A Biologically Motivated Paradigm for Heuristic Motor Control in Multiple Contexts

by

Max Berniker

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Masters of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2000

Author ....
Department of Mechanical Engineering
August 4, 2000

Certified by... [illegible] ....
Neville Hogan
Professor
Thesis Supervisor

Accepted by .... [illegible] ....
Ain Sonin
Chairman, Department Committee on Graduate Students

A Biologically Motivated Paradigm for Heuristic Motor Control in Multiple Contexts

by

Max Berniker

Submitted to the Department of Mechanical Engineering
on August 4, 2000, in partial fulfillment of the
requirements for the degree of
Masters of Science in Mechanical Engineering

Abstract

A biologically motivated controller is presented and investigated in terms of motor control and motor learning, especially motor learning of different behaviors. The controller, based on several ideas presented in the literature, used multiple paired inverse and forward internal models to generate motor commands and estimate the current environment within which the controller operated. The controller demonstrated an ability to control a simulated arm-like manipulator making $20cm$ reaching movements with an RMS position error of $4.6mm$ while making unperturbed movements, and $6.40mm$ while making the same reaching movements within a curl field. However, the ability to learn and retain multiple motor behaviors was proven not to be possible with the current paradigm and its learning algorithm. The results of the investigation were viewed in the scope of motor control for engineering purposes, and for use in developing a model for human motor control to further the science of neuro-recovery.

Thesis Supervisor: Neville Hogan
Title: Professor

Acknowledgments

I would like to start by thanking my advisor, Neville Hogan. I am very grateful for all that I have learned from him. His analytical skills, technical knowledge and practical know-how combine to rattle off insightful gems at a machine gun pace -it's something to behold -it's inspiring. Truly, it was an honor to work under him.

As always, in life as well as work, it's the people you share it with that make for especially memorable and valuable experiences, ...so I'd like to give a shout-out to all my hommies in the mezzanine! I'd like to thank Steve, one sharp kid who taught me the correct use of a semicolon; it's the little comma looking-thing you place between to complete sentences. I'd like to thank Brandon, for always allowing me to use him as a sounding board. His sage and thoughtful advice were always helpful. I'd like to thank Brian for enlightening me as to the ways of the farm (and big-city dating techniques). And Patrick and Keng, despite their spoiled Hunter lab background, are a welcome addition to any up-and-coming boy band (373K baby!).

I would also like to thank the members of Neville's research group, Dustin, Kris, Jerry and especially Lori and Igo, for all their support in my studies.

Of course, I must thank my team, "Newman's Neanderthanls", a rag-tag band of misfits that I singlehandedly whipped into an unstoppable hockey machine! John, Peter, James, Bobby D., Tanya, and our golden boys, Verdi and Matt. They would make any captain glow with pride.

I'd also like to thank one of our fallen mezzanine members, Luke Sosnowski, no longer with us, he will forever work in the mezzanine of our hearts.

I owe a great debt to Robin (you know who you are). Thanks honey-bunch.

And finally, I would like to thank the lovely ladies of the mechanical engineering graduate office, Leslie Regan and Joan Kravit. I believe that they, more than anyone else in the department, are responsible for making graduate studies at MIT a more "human" experience.

This work was supported in part by NIH Grant R01-HD37397 and by the Winifred Masterson Burke Medical Research Institute.

for mom and dad and chris

Contents

List of Figures

Chapter 1

Introduction

Science and technology exist in such highly evolved states today as to approach most technical problems in exceedingly complex manners. But perhaps more elegant solutions exist –solutions that do not rely on precise physical representations and brute force computation, but rather insight into natural processes that have already skillfully met the challenge. Many of the technical problems we face have parallels in nature. Biomimicry and a search for parallels in nature for these technical problems are gaining a broader acceptance. By inspecting nature for similar problems and examining the solutions to these natural systems, an otherwise untapped source of inspiration for strategies in our technical solutions are offered. A wise choice in confronting engineering problems might be to seek out appropriately similar natural systems; do they have similar goals and/or objectives to our current problem? By examining the solutions to these parallel systems we can identify what, if any, characteristics these natural systems have that might be practical or pragmatic for emulating.

From an engineering standpoint people represent excellent control systems. In homage to the competence of this natural machinery they have been emulated time and time again in one form or another, from factory floor automation that bears a striking functional resemblance to a human arm, to neural networks that mimic human minds in their calculations, to robots, made in man's image.

How do these man-made machines that seek to outdo their biological counterparts compare with humans and other biological entities? Certainly they are marvelous tributes to science and the technology that drives it. Generally speaking, machines can be built for a particular task to move faster, stronger, longer and more accurately than any person or animal ever will. Yet they usually are built to perform this very specific task. Their ability to perform outside the scope of their intended context is limited. Often a great deal of *a priori* knowledge is required as well. The machine must be given rules and guidelines for all the decisions it will be faced with concerning its task and the environment therein. If and when the machine's knowledge of its environment is lacking, failure is likely.

Humans on the other hand are endowed with an enormous capacity for varying tasks. Grasping, reaching, walking, running, dancing; the list of behaviors humans can exhibit goes on without end. An equally obvious observation is the ability we possess to quickly adapt to unknown tasks by generalizing previously learned knowledge and skills; learn to juggle and one can easily conquer the task while walking as well. The capacity for retrieving learned behaviors when confronted with unknowns and adapting in a timely and stable fashion is so commonplace, indeed so second nature, that one barely takes note of the remarkable motor control and motor learning involved.

This is where we as humans tip the scales on performance next to our man-made machines and the intelligence (or lack thereof) that controls them. Our aptitude for interacting through our bodies with unknown environments or contexts, as they shall be referred to, is a challenge that is poorly met by our man-made counterparts. These contexts represent a continuum from the completely unknown to the completely known. This is testament to the robust motor control and motor learning abilities that humans are equipped with.

Reaching for an object is no difficulty for a normally functioning human. A robotic arm can be made to execute the same task. Depending on the level of performance desired and the knowledge required for the task, a considerable amount of time and effort might be devoted to computing the proper chain of events that would lead to a successful grasp. When reaching for the same object through a viscous fluid, highly nonlinear forces difficult to quantify will perturb your arm. You are compelled to compensate. Yet again the task represents little to no difficulty for a human. How does the robotic arm fare? The same task, yet the environment or the context within which the robot arm is operating has changed. The required actions necessary will be considerably different for the robotic arm now. This same task performed in a different environment will be referred to as a new behavior. For the robotic arm the two behaviors represent interactions within different contexts that require different sets of rules. It would further require some way of determining when to act according to which set. Humans must also cope with these problems, problems of motor learning, though apparently without difficulty.

How does the human motor control system work? It is not completely understood. There is a great deal of evidence to support the claim that humans generate the actions necessary for motor control based solely on experience and models built of this experience. In some sense it could be argued that one does not require extensive intelligence if one has an extensive memory. Internal models learned through empirical data can in theory allow an empty substrate to gain a working picture of its environment without any *a priori* knowledge. Simply through an elaborate scheme of trial and error, models can be built that help to describe unknown processes. For instance, if one merely remembered every task performed and all the necessary bits of information that one would require to reconstruct that task (the actions taken and the resulting outcomes) then new tasks could be adapted to according to previously experienced ones. These learned internal models allow for informed actions and insight when encountering unknown contexts.

So it appears nature has provided an inspirational example for motor control and motor learning in ourselves. Whether or not this biological system will prove beneficial to emulate remains to be seen. The first step though is to investigate how we manage these worthy qualities, motor control and learning. With the human system as a prototype, control systems may be designed that complement human control aptitudes with current control theory for increasingly sophisticated technical solutions.

The benefits of emulating human-like motor control and motor learning may not be limited to better control systems. With a clearer understanding of human motor control and motor learning we are in a position to make more effective decisions concerning these aspects with regard to our health. With human motor control understood in the context of engineering control systems, well traveled analysis tools may be employed for insight into the human system and any problems that may arise within it. This could be beneficial not only in the treatment of human pathologies, but also for extending normal human capacities.

Recently studies have provided evidence to suggest that robot-aided therapy can be of assistance in the rehabilitation of individuals with neurologic injuries. It is an auspicious augmentation to the time-honored rehabilitation process of a therapist and patient manually working the impaired limbs and limb segments. It is not unreasonable to suspect that, with a deeper understanding of the mechanisms behind human motor control, this rehabilitation process can be made more effective.

A model of the human motor system would be useful in this regard. The model might elucidate the processes that underly the human system. For instance, a prominent question concerning human motor control is whether or not recovery (from neurologic injury effecting motor control) is similar to motor learning. With such a model this question could be addressed. Similarly, different rehabilitation procedures could be quickly evaluated in terms of their efficacy and expediency by determining how they would interact with a model that emulates human motor control.

This thesis seeks to explore current theories concerning human motor control and motor learning in an effort to emulate these mechanisms. The goal is to examine a biologically motivated controller as a means to intelligent, modular and hierarchical control with a heuristic approach to learning multiple behaviors and a method for retrieving them when appropriate.

