}.force = zeros(3,dataMax);
    movementData{i}.timeStamp = zeros(1,dataMax);
    movementData{i}.toc = zeros(1,dataMax);
    movementData{i}.scale = zeros(1,dataMax);
end


nMovement = 1; %number of reaching movement
j=1; %data sample index
nHit = 1; %index at which the target is reached


%% Dynamic Variables

rg = .827*(g_r-handLength); %radius of gyration of the arm as estimated
from anthropometric tables
armMass = cForce*g_r/rg;
armInertia = armMass*rg^2;


armLoadMassRatio = 4; %ratio of mass of medium load to arm mass
midHapticMass = max([armMass*armLoadMassRatio,3]);
midBaseInertiaRatio =
(midHapticMass*cForce+armInertia)/(baseHapticMass*cForce+armInertia);%r
atio of base rotational inertia to medium rotational inertia

hiMidMassRatio = 1.2;%ratio of mass of medium load to mass of high load
hiHapticMass = midHapticMass*hiMidMassRatio;
hiMidInertiaRatio =
(hiHapticMass*cForce+armInertia)/(midHapticMass*cForce+armInertia);

%ratio between change of arm angle and change of virual object position
baseScale = 3;
```

```
midScale = midBaseInertiaRatio*baseScale;
hiScale = hiMidInertiaRatio*midScale;


scale = baseScale;%start at low scale
offset = 0;%offset of position to allow smooth switching of scale

prevTheta = 0;%store 1 lag of virtual position to calculate velocity

repCount = 0;


%% Send data
fwrite(g_emgChannel,'begin') %start emg collection

tInRange = tic;
tBegin = tic;

while repCount < nReps;


    %calculate virtual position and send to unity
    thetaRaw = atan2(g_position(3)-g_k,-g_position(1)-g_h)-
g_thetaStart;
    theta = thetaRaw*scale+offset;


    posData = ['X0','Y',num2str(theta),'Z'];
    fwrite(g_posChannel,posData);

    spd = abs(prevTheta-theta); %cursor speed

    %record keeping

    movementData{nMovement}.theta(j) = theta;
    movementData{nMovement}.rawPosData(:,j) = g_position;
    movementData{nMovement}.force(:,j) = g_force;
    movementData{nMovement}.timeStamp(j) = g_timestamp;
    movementData{nMovement}.toc(j) = toc(tBegin);
    movementData{nMovement}.scale = scale;
    movementData{nMovement}.targetType = targetType;



    %CHECKING IF HALF TARGET IS REACHED
    if targetType==1
        if abs(theta-1)<targetWidth
            if toc(tInRange)>= hHold && spd<speedThreshold


                targetType = 2;%flag the full target is being sought
                fwrite(g_trgtChannel,'t2','char');% send full target to
unity


                %switch scales ~20% of the time
                if nScaleSwitch <= length(sw) && repCount+1 ==
sw(nScaleSwitch)

                    nScaleSwitch = nScaleSwitch+1;%increment number of
scale switches

                    scale = hiScale;%change scale
                    offset = (baseScale-hiScale)*thetaRaw;%offset
virtual position so that cursor doesn't jump


                    %switch inertia
                    if hiHapticMass<0.2
                        error('Specified mass is too small')
                    end

                    Test_Haptic2(hiHapticMass);

                    fwrite(g_trgtChannel,'ScaleSwitched','char')
                else
                    scale = midScale;
                    Test_Haptic2(midHapticMass);
                    offset = (baseScale-midScale)*thetaRaw;%offset
virtual position so that cursor doesn't jump

                end


                %start countdown to reach full target
                cumOutRange = 0;
                tOutRange = tic;

                movementData{nMovement}.halfReached = nHit;%store index
at which half target was reached
            end
        else
            nHit = j; %update the index at which the target was first
hit
            tInRange = tic; %restart timer if cursor isn't close enough
to target
        end


        %CHECKING IF FULL TARGET IS REACHED
    elseif targetType ==2
        if abs(theta-2)<targetWidth