The controller consists of multiple paired, forward and inverse models. Forward models help to estimate the current context within which the controller is interacting. Inverse models are used to generate motor commands. With multiple paired models the controller should be able to store learned behaviors for different motor tasks and still adapt to new behaviors without sacrificing those previously learned. Utilizing neural networks the models are built upon the experience gained through interacting within different contexts. A method for switching between the learned behaviors based on sensory feedback is presented. The controller is used to mimic the movements of a human arm. To this end a human-like arm is simulated with anthropomorphic data and anatomical principles.

Two questions will be addressed: Is the controller a useful control alternative? And does the controller shed light onto the field of human motor control? In order to explore the utility of this control system, an experiment was devised that would investigate its ability to learn and retain new behaviors, the prudence to call upon this information when appropriate, and the insight to use these tools in unknown situations. Should the controller exhibit these abilities, it will have proven to display similar qualities as the human motor system. To address the question concerning human motor control, the results of a similar experiment conducted with human subjects will be compared with the results found here in order to draw conclusions about the controller and its possible similarities with its human counterpart. Should the controller exhibit these similarities, it can be investigated further to help analyze human motor control.

In the chapters that follow pertinent information for the motivation of the research work will be presented. Chapter two will discuss what is thought to underly the human motor control system in an effort to better describe the approaches taken in this thesis. Chapter three is a brief introduction to neural networks, key to implementing the controller. Chapter four presents the simulated physical apparatus upon which the experiments are conducted. Chapter five will present the control paradigm under investigation. Chapter six will explain the experiment tasks to be completed and the remaining chapters shall cover the experiments conducted and the results found.

Chapter 2

Human Motor Control

This thesis is concerned with the mechanisms by which humans generate motor control and motor learning in terms of conventional control systems engineering. With that objective in mind the next step should be to analyze human motor control. This poses a respectable problem. Psychophysical systems, including the human one, are notoriously difficult to analyze in a traditional engineering sense. Even the simplest of behaviors involve a great number of nonlinearities and redundancies that make characterization intractably difficult. For instance, the act of maintaining an arbitrary posture, arguably the most fundamental of human motor behaviors, can represent a staggering number of modeling difficulties. Take the human arm, seven degrees of freedom (not counting the hand) making it kinematically redundant. Over 26 muscles actuate the arm, each highly nonlinear and exceedingly difficult to parameterize. The forces they impose on the skeleton depend on the configuration of the arm. Communication between the central nervous system and these muscles is conducted through a great number of descending and ascending neural pathways. Where does one begin? Attempting to model the complete system would be an effort made in vain. The conclusions drawn from such a highly complex model would be dubious at best.

In contrast to the problems facing this bottom up approach to analyzing the human motor system, taking a top down approach has made great strides. By examining the large picture one does away with the need to model and analyze the finer details.

Instead the focus is broadened and notice is brought to bear upon global features. Concerning human motor control, a wise course taken in the past has been to observe natural human phenomena for patterns.

2.1 Organizing Models: A Search For Invariants

Throughout the great majority of normal human movements there are several patterns. Work done by Morasso [25] in analyzing human reaching movements revealed two kinematic invariants. Human reaching movements tend to be straight with respect to the hand and have a characteristic bell shaped velocity profile. These features are only found when measured in a fixed Cartesian reference frame. Similar invariants could not be detected for the angular components of these reaching movements. These smooth hand trajectories were observed with such consistency that investigators proposed that the central nervous system might plan movements with respect to the motion of the hand and worked to quantify this behavior.

Hogan [14] found that in the single degree of freedom case these kinematic invariants could be explained assuming the central nervous system organizes movements such as to optimize the smoothness of the hand's trajectory (assuming normal unperturbed reaching movements). To derive an analytical expression for describing these movements an appropriate cost function was formed. The smoothness was defined as the mean squared jerk integrated over a reaching movement. Utilizing optimization theory to minimize this functional results in a quintic polynomial for the hand's trajectory. Imposing the appropriate boundary conditions this minimum jerk profile (the quintic polynomial) was found to agree fairly well with experimental data. This idea was then extended to the multi degree of freedom case by Flash and Hogan [8]. It was found that similar conclusions could be made by minimizing the mean squared jerk in Cartesian hand coordinates [1].

Similar attempts were made to characterize human reaching movements in terms

[1]an analogous polynomial can be obtained by minimizing jerk in joint coordinates but the resulting hand trajectories vary considerably from actual hand paths further supporting the claim that the central nervous system plans movements in hand coordinates

of "effort". Uno et al. [33] made use of similar optimization principles by minimizing the integral of the time derivative of torque. The resulting expression describing the torque necessary for a smooth movement captured the experimental data as well. A similar level of success was to be expected for the single degree of freedom movements though, for in the single joint case the time derivative of torque and jerk (the third derivative of position) differ by a scaling factor, the inertia of the limb. The ideas were extended to the multi-degree of freedom case but with less success [2].

2.1.1 Kinematic vs dynamic based models

The work of Hogan and Flash carries with it some distinct and fortunately quantifiable implications. If human movements can be typified merely by minimizing a derivative (the third time derivative) of a coordinate based variable then what can be said of the organization of movements? Clearly if this is *all* the information the central nervous system incorporates, then movements must be organized kinematically. Some definite predictions for a kinematic based model can be inferred for motor control. Perceived errors in the hand trajectory should elicit corrective motor commands. This means that for varying conditions varying motor commands would be required for the same trajectory.

A further implication of the kinematic model lies in the progression of neural events that conclude in motor commands. Once a kinematic trajectory has been fashioned (based entirely on kinematics) the appropriate motor commands must in turn be generated. The presumption here is a hierarchy underlying the organization of movement -kinematics and then dynamics.

The work done by Uno, Kawato and others pursuing this same line of ideas carries its own implications as well. If human movements can be typified by minimizing a derivative of a force variable (first time derivative) the underlying organization must be dynamic in nature. In contrast to the kinematically organized model, a dynamically organized model that perceives errors in the hand trajectory, provided

[2] A closed form solution is not available in the multi degree of freedom case and iterative methods must be employed

they do not alter the desired terminating state, will elicit no motor response.

Continuing the comparison between the two models, consider the progression of neural events that must take place when movement is organized with a dynamic based model. Whereas the kinematic based model must tackle the distinct steps of trajectory formation and corresponding motor command generation, the dynamic organization of movement results in explicit motor commands. There is no need for separating the organization of motor planning and the resulting motor commands into two distinct processes.

Proponents of both camps have sought to provide evidence for these two opposing concepts regarding motor control. As mentioned previously, kinematically organized movement does not take into consideration the forces required when creating hand trajectories. If human movements are organized as such, then perturbed hand paths would elicit motor commands such as to return the hand to its desired trajectory. An experiment was designed to test this exact deduction (Wolpert et al. [35]). Participants were asked to make point to point movements while the position of their hand was displayed to them via a monitor. While the trials took place their hand path was visually perturbed so as to appear more curved than was the case. The participants almost invariably compensated for the perceived perturbation.

Many similar experiments have been performed with curl fields ([5, 11]). Participants (both primates and humans) were instructed to make normal reaching movements to displayed targets. A velocity dependent force tending to curve their hand paths was applied. Just as in Wolpert's experiments, subjects always displayed a marked level of adaptation so as to straighten their hand paths. What these studies have shown, is the propensity humans have to produce straight line hand paths, even when perturbed such that forces in excess of those normally required are necessary.

It should be noted that the kinematic based model makes no claims about the process of generating motor commands. It solely claims to describe the underpinnings of the organization of movement. The dynamic model on the other hand offers a solution by predicting the necessary forces for a movement.

2.2 Equilibrium Point Control Hypothesis

An early theory concerning the generation of motor commands proposed by Feldman [7] stated that neural activity, descending and ascending pathways and peripheral spinal loops, whose aggregate response was tantamount to specifying an equilibrium configuration, initiated movement. This idea, termed Equilibrium Control, is an attractive proposition in the light of a kinematic based model. So whereas the issue of motor commands is somewhat circumvented with the kinematic based model for the organization of movements, when combined with equilibrium control a plausible solution surfaces.

If motor commands are in effect attractors within muscle space (the space representing muscle lengths), allowing the limb to tend to an equilibrium configuration, then movement might be initiated by varying this equilibrium configuration to correspond with a desired hand trajectory. The inherent stiffness properties of the neuromuscular system might then provide an adequate form of closed loop feedback about this trajectory. This means that the central nervous system might (at least for normal reaching movements) require little or no knowledge of the dynamics involved therein.

To help place this into perspective a review of some of the basics of the neuromuscular system is in order. As previously stated the properties of muscle are highly non-linear. The force they generate is a function of a pool of neural activity, their resistance to deformation (stiffness) and deformation rate (viscosity). Yet there is a degree of design in place that makes for some sound simplifications.