            %update cummalative time out of range if necessary
            if ~inRange
                cumOutRange = cumOutRange+toc(tOutRange);
                inRange = 1;
            end

            if toc(tInRange)>= fullHold

                targetType = 0; %flag that the center is being sought
                fwrite(g_trgtChannel,'center','char');
                repCount=repCount+1;
                fwrite(g_emgChannel,'next');


                % set scale and offset for return to center
                scale = abs(theta/thetaRaw);
                offset = 0;


                movementData{nMovement}.success=1;%record movement as a
success
                nMovement = nMovement+1;%increment movement number
                j=0; %reset index for next movement


            end

            %reset timer for time out of range
            tOutRange = tic;
        else

            inRange = 0;

            %if time runs out or out of bounds
            if
toc(tOutRange)+cumOutRange>fullHold||abs(theta)>=outOfBounds

                targetType = 0; %flag that the center is being sought
                fwrite(g_trgtChannel,'centerF','char');
                repCount=repCount+1;
                fwrite(g_emgChannel,'next');


                % set scale and offset for return to center
                scale = abs(theta/thetaRaw);
                offset = 0;

                movementData{nMovement}.success = 0;%record movement as
a failure
                nMovement = nMovement+1; %increment movement number

                j=0; %reset index for next movement


            end
            tInRange = tic;
        end


        %CHECKING IF CENTER IS REACHED
    elseif targetType == 0
        if abs(theta)<centerWidth
            if toc(tInRange)>= cHold
                targetType = 1;
                fwrite(g_trgtChannel,'ht2','char');
                hHold = meanHold + hHoldRange*(rand-.5); %randomize
hold time


                %reset to low scale if necessary
                scale = baseScale;
                offset = 0;

                if baseHapticMass<0.2
                    error('Specified mass is too small')
                end

                Test_Haptic2(baseHapticMass);
                %xDamp = hm_damping([0,xDamping,0]);





            end


        else

            tInRange = tic;%reset timer for being in range
        end
    end



    prevTheta = theta; %update memory bank
    j=j+1; %update index

end

fwrite(g_emgChannel,'stop') %end emg collection

% %truncate data arrays
% for i=1:nReps
%     nLast = find(movementData{i}.time,1,'last'); %find last used
index
%     movementData{i}.theta = movementData{i}.theta(1:nLast);
%     movementData{i}.rawPosData =
movementData{i}.rawPosData(:,1:nLast);
%     movementData{i}.force = movementData{i}.force(:,1:nLast);
%     movementData{i}.timeStamp = movementData{i}.time(1:nLast);
%     movementData{i}.toc = movementData{i}.time(1:nLast);
%     movementData{i}.targetType = movementData{i}.targetType(1:nLast);
%
% end

saveas = SaveAs;
if ~strcmpi(saveas,'null')
save(['C:\Users\admin\Documents\MATLAB\work\Oyinda\SavedData\',saveas,'
.mat']);
fwrite(g_emgChannel,['saveas',saveas]);
end
```

```
hm_dcf(antiGrav);
```

A.2 C# Code for Task

```
using UnityEngine;
using System;
using System.Collections;
using System.Net;
using System.Net.Sockets;
using System.Text;

public class Effects : MonoBehaviour {

    //references to targets in order to display and hide them
    public GameObject
_ht1,_ht2,_ht3,_t1,_t2,_t3,_center,_tele1,_tele2,_tele3;

    //used to set position of cursor
    private Vector3 _pos;

    //used to set position and rotation of fireworks
    private Vector3 _expPos;
    private Quaternion _expRot;

    //used to hold fireworks object
    public GameObject _explosion;

    //used to indicate whether or not explosion is triggered
    private bool _expTrue;
    //used to indicate which targets are visible
    private string _centerActive="1", _htActive="0", _tActive="0";

    //used to flag if visible targets need to be updated
    private int _updateTargets=1;

    //keeps track of number of successful movements
    private int _score = 0;