For a fixed level of activation muscles exhibit characteristic tension vs. displacement and tension vs. displacement rate curves. Although these dependencies are nonlinear they nonetheless attribute a visco-elastic property to muscle. Peripheral reflex loops also elicit a spring like quality. The stretch or myotatic reflex is elicited through a relatively simple loop of neural activity. When muscle stretches intrafusal muscle spindle fibers are excited. These neurons form a feedback loop with efferent alpha motor neurons whose activity is responsible for stimulating muscle. When alpha motor neurons fire they in turn excite gamma motor neurons. These gamma motor neurons innervate the muscle spindle fibers and through their activity effectively change the null point of the spindle fibers response. In a normally functioning person this all contributes to a relatively fast reflex loop allowing for corrective forces when the muscle is perturbed.

A complete description of the neurophysiology of the neuromuscular system is beyond the scope of this thesis. Suffice it to say that the aggregate behavior of the muscle and this peripheral reflex loop is like that of a controllable, damped spring. A spring's resistance to stretch is a conservative force. The energy stored in a deformed spring (linear or non-linear) or any collection thereof can be represented by a potential function. Furthermore, potential functions by definition have zero curl. Therefore you would expect that if the human body, or in particular the human arm, could be represented by a collection of springs then a potential could be found for it with zero curl.

Several such experiments have found this to be fairly true under particular conditions. Shadmehr et al.[31] and Mussa-Ivaldi et al.[26], measured the postural hand stiffness of human subjects and found that the stiffness tensor was virtually curl free.

With spring-like muscles, somewhat analogous to proportional-plus-rate control, the human system is equipped with instantaneous corrective forces when perturbed from its intended trajectory. More support for this notion can be found in the performance characteristics of human movement.

Descending cortical motor signals have a transmission delay of approximately 100 ms. Peripheral spinal reflex loops have a transmission delay of approximately 30ms (for the upper limb). By some estimates these transmission delays would put a limit on the effectiveness of any control based on feedback at around .5Hz (one twentieth of the inverse of the transmission delay). However many human movements exhibit frequency components of up to 15Hz [18]. Muscle stiffness then offers a plausible explanation for the bandwidth of human movements;[3] the intrinsic elasticity, damping and peripheral reflexes act as an internal feedback loop.

[3]the redundancy of the human musculature in the form of agonist and antagonist muscle pairs, also play a critical role in this elasticity

Returning to the equilibrium control hypothesis we see that combined with muscle elasticity and a kinematically organized movement strategy motion can be produced by an appropriate set of neural commands such as to shift the equilibrium point of the neuromuscular system. Motor commands constructed in such a way entail little more than a mapping from a desired equilibrium configuration to an appropriate set of neural commands.

Experiments conducted on spinalized frogs [3] provided some interesting and apparent support for an equilibrium point control scheme. By stimulating distinct neurons in the spinal cord, groups of muscles could be activated whose coordinated contractions generated a force field about an equilibrium point in space. What is more, it was found that when stimulating two sites simultaneously the resulting force field was the vector sum of the individual respective fields. The natural conclusion seems to be that the central nervous system can define any equilibrium posture, provided it can be composed by a superposition of these discrete equilibrium points, by conjointly varying the activation of the respective spinal neurons.

Further evidence was found in the work done by Flash in human reaching movements [9]. She demonstrated how by merely modeling the muscles of the arm as linear springs and dampers in parallel and the descending neural signals as equilibrium joint motor commands, she was able to produce hand trajectories that agreed remarkably well with experimental data.

The advantages of an equilibrium point (EP) control hypothesis, combined with a kinematic strategy for motion and elastic muscles, are twofold. Such a strategy provides a means to generate motor commands, and implies a relative ease of neural computation required. It should be noted here that any conjecture as to what the central nervous system and the brain find difficult to compute is simply that, conjecture, but provided the relative difficulty of neural computation is analogous to the relative difficulty of conventional computation there are some advantages. If descending motor commands correspond to a time varying equilibrium arm configuration or a virtual trajectory [14], then this at least in part alleviates the need for precise estimates of inertial and viscous parameters should an inversion of the dynamics be needed.

2.3 Inverse Dynamics Model Hypothesis

Another prevalent theory concerning the initiation of movement asserts that while the central nervous system must proceed from a kinematically defined trajectory to limb motion, it is done by inverting the dynamic equations of the limb to compute the necessary muscle forces. In contrast to the equilibrium control hypothesis, this would require precise estimates of the inertial, viscous and stiffness parameters of the limb.

Investigating the use of internal models Gomi et al.[12] criticized Flash's work asserting that the muscle stiffness values she used were too high. Flash's simulations of reaching movements were based on a two-link manipulator model of the arm. The stiffness and damping values representing the muscles behaved as a proportional-plus-rate controller. No further assumptions are necessary to conclude that provided the arm isn't excited at resonance, higher gains (stiffer muscles) will translate to better trajectory following. Gomi and Kawato measured arm stiffness on separate occasions and found the numbers to be much lower than those used by Flash.

2.4 Conclusion

There is a great deal of support for kinematically organized movements that correspond to straight line hand paths with bell shaped velocity profiles. These movements might be controlled via an equilibrium point. Following a virtual trajectory in muscle space based entirely on the desired trajectory, elastic muscles might account for movement. This hypothesis might decrease the amount of computation necessary for the central nervous system, but it is lacking in some regards. The values needed for muscle stiffness to account for the trajectory following observed are in question. The support for the inverse dynamics model, wherein motion is created by inverting the dynamics of the arm to generate the forces necessary to follow the desired trajectory, is not as strong. But it too has some attractive qualities. If the central nervous system were to incorporate the dynamics involved in motion for the creation of motor commands, then low muscle stiffness might be reconciled.

Chapter 3

Networks

3.1 Neural Networks

Neural networks are fundamental to the work presented in this thesis. As a result, some effort in explaining the material is in order. First and foremost, what are neural networks? By one author's definition,

> "A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use.[1]"

Neural networks, or more simply networks learn mappings. In their most fundamental interpretation they are a method of function approximation. They can be considerably sophisticated or considerably commonplace. In addition to their respectable utility they carry a commensurate number of allusions to intelligence, both of the machine and biological sort. For the purposes of this thesis though, the analogy of networks with supposed mechanisms of biological intelligence is only an interesting and perhaps insightful footnote. This is not meant to dismiss any value that neural networks might have in deciphering biological intelligence. On the contrary, there is much to be gained through the analogy. This is only meant as a qualifying statement concerning the goals of this research. For although neural networks fit nicely into

[1]Taken from *Neural Networks* by Simon Haykin, page 2.

the current framework of a biologically inspired control architecture, they are of most benefit as a robust and heuristic approach to function approximation.

Regression is an attempt to estimate a function from some example input-output pairs. The distinguishing feature of non-parametric regression is that there is little or no *a priori* knowledge of the form of the true function. For example, if the relation

$$y = f(x) \tag{3.1}$$

is to be estimated, nothing needs to be known of the nature of f. The estimate is formed through use of a model ϕ such that

$$\hat{y} = \phi(x) \tag{3.2}$$

The form of the model ϕ arises while attempting to match the example input-output pairs. Networks are non-parametric models. Based loosely on the structure of the central nervous system, they normally consist of a group of simple processing elements referred to as *nodes*, which are analogous to neurons. These nodes have a large interconnected topology such that there is a great deal of interaction between them. The resulting activity of any individual node is a function of the weighted inputs of the nodes connected to it.

Traditionally networks have been organized into layers wherein each layer receives input from the one before it and projects its outputs to the one ahead of it. These multi-layer feed-forward networks have the capability of learning any function within arbitrary accuracy, provided they have at least three layers. More layers may be useful for functions that contain discontinuities or similar pathologies.

One particularly alluring class of networks is the radial basis function network, or RBF network. An RBF network is a single-layer linear model (figure3-1), hence the mathematics are relatively simple. As a result, linear optimization tools may be employed when desired. It has also been demonstrated that while multilayer networks have the ability to approximate arbitrarily well any continuous function, RBF networks have similar properties and map directly onto this class of three-layerered network. Unlike the nonlinear multi-layer networks, the RBF network does

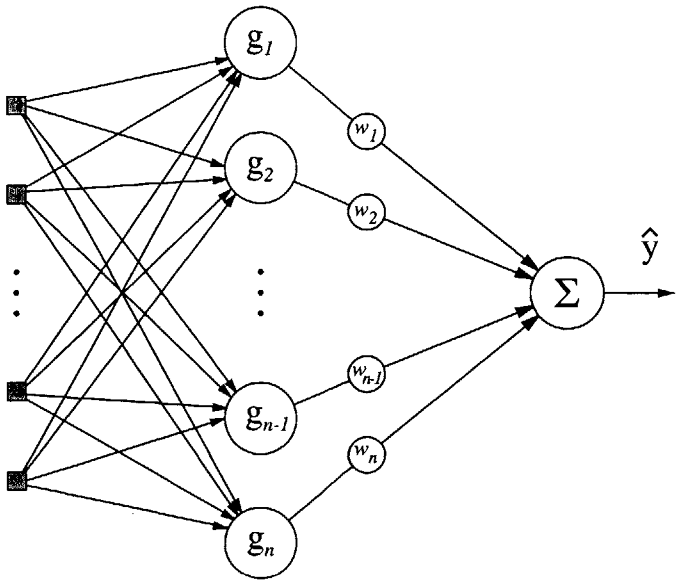

Figure 3-1: Diagram of Radial Basis Function Network

not face the same difficulties with local minima that hamper traditional multi-layer network training.