    //Socket members
    private Socket _servSock1, _clientSock1, //for positions
    _servSock2, _clientSock2; //for targets

    private IPEndPoint _localPort1 = new
IPEndPoint(IPAddress.Parse("128.235.117.204"),4949),
    _localPort2 = new
IPEndPoint(IPAddress.Parse("128.235.117.204"),4950);

    private byte[] _buffer1,_buffer2;
    // Use this for initialization
    void Start ()
    {
        Application.runInBackground = true;
        StartServers();
        _expRot  = _t1.transform.rotation;

    }

    // Update is called once per frame
    void Update ()
    {
        //update cursor location
        transform.position = _pos;

        //instatiate explosion if necessary
        if (_expTrue)
        {
            Instantiate (_explosion, _expPos, _expRot);
            _expTrue = false;
        }

        //update displayed targets if necessary
        if (_updateTargets==1) {
            _updateTargets = 0;// unflag update targets

            //default all targets to invisible
            _ht1.SetActive(False);_ht2.SetActive (False);_ht3.SetActive
(False);
            _tele1.SetActive (False);_tele2.SetActive
(False);_tele3.SetActive(False);
            _t1.SetActive (False);_t2.SetActive (False);_t3.SetActive
(False);



            //turn flagged targets back on
            if (_centerActive=="1") {
                _center.SetActive (true);
            }else{
                _center.SetActive(False);
            }

            if (_htActive!="0") {
                switch (_htActive) {
                case "1":
                    _ht1.SetActive (true);
                    _tele1.SetActive(true);
                    break;
                case "2":
                    _ht2.SetActive (true);
                    _tele2.SetActive(true);
                    break;
                case "3":
                    _ht3.SetActive (true);
                    _tele3.SetActive(true);
                    break;

                }
            } else if (_tActive!="0") {
                switch (_tActive) {
                case "1":
                    _t1.SetActive (true);
                    _expPos = _t1.transform.position;
                    break;
                case "2":
                    _t2.SetActive (true);
                    _expPos = _t2.transform.position;
                    break;
                case "3":
                    _t3.SetActive (true);
                    _expPos = _t3.transform.position;
                    break;

                }
            }

        }

    }


    private void StartServers()
    {

        //open sockets
        _servSock1 = new Socket (AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
        _servSock2 = new Socket (AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);

        //bind ports
        _servSock1.Bind (_localPort1);
        _servSock2.Bind (_localPort2);

        //begin listening for connections
        _servSock1.Listen (0);
        _servSock2.Listen (0);

        Debug.Log ("listening");

        //begin accepting connections
        _servSock1.BeginAccept (AcceptCallback1, null);
        _servSock2.BeginAccept (AcceptCallback2, null);
    }

    private void AcceptCallback1(IAsyncResult AR)
    {
        _clientSock1 = _servSock1.EndAccept (AR);
        _buffer1 = new byte[1024];//create buffer
        _clientSock1.BeginReceive (_buffer1, 0, _buffer1.Length,
SocketFlags.None, new AsyncCallback (PositionReceivedCallback), null);
        Debug.Log ("Positions connection made");
    }

    private void AcceptCallback2(IAsyncResult AR)
    {
        _clientSock2 = _servSock1.EndAccept (AR);
        _buffer2 = new byte[1024];//create buffer
        _clientSock2.BeginReceive (_buffer2, 0, _buffer2.Length,
SocketFlags.None, new AsyncCallback (TargetsReceivedCallback), null);
        Debug.Log ("Targets connection made");
    }

    private void PositionReceivedCallback(IAsyncResult AR)
    {
        int received = _clientSock1.EndReceive(AR);//amount of received
data
        Array.Resize(ref _buffer1, received);//truncate buffer to used
portion
        //Debug.Log ("Checkpoint A");


        string pText= Encoding.ASCII.GetString (_buffer1);//interpret
received data as a string

        //find starts and ends of x and y data in string
        int xStart = pText.IndexOf ("X");
        int yStart = pText.IndexOf ("Y");
        int yEnd = pText.IndexOf ("Z");
        //Debug.Log ("Checkpoint B");

        //parse received positioin data if possible
        if (xStart >= 0 && yStart >= xStart+1 && yEnd >=yStart+1)
        {
            string xText = pText.Substring (xStart + 1, yStart - xStart
- 1);
            string yText = pText.Substring (yStart + 1, yEnd-yStart-1);
            //Debug.Log("Checkpoint C");



            //convert position from text to float
            float xPos = Convert.ToSingle (xText);
            float yPos = Convert.ToSingle (yText);

            //set position
            _pos.x = xPos;
            _pos.y = yPos;

            //Debug.Log("Checkpoint D");
        }
        //clear out buffer and begin receiving again
        _buffer1 = new byte[1024];
        _clientSock1.BeginReceive (_buffer1, 0, _buffer1.Length,
SocketFlags.None, new AsyncCallback (PositionReceivedCallback), null);

        if (received == 0)
        {
            _servSock1.BeginAccept (AcceptCallback1, null);//begin
accepting connections

        }
    }

    private void TargetsReceivedCallback(IAsyncResult AR)
    {

        int received = _clientSock2.EndReceive(AR);//amount of received
data
        Array.Resize(ref _buffer2, received);//truncate buffer to used
portion



            string targetList = Encoding.ASCII.GetString
(_buffer2);//get list of active targets as string


        //flag active targets for update thread
        if (targetList.IndexOf ("ht") == 0)
        {
            Debug.Log ("center reached");
            _updateTargets = 1;//flag targets for update
            _htActive = targetList.Substring (2, 1);
            _tActive = "0";
        } else if (targetList.IndexOf ("t") == 0) {
            Debug.Log ("half target reached");
            _updateTargets = 1;
            _tActive = targetList.Substring (1, 1);
            _htActive = "0";
        } else if (targetList.IndexOf ("center") == 0) {
            Debug.Log ("full target reached");
            _updateTargets = 1;
            _tActive = "0";
            _htActive = "0";

            if(targetList.IndexOf("centerF")!=0)
            {
                _expTrue = true;
                _score++;
            }

            if(targetList.IndexOf("reset")==0)
            {
                _score=0;
            }

        }


        //clear out buffer and begin receiving again
        _buffer2 = new byte[1024];
        _clientSock2.BeginReceive (_buffer2, 0, _buffer2.Length,
SocketFlags.None, new AsyncCallback (TargetsReceivedCallback), null);

        if (received == 0)
        {
            _servSock2.BeginAccept (AcceptCallback2, null);//begin
accepting connections
            Debug.Log ("listening");

        }

    }

    void OnGUI () {
        // Make a background box
        string score = _score.ToString ();

        GUIStyle scoreStyle = new GUIStyle ("button");
        scoreStyle.fontSize = 40;
        if (GUI.Button(new Rect(10,10,250,45), "Points:
"+score,scoreStyle))
            _score = 0;


    }





}
```

A.3 Matlab Code for Cross-Determination

```
ids = {'AA','AR','KK','YW','CM'};
warning off %#ok<WNOFF>


resampled = cell(20,1);

TR = 3; %amount of time to resample in seconds
rfs = 2000; %resampling frequency
tR = 0:1/rfs:TR; %query times
```

```
tStart = 0.65;
tEnd = 1.2;
nStart = ceil(tStart*rfs);
nEnd = floor(tEnd*rfs);


upAccThreshhold = .02; %minimum magnitude of acceleration to count for
upward phase
downAccThreshhold = -.02; %maximum acceleration to count for downward
phase
[B,A] = butter(2,50/(2*rfs));