An RBF network is so named for its basis functions. Radial basis functions monotonically decrease (or increase) away from a center. The basis function most often used, and the one used in this thesis, is the Gaussian. The GRBF (Gaussian-RBF) network is a linear sum of these weighted Gaussian basis functions. By adjusting the weights associated with individual basis functions the model "learns" the unknown function. This is the process of training. Unlike conventional feed-forward networks where the output is a nonlinear function of the varying weights the GRBF network is a linear summation of nonlinear basis functions. The resulting model ϕ has the form:

$$\phi(x, \boldsymbol{w}) = \sum_{i=1}^{n} w_i g_i(x) \qquad \text{where} \qquad g_i(x) = \exp\left(\frac{-(x-\xi_i)^2}{\sigma_i^2}\right) \tag{3.3}$$

where ξ_i and σ_i^2 are the center and variance of the i^{th} basis function.

The GRBF networks are analogous to a nonlinear form of signal reconstruction. Instead of using a sample-and-hold scheme, which can be thought of as a series of step functions scaled by a sampled value, we use Gaussians and scale them to most closely match the sampled signal. Moreover, because the Gaussians are a function of the input variables, an estimate of the true function can be built. Using similar ideas, Sanner and Slotine [29, 30] derived some elegant methods for defining error bounds when using these networks.

3.2 Training Algorithms

Training a network can be accomplished by minimizing the error between its estimated value and the desired, or true, values. Given a training set of input-output pairs, $\mathcal{T} = \{x_i, y_i\}_{i=1}^{p}$ a cost function is defined. For instance, the sum of squared differences between the true outputs and the network's response given the corresponding input:

$$E = \frac{1}{2}\sum_{i=1}^{p}(y_i - \hat{y}_i)^2 \tag{3.4}$$

The goal is to minimize this cost function over the training set. To do this we need to express how each weight in the network contributes to the error functional so that the weights can be adjusted accordingly. This is done by back-propagating through the function E to compute the gradient. Taking the partial derivatives of E with respect to the weights we have:

$$\nabla_{\boldsymbol{w}} E = -\frac{\partial \hat{y}^T}{\partial \boldsymbol{w}}(y_i - \hat{y}_i) \tag{3.5}$$

Thus the gradient is computed by multiplying the error by the transpose of the Jacobian matrix, $\partial \hat{y}^T / \partial \boldsymbol{w}$. This expression describes a vector (in n dimensional space where n is the number of weights) pointing in the direction of greatest increase in the cost function, or the error. To reduce this error the weights are moved in the opposite direction. This is the basic description of a gradient descent training algorithm; evaluate the Jacobian at the current x and increment the weights in proportion to the current error. Fortunately, computing the Jacobian of the GRBF network is relatively easy because of the linear nature of the network (Appendix A). Other approaches of varying levels of sophistication can be taken as well.

The gradient descent algorithm is well suited to an "on-line" learning scheme, where on-line describes an iterative routine of incremental adjustments to the network at each sampling cycle. With this approach, there is no training set to start with and training pairs are measured at each sample. The training set will then consist either of a single training pair or a running tally of training pairs constituting a growing training set. The weights are adapted each training cycle by computing the gradient and adjusting with some step size α as follows:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \alpha \nabla_{\boldsymbol{w}} E \tag{3.6}$$

One of the major hindrances to learning is the problem of local minima. A conceptual construct to help visualize the way in which the error changes as the weights vary is an error landscape, similar to a potential energy landscape. While traveling through this error landscape of the cost function a gradient descent algorithm may become "stuck" within a relatively flat area of the terrain where the gradient is small or zero. This is an especially difficult problem with nonlinear networks. Fortunately, this is not the case with the linear GRBF network. The weights modulate the network's response in a linear fashion and therefore do not introduce multiple minima.

A further complication to training is the concept of over-training. Once the network has been trained on the data set $\mathcal{T}$, it should be validated. The network's response at previously untrained data points is compared with the corresponding true values. An overtrained network will give estimates within defined tolerance at the trained points but unsatisfactory estimates at these validation points. Without knowledge of the precision capabilities of the network to guide training, judging when a network has been sufficiently trained to yield a globally optimal solution is difficult. Generally this problem is solved through trial and error.

Because GRBF networks are linear they can also make use of linear optimization tools. A least squares solution can be obtained for the optimal weight vector of a given training set. This approach can be used on-line as well, but in order to be effective a large training set is necessary.

3.3 Training Paradigms

Algorithms describe systematic steps to be taken towards solving a problem. Paradigms describe the manner in which you pose the problem to obtain a desired solution. The

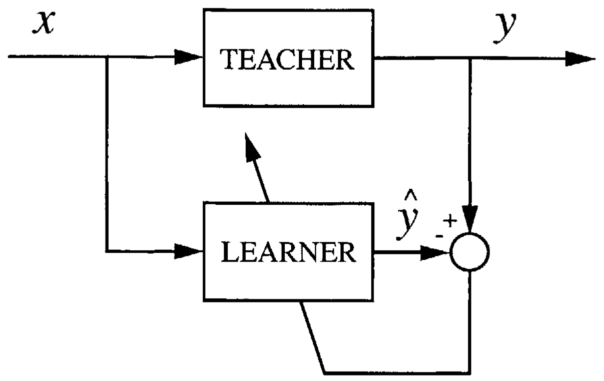

Figure 3-2: Schematic representation of supervised learning.

algorithm presented above, the gradient descent, is a tool for training networks. The paradigms explain how to use these tools. The choice of a paradigm may appear to be a trivial step, but it can make a significant difference in training a network.

Supervised learning is characterized by training with a teacher. The teacher provides correct answers to the learner, or regression model, in an effort to help it learn by example. The input x is made available to both the learner and the teacher. The teacher then provides the correct response, completing the training pair. The learner, while adapting according to the methods outlined above, is modeling the teacher (figure 3-2). Supervised learning is especially useful when the learner is a forward model. Forward models learn the causal relationship between inputs and outputs or actions and outcomes.

Training an inverse model is more difficult. The inverse model, equation3.7

$$\hat{x} = \psi(y, \boldsymbol{v}) \tag{3.7}$$

learns to predict the action for a given outcome. In this case it learns the correct input x, for a given output y. Often though, the correct action for a given outcome is unknown (e.g. the forward relationship might be many-to-one and not invertible). For example, with the arm of the experiment, the forward map from joint to endpoint coordinates is many-to-one, the corresponding inverse map, from endpoint to joint coordinates is one-to-many. In instances such as this, the inverse model works most

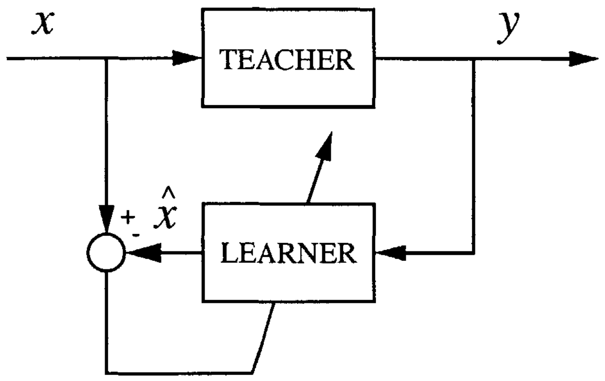

Figure 3-3: Schematic representation of direct inverse learning.

efficiently when estimating a subset of the inverse mapping.

There are several approaches to training the inverse models: direct inverse learning, feedback error learning, and distal supervised learning. Figure 3-3 shows a schematic for training an inverse model using direct inverse learning (classical supervised learning). The teacher's outcome and action, which are the learner's input and output respectively, constitute the training pair. The learner then adapts accordingly, modeling the inverse mapping of the teacher. Yet problems arise when the mapping is many-to-one, and more particularly when the mapping is non-convex. A set is said to be non-convex if for every pair of points in a set, all points on a line between them do not lie in that set. Returning to the example of the arm, there are two sets of joint coordinates for each endpoint, corresponding to the elbow up and elbow down positions. Yet any set of joint coordinates between the elbow up, and elbow down coordinates, results in an incorrect configuration. Using a gradient descent scheme in conjunction with the direct supervised learner resolves this one-to-many problem by averaging the error across these non-convex points. Therefore an inverse kinematic map, learning to predict joint coordinates from desired endpoint coordinates, would be driven to predict a set of joint coordinates between the correct, though redundant, elbow up and elbow down configurations. Such a result would be an incorrect response.

This problem can be overcome if the inverse model is presented with desired

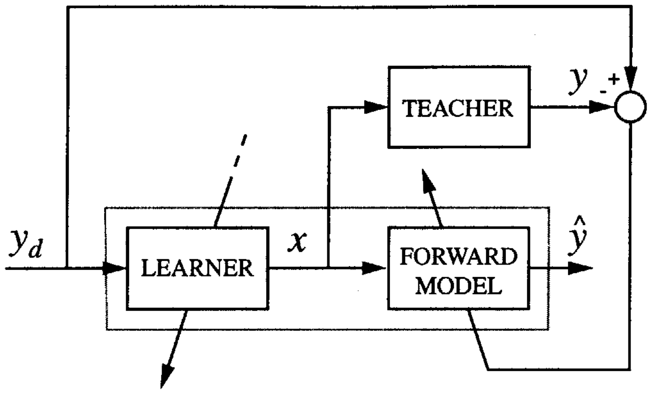

Figure 3-4: Schematic representation of distal supervised learning.

actions (joint coordinates) that maintain an appropriate consistency (elbow up or down) such that the inverse model can learn an invertible subset of the mapping. Although this assumes knowledge of the desired actions, and they are not always known. In some cases only the outcome is known, in these situations the direct inverse learning approach is not of use.

There is another similar approach, with a subtle difference. With feedback error learning, access to the desired action, x^d, is not required. Instead, a signed error signal $\tilde{x}$, proportional to the difference between the learner's output and the desired action, is utilized. Conceptually, the schematic in figure 3-3 still applies, although now equation 3.5 (altered to fit the inverse model) is changed to include the error signal when evolving the weights,

$$\nabla_{\boldsymbol{v}} E = -\frac{\partial \hat{x}^T}{\partial \boldsymbol{v}} (\tilde{x}) \tag{3.8}$$

This method is susceptible to the same problems as those of direct inverse learning.

The third approach, the distal supervised learner, as the name suggests, does not train on proximal variables (the variables the inverse learner controls directly, the actions). Instead, it trains on the distal variables, the outcome. Now the learner is used to produce actions for the teacher. One more important change is introduced. The learner is placed in series with a second learner, a forward model (figure 3-4). The forward model is trained using a supervised learning scheme as outlined above. The learner's actions, x, are made available to both the teacher and the forward model and training on the error between the actual outcome y and the estimated outcome $\hat{y}$, we arrive at a forward model using equation 3.5.

To train the learner, it and the forward model are then treated as a single composite learner which can be trained with the supervised learning algorithm to minimize the error between the actual outcome y and the desired outcome, y^d. Using a similar approach to train the composite model as with a forward model, one would back-propagate through the cost function to compute the gradient with respect to the inverse model's weights. Symbolically this is represented by:

$$\nabla_{\mathrm{v}} E = -\frac{\partial y^T}{\partial x}\frac{\partial x^T}{\partial \mathrm{v}}\left(y^d - y\right) \tag{3.9}$$

Herein lies the problem that the direct inverse and feedback error learning methods are used to avoid. The models are assumed to be interacting with an unknown or indescribable environment hence $\partial y/\partial x$ is unknown. The composite system offers a solution to this (problem). Using the forward model's estimate of the teacher we have $\partial \hat{y}/\partial x$ as an estimate of the unknown Jacobian.

With this estimate of the gradient of the cost function, appropriate learning schemes may then be applied. There are two benefits to this approach. The correct actions need not be known and, unlike direct inverse learning, distal supervised learning can handle the many-to-one problem for the learner trains on the distal error. In the case of the arm, the distal error is the difference between the actual endpoint coordinates and the desired endpoint coordinates. Thus if the inverse model converges to a correct solution it has implicitly learned an appropriate invertible subset of the many-to-one mapping.

A circumstance that is especially helpful in grasping the paradigm is when the inputs to the inverse model are the desired outputs of the teacher. In training the composite model, an identity mapping from inputs to outputs is created. An inaccuracy in the forward model is a bias which the learner (inverse model) can then compensate for.

3.4 Dimensional difficulties

There is a price to pay for the GRBF network's advantages in speed of convergence and ease of computation. Inherent in the basis function is a dependency on dimension. Consider a model that estimates a mapping from one dimension to one dimension, $R \Rightarrow R$. To prime the space for use with the network, an arbitrary subset of the domain would then be meshed, placing centers at the mesh points for each Gaussian (basis function). If we choose to mesh the domain in one tenth increments ($\delta = 1/10$) there will be 10 centers in the domain with 10 respective Gaussians. Now consider a model for estimating a mapping from one dimension to two, $R \Rightarrow R^2$. With the same mesh size δ, one tenth in the first dimension of the domain, and one tenth in the second, 100 centers are needed, with 100 Gaussians. The number of nodes required for a network model increases with the product of the dimension, so for a mapping from $R \Rightarrow R^n$ the number of nodes will be on the order of

$$\prod_{i=1}^{n} 1/\delta_i \tag{3.10}$$

where δ_i is the mesh size for the i^{th} dimension. Furthermore, when attempting distal supervised learning and back-propagating through the cost function, the contribution of each node of the inverse model with respect to each node of the forward model (the Jacobian computation) must be computed for training. As a result, the number of calculations rises with the product of the nodes of both networks.

Some of these dimensional problems may be alleviated by using nonlinear methods. For instance, using fewer basis functions but allowing their centers or variances to wander. However, once down this route all advantages of a linear network model must be abandoned, thereby increasing the difficulties inherent in training.

As mentioned previously, Sanner and Slotine have shown how one can put bounds on the error of an GRBF network in theory. In practice, this proves problematic. There are several terms necessary to compute the error, one of which requires integrating the Fourier transform of the actual function to be estimated. The Fourier transform however is defined for aperiodic functions that decay to zero, characteristics that the functions to be approximated in this thesis do not have.

Chapter 4

Physical Apparatus

In order to explore the abilities of the present control scheme, a test bed upon which the experiment may be performed is needed. The test bed, an arm, will allow the controller to produce the reaching movements necessary for the experiments. Instead of using physical "hardware", the test bed was simulated. This chapter will describe the model of the arm that was used in the experiments performed in this thesis.

As previously mentioned, the human arm is a complex mechanism by any engineering measure. Yet for stereotypical reaching motions confined to a plane, realistic representations of the mechanics of the arm can be obtained by modeling it as a two-link open kinematic chain. This is the model that has been employed by a number of researchers in the past with a good deal of success [9, 12, 13]. Not only does this simplify the analysis of human reaching movements, it also helps to parallel the human arm and a robotic manipulator. Thus conjecture for the use of biomorphic motor control in robotics can be drawn more clearly.

4.1 Physical Model

For the two-link manipulator representation of the arm some assumptions must be stated concerning the reaching movements considered. It is assumed that these movements can be completely captured by movement in a plane. It is further assumed that these movements can be depicted without the need of wrist movements. More

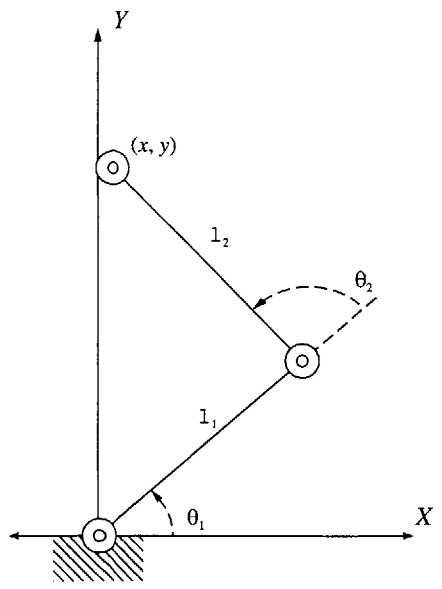

Figure 4-1: Diagram of arm-like manipulator.

assumptions concerning stiffness and viscosity of the muscles will be made below. With these caveats in place, the arm can be mathematically modeled as two rigid links, each with one rotational degree of freedom (figure 4-1).

4.2 Mechanics

4.2.1 Kinematics

The geometry of the arm is fairly straightforward. The configuration is characterized by the relative joint angles θ_1, θ_2 which in turn determine the Cartesian endpoint coordinates x, y. The forward (from joint angles to endpoint coordinates) kinematic relationship describing the geometry of the arm can be expressed symbolically as, $\boldsymbol{x} = L(\boldsymbol{\theta})$, or,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \end{bmatrix} \tag{4.1}$$

Taking the time derivative of equation 4.1, the forward relationship describing endpoint velocity is,

$$\dot{\boldsymbol{x}} = (\partial L/\partial \boldsymbol{\theta})\dot{\boldsymbol{\theta}} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{4.2}$$

where $\mathbf{J}(\boldsymbol{\theta})$ is the Jacobian matrix. Similarly, the forward relationship for endpoint acceleration is found by taking the time derivative of equation 4.2 to yield,

$$\ddot{\boldsymbol{x}} = \dot{\mathbf{J}}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} + \mathbf{J}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} \tag{4.3}$$

The inverse relationships can be found as well, that is, given the endpoint coordinates, velocity and acceleration, a solution for the corresponding joint coordinates, joint velocities and joint accelerations may be obtained. Although there is no solution for the inverse kinematic relations in general, imposing that the elbow joint (θ_2) is confined to positive angles solutions can be found for the following relations $\boldsymbol{\theta} = L(\boldsymbol{x})^{-1}$, $\dot{\boldsymbol{\theta}} = \mathbf{J}(\boldsymbol{\theta})^{-1}\dot{\boldsymbol{x}}$ and $\ddot{\boldsymbol{\theta}} = \mathbf{J}(\boldsymbol{\theta})^{-1}[\ddot{\boldsymbol{x}} - \dot{\mathbf{J}}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}]$.