%span of cross correlation
lagMin = -0.2;
lagMax = .2;
lags = round(lagMin*rfs):round(lagMax*rfs);

for idn = 1:5
    load([ids{idn},'keep'])
    nNormal = 0;
    nCatch = 0;

    normalTrialTheta = zeros(1,length(tR));
    catchTrialTheta = zeros(1,length(tR));
    normalTrialForce = zeros(3,length(tR));
    catchTrialForce = zeros(3,length(tR));
    normalTrialPos = zeros(3,length(tR));
    catchTrialPos = zeros(3,length(tR));
    normalTrialRawTheta = zeros(1,length(tR));
    catchTrialRawTheta = zeros(1,length(tR));


    upSsr1 = 0*lags;
    upSsr2 = 0*lags;
    upSsr3 = 0*lags;
    downSsr1 = 0*lags;
    downSsr2 = 0*lags;
    downSsr3 = 0*lags;
    upSst1 = 0 *lags;
    upSst2 = 0 *lags;
    upSst3 = 0 *lags;
    downSst1 = 0 *lags;
    downSst2 = 0 *lags;
    downSst3 = 0 *lags;


    %%
    for nSes = 1:5
        for nTrial = 1:6
            try

load(['C:\Users\Me\Documents\MATLAB\Research\',ids{idn},'\',ids{idn},nu
m2str(nSes),'_',num2str(nTrial)]);
                baseInertia = armInertia+baseHapticMass*g_r;
                highInertia = armInertia + hiHapticMass*g_r;
            catch
                continue
            end
            for nRep = 8:20
                if goodTrials(nSes,nTrial,nRep) &&
(goodTrials(nSes,nTrial,nRep-1) || (nRep<20 &&
goodTrials(nSes,nTrial,nRep+1)))
                    %remove repetitions
                    [~,uI,rI] =
unique(movementData{nRep}.timeStamp,'stable');
                    movementData{nRep}.toc =
movementData{nRep}.toc(uI);
                    movementData{nRep}.rawPosData =
movementData{nRep}.rawPosData(:,uI);
                    movementData{nRep}.timeStamp =
movementData{nRep}.timeStamp(uI);


                    %trim arrays
                    nLast = find(movementData{nRep}.toc,1,'last');
%find last used index
                    movementData{nRep}.rawPosData =
movementData{nRep}.rawPosData(:,1:nLast);
                    movementData{nRep}.timeStamp =
movementData{nRep}.timeStamp(1:nLast)*0.001; %adjust timestamps to
seconds
                    movementData{nRep}.toc =
movementData{nRep}.toc(1:nLast);
                    %movementData{nRep}.scale =
movementData{nRep}.scale(1:nLast);


                    %resample
                    nOnset = rI(movementData{nRep}.halfReached);
                    tOnset = movementData{nRep}.timeStamp(nOnset);

                    rawPos = interp1(movementData{nRep}.timeStamp-
movementData{nRep}.timeStamp(nOnset),movementData{nRep}.rawPosData',tR)
';
                    rawTheta = pi/180*asin((rawPos(3,:)-g_k)/g_r);


                    %calculate accelerations
                    startingVal = median(rawTheta(1:ceil(rfs*.01)));
                    filtTheta = filter(B,A,rawTheta-
startingVal)+startingVal;
                    vel = diff(FiltTheta)./diff(tR);
                    acc = diff(vel)./diff(tR(2:end));
                    resampledData{nRep}.acc = filter(B,A,acc-
acc(1))+acc(1);
                    resampledData{nRep}.rawTheta = rawTheta;
                    resampledData{nRep}.vel = vel;

                    %compute torques
                    if any(sw == nRep)
                        resampledData{nRep}.Torque =
highInertia*resampledData{nRep}.acc;
                    else
                        resampledData{nRep}.Torque =
baseInertia*resampledData{nRep}.acc;
                    end


                end


            end




            for nRep = 9:20
                if goodTrials(nSes,nTrial,nRep-1) &&
goodTrials(nSes,nTrial,nRep) && any(sw==nRep)

                    upTorqueRatio = zeros(nEnd-nStart,1);
                    upAngleDiffLag = zeros(nEnd-nStart,length(lags));
                    upAngleDiff = zeros(nEnd-nStart,1);
                    upVelDiff = zeros(nEnd-nStart,1);
                    upTorqueDiffLag = zeros(nEnd-nStart,length(lags));
                    downTorqueRatio = zeros(nEnd-nStart,1);
                    downVelDiff = zeros(nEnd-nStart,1);
                    downTorqueDiffLag = zeros(nEnd-
nStart,length(lags));
                    downAngleDiffLag = zeros(nEnd-nStart,length(lags));
                    downAngleDiff = zeros(nEnd-nStart,1);