4.2.2 Dynamics

With the above model and the assumption that the arm lies in a plane perpendicular to the acceleration of gravity, the equations of motion can be derived. Using an Eulerian or Lagrangian approach results in the following (shown symbolically),

$$\boldsymbol{H}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} = \boldsymbol{\tau} \tag{4.4}$$

where $\boldsymbol{H}$ is a matrix of configuration dependent inertia terms and $\boldsymbol{C}$ denotes the velocity induced acceleration terms (Coriolis and centripetal). $\boldsymbol{\tau}$ is a two element array of generalized joint torques (for a more complete description see Appendix A).

The physical parameters, arm length, mass, inertia etc., are chosen from tabulated anthropomorphic data [34] so as to be comparable with those of an average person.

4.3 Motor Commands

There is still the matter of actuation. In the arm this is achieved through the use of muscles. Although much is known about the nature of the electro-chemical processes that underlie the tension properties of muscle, there is no need to model it. In fact, there is no need to explicitly model the muscles at all.

Hogan in particular [13, 14, 16] has shown how the aggregate behavior of the muscles employed in posture defines a resistance to deflection, a resistance to the rate of deflection, and an equilibrium configuration. These are a function of neural inputs and reflex loops and can vary widely. However, the dominant mechanical behavior of muscle can be captured by considering it's dependence on deflection and rate of deflection only. The force produced, perhaps a coupled nonlinear function, is

$$f = f(l, \dot{l}) \tag{4.5}$$

where f is the tension, l is the length, $\dot{l}$ is the time rate of change of length. Expanding this function in a Taylor series[1] about an operating point $(l_o, \dot{l}_o)$ we get

$$f(l, \dot{l}) = f(l_o, \dot{l}_o) + \frac{\partial f(l_o, \dot{l}_o)}{\partial l}(l - l_o) + \frac{\partial f(l_o, \dot{l}_o)}{\partial \dot{l}}(\dot{l} - \dot{l}_o) \ldots h.o.t. \tag{4.6}$$

If one were to truncate the expansion with the first order terms then the stiffness and viscosity can be defined respectively as,

$$k = \frac{\partial f(l_o, \dot{l}_o)}{\partial l} \tag{4.7}$$

$$b = \frac{\partial f(l_o, \dot{l}_o)}{\partial \dot{l}} \tag{4.8}$$

Without loss of generality we can extend this line of reasoning to any number of muscles involved in the actuation of a limb to define a tension vector. Making the further simplifying assumption that the musculo-skeletal interactions have constant moment arms, the nonlinear mapping from muscle length to joint angles vanishes and a one-to-one correspondence between the two remains. The resulting torques about these joints can then be represented as

$$\boldsymbol{\tau} = \boldsymbol{K}(\boldsymbol{\theta}_o - \boldsymbol{\theta}) + \boldsymbol{B}(\dot{\boldsymbol{\theta}}_o - \dot{\boldsymbol{\theta}}) \tag{4.9}$$

[1]For the sake of argument we lay aside the question of whether or not this series converges.

where $\boldsymbol{\theta}_o$ and $\dot{\boldsymbol{\theta}}_o$ correspond to an operating point vector. Incorporating the principles of an EP control theory, $\boldsymbol{\theta}_o$ (and maybe $\dot{\boldsymbol{\theta}}_o$) is neurally specified and can vary in time to create a virtual trajectory. Using this virtual, or desired trajectory, equation 4.9 is substituted into equation 4.4 (damping will be modeled as acting with respect to ground, i.e. $\dot{\boldsymbol{\theta}}_o = \mathbf{0}$ and the subscript o shall be replaced with d to distinguish that this is the desired equilibrium trajectory) to arrive at the governing equations of motion for the arm.

$$\boldsymbol{H}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{B}\dot{\boldsymbol{\theta}} + \boldsymbol{K}\boldsymbol{\theta} = \boldsymbol{K}\boldsymbol{\theta}_d \tag{4.10}$$

External torques applied to the arm are summed in the right-half of the equation as follows,

$$\boldsymbol{H}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{B}\dot{\boldsymbol{\theta}} + \boldsymbol{K}\boldsymbol{\theta} = \boldsymbol{K}\boldsymbol{\theta}_d + \boldsymbol{\tau} \tag{4.11}$$

Chapter 5

A Motor Control Paradigm

With that informal introduction to some of the prominent concepts and theories of human motor control as well as the use of neural networks and a model for simulating a human-like arm, we are in a position to describe a control architecture that strives to emulate the functionality of the human motor system.

Recapitulating some important points for this work, there are slow feedback loops that limit the highest frequency for feedback control to be effective. Feedback is likely then not the sole basis for control. Movements are probably organized kinematically, more specifically, about the hand's path. A feed-forward equilibrium point trajectory based on a minimum jerk path, combined with muscle elasticity, might be adequate for slow movements, but is not sufficient for faster movements where the dynamics of motion are more prominent. Internal models though, might be useful in compensating for these shortcomings.

Much of the research presented so far, has in essence, emulated human motor control using the aforementioned theories and principles: kinematically organized movements, muscle stiffness, EP or inverse dynamics control hypothesis. But what of motor learning? Can one devise a control system, that like the human one, has the capacity to learn and retrieve multiple behaviors, and generalize that knowledge to help adapt to new behaviors? With such a model in place one would be in a position to exploit the utility of the human motor system. This would also afford new information concerning its malfunction, and recovery.

Humans have a large repertoire of motor behaviors for varying environments. We can adapt to differing environments and are almost always stable. We can generalize knowledge learned from previously experienced motor behaviors to new behaviors. What could this all be attributed to? Human motor control could be a single amorphous controller that continually adapts according to the situation at hand (within tolerances). Yet if this were so, we would continually experience transients in our behavior while we learned to model the new environment, which does not seem to be the case. On the contrary, once we have been exposed to a new environment, we can adapt to it more quickly the second time around. This suggests a more plausible alternative, that humans learn motor behaviors, set aside that knowledge and retrieve it when necessary. This might be accomplished through the use of internal models.

5.1 Internal models

5.1.1 Forward models

Forward models learn the causal relationship between inputs and outputs (actions and outcomes). This has several practical uses. A forward model can predict the state of a system at some point in the future, or the rate at which a state is changing. For example, a forward model could be used to estimate the equations of motion for a system. The forward estimate would then be the first time derivative of that system's state. Integrating the forward model, we have an estimate of the state that can be used when sensory feedback is too slow to be effective for control purposes. Inherent in the forward model are the dynamics of the context in which it is trained; this makes for another use. Comparing the actual forward state with the estimate, is in essence, measuring to what extent the current context matches the context upon which the forward model was trained. This allows for a convenient means of context estimation. In addition, the forward models can be used to help learn an inverse model, as previously explained with the distal supervised learning scheme.

5.1.2 Inverse models

Models that learn to invert a mapping from inputs to outputs are inverse models. This makes them naturally suited for motor controllers. If an inverse model can learn to invert a mapping from motor commands to a forward state, then presenting it with a desired forward state will produce the correct motor command. In general, the inverse model is more difficult to learn than a forward model. Often the forward relationship is many-to-one and not invertible. This means there are multiple actions for the same outcome. This can lead to many difficulties and steps should be taken to ensure that this does not occur.

5.2 The Control Paradigm

Now we are ready to present the control architecture based on some plausible theories concerning human motor control and some practical ideas concerning motor learning. Before proceeding any further, the work and ideas upon which this thesis are in large part founded, must be noted with due consideration. In a paper by Shadmehr [32] we are introduced to a controller based on the CMAC, the cerebellar model articular controller (originally proposed by Albus [1]). The concept was to place a learner in parallel with the controller. The controller generated feed-forward joint coordinates based on a desired kinematic trajectory. The learner then adapted to compensate for the dynamics of the plant. This work was an attempt at human motor control mimicry as well, for it was argued that the learner was performing a function analogous to that of the cerebellum. By comparing the feed-forward motor commands (presumably from the motor cortex) with the state feedback (afferent sensory inputs) the learner was able to supplement the kinematically generated motor commands to achieve better trajectory following performance.

Jordan [20] has examined the utility of using neural networks in the form of a hierarchy of experts. In this scheme, the knowledge is parsed amongst one or more experts that learn a subset of the desired mapping. A gating system for the separate networks was also described based on a probabilistic interpretation.

Wolpert and Kawato [38] have taken the mixture of experts concept and provided a framework for implementing it in an effort to learn multiple motor behaviors. They suggest a method for combining multiple experts, each responsible for learning a different behavior or portion thereof, and a method for selecting amongst these experts, both while learning and while generating motor commands. They also claim that the controller has many analogs with the human motor control system.

The controller presented in this paper is a combination of these ideas. While easily generalized to any dynamic system, this thesis concerns itself with control of an arm-like manipulator. The arm represents a compromise; there is a desired to emulate the physiological structure in order to draw conclusions about the human motor control system, and a desire for the model to remain practical enough to be implemented in hardware, such that it may be considered for a control alternative.

The control system consists of multiple paired forward and inverse models. All models are constructed from networks and trained while the arm interacts with its environment. A single kinematic module, a forward and inverse kinematic model, is used to generate feed-forward motor commands. Multiple dynamic modules, paired forward and inverse dynamic models, generate feedback motor commands to supplement the kinematically generated commands. These modules represent experts for learned, or as yet unlearned, motor behaviors in different contexts and are employed in parallel to compete for motor commands. The inverse models generate motor commands while their paired forward models (save the forward kinematic model) are used to estimate the current context within which the controller is acting. A gating mechanism is used to blend a module between "on" and "off" according to how well the current context matches the context it has been trained for. In theory, any number of modules may be had, allowing for an ever increasing level of adaptability and versatility in varying contexts.

The forward kinematic model produces an estimate of the "hand" position of the arm, the endpoint. Based on a purely kinematic relation it learns a mapping from joint coordinates to the Cartesian endpoint coordinates. Once the model has been learned, within a suitable level of precision and accuracy, it is no longer trained and remains static. This is based on the speculation that humans, and perhaps other animals, may realize the sense of permanency associated with these kinematic relations, regardless of environment or dynamics. Although this model is not put to use for motor control, it, like the other forward models, can be used to help learn the inverse relationship.

In addition to the forward kinematic model we shall use two (in principle any amount may be used but for present purposes two are sufficient) forward dynamic models. Strictly speaking, for this two-link mechanism the state is defined in terms of the joint coordinates and their first derivatives. Yet for this simulation, the arm is confined to a restricted workspace. Similar to a human, the elbow joint cannot achieve negative angles. In such an instance the endpoint coordinates and their first derivatives uniquely define the state of the arm and will be considered a pseudo-state. Rather than estimate the full pseudo-state time derivative (the endpoint velocity and acceleration in Cartesian coordinates), they simply estimate the acceleration of the endpoint. For the i^{th} forward model, with weight vector $\boldsymbol{v}_i$ and control input $\boldsymbol{u}$ we have,

$$\hat{\ddot{\boldsymbol{x}}}_i = \phi_i(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{v}_i) \qquad \text{where} \qquad \hat{\ddot{\boldsymbol{x}}} = [\hat{\ddot{x}}, \hat{\ddot{y}}]^T \tag{5.1}$$

where ϕ is a GRBF network, as described in Chapter 3. Equation 5.1 is an estimate of the arm's equations of motion, equation 4.10, mapped forward to the endpoint coordinates via equations 4.1, 4.2, and 4.3. In utilizing several forward dynamic models, we will examine the controller's ability to learn just as many forward relationships for the motor system. Each of these relationships characterizes the dynamics of the system while interacting in a particular context.

With several learned forward relationships we are afforded a means of estimating contexts. In effect, each forward model is a description of a previously experienced context. Calling upon this information can guide the controller in assessing the current context within which it is interacting. As previously described (Chapter 3.3), the forward dynamic models can also be used to train the inverse dynamic models.

For each forward model there is a paired inverse model. Inverse models, as stated before, are well suited for motor commands, and this is exactly how they are implemented. Just as Shadmehr's CMAC had a kinematically driven component to the motor command so too does this controller. The kinematic module is in some sense an expert itself and allows for feed-forward motor commands when there are no learned behaviors, or feedback available. The inverse kinematic model estimates the joint coordinates that correspond to a given endpoint position of the arm. For a desired endpoint $\boldsymbol{x}_d$ and weight vector $\boldsymbol{w}_k$ the kinematic estimate is,

$$\boldsymbol{u}_k = \psi_k(\boldsymbol{x}_d, \boldsymbol{w}_k) \tag{5.2}$$

where again, ψ, is a GRBF network.

The inverse dynamic models supplement these joint coordinates to account for the dynamics of the current context. Note that these inverse dynamic models are not actually learning to invert the dynamics of any context. Because of the strictly kinematic component of the motor command the inverse dynamic models actually represent the residual dynamics, so to speak, the dynamics that remain after the kinematic component has been teased out. They are computed as follows,

$$\boldsymbol{u}_i = \psi_i(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{x}_d, \boldsymbol{w}_i) \tag{5.3}$$

Note that the motor command is a function of the desired endpoint as well as the pseudo-state. This is consistent with the assumption that the human motor system organizes movements based on the coordinates of the hand and not the joints. Together, the inverse kinematic and dynamic models, generate an equilibrium trajectory for the arm to follow.

With several modules (again, each module represents a coupled forward and inverse model) acting at once, the problem of selecting amongst them arises. A methodology for gating their outputs according to the current context must be devised. This method has already been alluded to above. Each forward dynamic model is trained within a specific context to learn the respective dynamics therein. Presumably then, the forward model that best estimates the acceleration of the endpoint at any one time, has most closely described the dynamics of the current context. Similarly, if two forward models estimate the acceleration equally well, then the current context can be assumed to be a corresponding mixture of the two. This allows for blending of previously learned behaviors, a form of generalization.

A gating, or scaling, value is computed using the soft max function. This can be thought of as the probability that a forward model presently accounts for the current context. These probabilities, or responsibilities, as we shall refer to them, are positive and sum to 1. For the i^{th} module, the soft max is computed according to the following,

$$soft\ max = \frac{\exp(||\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_i||^2/\sigma_i^2)}{\sum \exp(-||\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_j||^2/\sigma_j^2)} \tag{5.4}$$

The responsibility signal will be updated often during the operation of the controller and the values obtained from the equation above will likely fluctuate to some degree due to inaccuracies in the forward estimates. These high frequency fluctuations are undesirable; a somewhat slower varying signal will be more advantageous for the training of the modules and switching between contexts. To this end, equation 5.4 should be considered a responsibility update signal, to be filtered. The responsibility, λ_i, for the i^{th} module, will then be computed as follows,

$$\dot{\lambda}_i = \frac{1}{\tau_\lambda}\left[-\lambda_i + \frac{\exp(-||\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_i||^2/\sigma_i^2)}{\sum \exp(-||\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_j||^2/\sigma_j^2)}\right] \tag{5.5}$$

This is a first order filter with a time constant τ_λ. The variance, σ_i^2 in equation 5.5, can have the effect of favoring one module over another. When the variance for a particular module is relatively large, it's contribution to the responsibility update is favored (a large variance widens the Gaussian associated with that module's forward estimate). To take advantage of this, the variance associated with a module will be made proportional to the mean squared error (MSE) in the forward estimate for that module. Trained modules, with relatively small MSE's, will effectively have narrower regions of high responsibility, reflecting the fact that relatively large errors in their forward estimate likely suggest the controller is operating in a context that they have not been trained for. Taking into account the effect training will have on forward estimates, the variance for a module will have to be filtered as well. The variance,

σ_i^2, will be computed as follows,

$$\dot{\sigma}_i^2 = \frac{\lambda_i}{\tau_{\sigma^2}}\left[-\sigma_i^2 + \alpha\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_i\|^2\right] \tag{5.6}$$

The appearance of λ_i in the time constant, ensures that modules with low responsibility receive relatively little updating, as the current context does not correspond to that modules trained context. The constant α is an arbitrary proportionality constant.