                    kUp = 0;
                    kDown = 0;

                    for i = nStart:nEnd
                        if resampledData{nRep}.acc(i) > upAccThreshhold
&& resampledData{nRep-1}.acc(i) > upAccThreshhold
                            kUp=kUp+1;
                            upVelDiff(kUp) = resampledData{nRep-
1}.vel(i)-resampledData{nRep}.vel(i);
                            upTorqueRatio(kUp) =
log(resampledData{nRep-1}.Torque(i)/resampledData{nRep}.Torque(i));
                            upAngleDiff(kUp) = resampledData{nRep-
1}.rawTheta(i)-resampledData{nRep}.rawTheta(i);
                            for m = lags
                                upTorqueDiffLag(kUp,m-lags(1)+1) =
resampledData{nRep-1}.Torque(i+m)-resampledData{nRep}.Torque(i+m);
                                upAngleDiffLag(kUp,m-lags(1)+1) =
resampledData{nRep-1}.rawTheta(i-m)-resampledData{nRep}.rawTheta(i-m);

                            end
                        elseif resampledData{nRep}.acc(i) <
downAccThreshhold && resampledData{nRep-1}.acc(i) < downAccThreshhold
                            kDown=kDown+1;
                            downTorqueRatio(kDown) =
log(resampledData{nRep-1}.Torque(i)/resampledData{nRep}.Torque(i));
                            downVelDiff(kDown) = resampledData{nRep-
1}.vel(i)-resampledData{nRep}.vel(i);
                            downAngleDiff(kDown) = resampledData{nRep-
1}.rawTheta(i)-resampledData{nRep}.rawTheta(i);
                            for m = lags
                                downTorqueDiffLag(kDown,m-lags(1)+1) =
resampledData{nRep-1}.Torque(i+m)-resampledData{nRep}.Torque(i+m);
                                downAngleDiffLag(kDown) =
resampledData{nRep-1}.rawTheta(i)-resampledData{nRep}.rawTheta(i);
                            end
                        end
                    end

                    up1 = 0*lags;
                    up2 = 0*lags;
                    up3 = 0*lags;

                    downSsr1 = 0*lags;
                    downSsr2 = 0*lags;
                    downSsr3 = 0*lags;
                    if kUp>0

                        upTorqueRatio = upTorqueRatio(1:kUp);
                        upTorqueDiffLag = upTorqueDiffLag(1:kUp,:);
                        upAngleDiffLag = upAngleDiffLag(1:kUp,:);
                        upAngleDiff = upAngleDiff(1:kUp);
                        upVelDiff = upVelDiff(1:kUp);


                        [ssr,sst] = getSS(upTorqueDiffLag,upAngleDiff);
                        upSsr1 = upSsr1+ssr;
                        upSst1 = upSst1+sst;

                        [ssr,sst] =
getSS(upTorqueDiffLag,[upAngleDiff,upVelDiff]);
                        upSsr2 = upSsr2+ssr;
                        upSst2 = upSst2+sst;

                        for i = 1:length(lags)
                            if(all(isfinite(upAngleDiffLag(:,i))))
                            [ssr,sst] =
getSS(upTorqueRatio,[upAngleDiffLag(:,i),ones(size(upTorqueRatio))]);
                            upSsr3(i) = upSsr3(i)+ssr;
                            upSst3(i) = upSst3(i)+sst;
                            end
                        end


                    end
```

```
                    if kDown>0

                        downTorqueRatio = downTorqueRatio(1:kDown);
                        downTorqueDiffLag = 
downTorqueDiffLag(1:kDown,:);
                        downAngleDiffLag = downAngleDiffLag(1:kDown,:);
                        downAngleDiff = downAngleDiff(1:kDown);
                        downVelDiff = downVelDiff(1:kDown);



                        [ssr,sst] = 
getSS(downTorqueDiffLag,downAngleDiff);
                        downSsr1 = downSsr1+ssr;
                        downSst1 = downSst1+sst;

                        [ssr,sst] = 
getSS(downTorqueDiffLag,[downAngleDiff,downVelDiff]);
                        downSsr2 = downSsr2+ssr;
                        downSst2 = downSst2+sst;

                        for i = 1:length(lags)
                            if(all(isfinite(downAngleDiffLag(:,i))))
                            [ssr,sst] = 
getSS(downTorqueRatio,[downAngleDiffLag(:,i),ones(size(downTorqueRatio)
)]);
                            downSsr3(i) = downSsr3(i)+ssr;
                            downSst3(i) = downSst3(i)+sst;
                            end
                        end

                    end

                end
            end
        end

    end

    figure
    subplot(2,3,1)
    plot(lags/rfs,upSsr1./upSst1)
    subplot(2,3,2)
    plot(lags/rfs,upSsr2./upSst2)
    subplot(2,3,3)
    plot(lags/rfs,upSsr3./upSst3)
    title(ids{idn})

    subplot(2,3,4)
    plot(lags/rfs,downSsr1./downSst1)
    subplot(2,3,5)
    plot(lags/rfs,downSsr2./downSst2)
    subplot(2,3,6)
    plot(lags/rfs,downSsr3./downSst3)
    save([ids{idn},'correlations'])

end
function [sse, sst] = getSS(y,x)

w = x\y;

e = (x*w-y);
sse = sum(e.^2,1);
sst = var(y,0,1)*size(y,1);

sse(isnan(sse)) = 0;
sst(isnan(sse)) = 0;
```

REFERENCES

[1] de Lussanet, M. H, Smeets, J. B & Brenner, E. "Relative damping improves linear mass-spring models of goal-directed movements." Human movement science 21.1 (2002): 85-100.

[2] DiZio, P., & Lackner, J. R. "Coriolis-force-induced trajectory and endpoint deviations in the reaching movements of labyrinthine-defective subjects." Journal of Neurophysiology 85.2 (2001): 784-789.

[3] Feldman, A. G., et al. "Recent tests of the equilibrium-point hypothesis (lambda model)." Motor Control 2.3 (1998): 189-205.

[4] Giszter, S. F., Mussa-Ivaldi, F. A., & Bizzi, E. Convergent force fields organized in the frog's spinal cord. The Journal of Neuroscience 13(2) (1993): 467-491.

[5] Hinder, M. R., & Milner, T. E. The case for an internal dynamics model versus equilibrium point control in human movement. The Journal of Physiology, 549(3), (2003): 953-963.

[6] Latash, M. L. Neurophysiological Basis of Movement. Champaign, IL: Human Kinetics, 2008. Print.

[7] Lemay, M. A., & Grill, W. M. "Endpoint forces evoked by microstimulation of the cat spinal cord." [Engineering in Medicine and Biology, 1999. 21st Annual Conference and the 1999 Annual Fall Meeting of the Biomedical Engineering Society] BMES/EMBS Conference, 1999. Proceedings of the First Joint. Vol. 1. IEEE, 1999.

[8] McIntyre, J., & Bizzi, E. Servo hypotheses for the biological control of movement. Journal of Motor Behavior, 25(3), (1993): 193-202.

[9] Shadmehr, R. "The equilibrium point hypothesis for control of movement." Baltimore, MD: Department of Biomedical Engineering, Johns Hopkins University (1998).

[10] Tresch, M. C., & Bizzi, E. Responses to spinal microstimulation in the chronically spinalized rat and their relationship to spinal systems activated by low threshold cutaneous stimulation. Experimental Brain Research, 129(3), (1999): 401-416.

[11] Van der Linde, R. Q., et al. "The HapticMaster, a new high-performance haptic interface." Proc. Eurohaptics. 2002.

[12] Winter, D. A. *Biomechanics and motor control of human movement*. Hoboken, NJ: John Wiley & Sons, 2009. Print.