The responsibility signal is key to a number of features of the controller, most importantly the coupling of the module's inverse and forward models. Each dynamic module is gated with its responsibility. Forward models with relatively large responsibility values provide correspondingly large contributions to the motor system, via their respective module, in the form of a forward estimate for the arm and a motor command. The forward estimate is then,

$$\hat{\ddot{\boldsymbol{x}}} = \sum \lambda_i \hat{\ddot{\boldsymbol{x}}}_i \tag{5.7}$$

The ultimate motor command is then,

$$\boldsymbol{u} = \boldsymbol{u}_k + \sum \lambda_i \boldsymbol{u}_i \tag{5.8}$$

Through the same line of reasoning, modules with large responsibility values incur a relatively large effort in training. When a module's responsibility is relatively large, then it would benefit most from training under the current context. Similarly, when a module's responsibility is low, then it has not captured the current context well and training under these circumstances would not be advantageous. So the responsibility signal, in addition to gating the motor commands, also gates the training of the modules. This can be seen when using a gradient descent algorithm to adapt the networks. For instance, defining a cost function for the forward models as,

$$\mathcal{L} = \frac{1}{2}\|\ddot{\boldsymbol{x}}_i - \hat{\ddot{\boldsymbol{x}}}\|^2 \tag{5.9}$$

has implicitly made use of the responsibility signal, for when computing the gradient of $\mathcal{L}$ with respect to the i^{th} forward dynamic model the responsibility surfaces,

$$\nabla_{\boldsymbol{v}_i}\mathcal{L} = -\lambda_i \frac{\partial \hat{\ddot{\boldsymbol{x}}}_i}{\partial \boldsymbol{v}_i}^T (\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}) \tag{5.10}$$

The networks have finite size and therefore finite precision in their estimates; this is a known and accepted consequence. This being the case, a boundary layer for their training might be necessary. The boundary layer is an arbitrarily defined area, within which no training of the networks takes place. For the forward dynamic models the boundary layer would represent a tolerance, or minimum level of error in the forward estimate, necessary for training to take place, say ϵ_{fd}. Similarly for the inverse models, a boundary layer representing the error in trajectory following, ϵ_{id}, would be used. Although the responsibility signal does modulate how much a particular network's weights adapt, it does not dictate whether or not that particular network's weights do adapt. The forward models for instance, will all train, regardless of their responsibility, as long as the forward dynamic estimate, equation 5.7 is within tolerance, or outside the boundary layer. To adjust the boundary layer, in parallel with the adaptation rate, it shall be scaled with the responsibility signal. Rather than simply use ϵ_{fd}, we use ϵ_{fd}/λ_i for the i_{th} forward model's boundary layer thickness. Similarly for the i_{th} inverse model the boundary layer thickness would be ϵ_{id}/λ_i.

To gain a better understanding of how the control architecture utilizes the modules and the responsibility signal, let us step through the control process. The diagram of figure 5-1 shows a schematic representation of the control architecture. We shall assume that the kinematic modules have already been trained. During reaching movements a time varying desired endpoint coordinate corresponding with a minimum jerk profile is obtained (this will be provided for the controller). This desired endpoint, $\boldsymbol{x}_d$, is presented to the inverse kinematic model for an estimate of the corresponding joint coordinates. At the same time, $\boldsymbol{x}_d$, as well as the current pseudo-state, $[\boldsymbol{x}, \dot{\boldsymbol{x}}]^T$, is made available to the dynamic modules. Each dynamic module's motor command is gated with it's responsibility signal, summed along with the inverse kinematic model's output (equation 5.8) and used to drive the "muscles" of the arm. Again, the motor commands here will correspond to equilibrium joint coordinates as in equation 4.10. Intermittently the pseudo-state and motor command, $\boldsymbol{u}$, is made available to the forward models. The subsequent response of each forward model, $\hat{\ddot{\boldsymbol{x}}}_i$, is obtained and compared with the actual endpoint acceleration $\ddot{\boldsymbol{x}}$ to update the responsibility

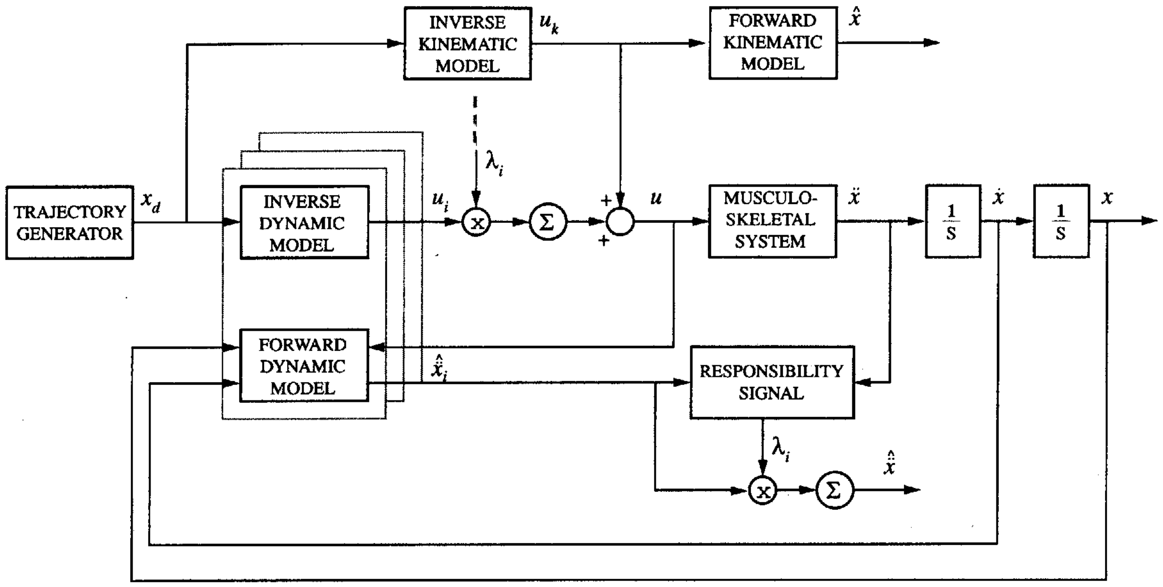

Figure 5-1: Diagram of control architecture.

signal λ_i according to equation 5.5. The variances, σ_i^2, are also updated at this time according to equation 5.6. The dynamic models are then trained, if need be, the forward models on the error in the forward estimate (equation 5.7), and the inverse models on the deviation from the desired endpoint's trajectory $\boldsymbol{x}_d$.

The present control architecture differs from the control systems presented thus far in a number of aspects. In the human reaching simulations mentioned previously, the controllers operated on the intrinsic coordinates, i.e. $\boldsymbol{\theta}$, and not extrinsic coordinates of the hand, $\boldsymbol{x}$. Previous investigations, and this one, have organized the intended movements based on the hand's path, straight hand paths with bell shaped velocity profiles. This intended motion was transformed to desired joint coordinates. The controllers then incorporated feedback and desired information, in these joint coordinates, to compute equilibrium joint coordinates. In so doing, however, the previous models have assumed that a transformation exists, from visual or some other sensory feedback of the hand's coordinates, to joint coordinates. The control system here assumes that no such transformations are necessary, and the desired motion of the hand and feedback of it's motion, are utilized and operated on directly in extrinsic coordinates to compute motor commands.

The use of multiple modules, while not entirely new to a control system, has not yet been used to learn multiple mappings. Jordan for instance, although also using multiple models to control a robotic manipulator, learned only one behavior [20]. The system here attempts to use multiple paired models in a novel attempt to learn and retain multiple behaviors. The ability to learn multiple behaviors would be an advantage over traditional adaptive control systems, e.g. sliding mode adaptive control, or in the linear case with estimators. Consider an adaptive controller such as these that modulates control parameters based on error. The controller will achieve a desired level of performance, tracking error, stability, etc., after sufficient time has elapsed for the parameters to converge to appropriate values (perhaps within a boundary layer). The controller will now have learned a motor behavior for a specific context. What happens when the context changes? Now the parameters will once more have to adapt and settle to new values; a new motor behavior will be gained at the expense of the old one. With multiple modules, experiences will be parsed within the controller in an effort to learn many behaviors. In addition, gating the modules with the responsibility signal, rather than turning them "on" or "off", will allow for adaptive behaviors not explicitly trained for when interacting in contexts that are, in a sense, somewhere between those previously experienced.

As noted several times this system is based on, and an attempt to emulate, the functionality of the human motor system. There is compelling evidence for forward motor commands [28]. There is also ample evidence of kinematic cortical mappings. The degree to which these are associated with motor commands is yet unknown, but we have attempted to incorporate them here. Visual feedback too is the basis for the feedback in extrinsic coordinates. Though not modeled here, area five is purported to be a candidate for trajectory generation as well [6, 21]. With that said, some qualifying remarks should be made. This control system is a functional model, not an anatomical model; it attempts to provide the correct outputs for the appropriate inputs, without specific regard to the machinery within. It is the functionality of the motor system, not the anatomy, that we strive to reflect.

A simulation cannot prove a hypothesis, it can only lend support. The strength of simulations lays in their ability to rule out, or display consistency of, a set of assumptions and a corresponding hypothesis. This model, is in essence, a set of assumptions built to support a hypothesis of the human motor system. If the model meets the criteria presented, and is consistent with empirical data, then it has proved itself to be consistent with the human motor system. This does not mean to say that no conclusions can be drawn from this model. On the contrary, if the model does characterize human reaching movements well, then a systematic analysis of the model might lend insight into the sources of human pathologies. If by "breaking" the controller, it exhibits pathological reaching movements analogous to those of humans with movement disorders, the broken machinery of the controller may elucidate the functions of the malfunctioning human mechanisms.

Chapter 6

Experiment

6.1 Objectives

The experiment conducted here is an attempt to asses the proposed architecture's properties as a motor controller and to further determine if it can exhibit motor learning. To answer both questions a series of tasks have been devised that, if successfully completed, will call upon both qualities. Based upon these findings we will be in a position to comment on similarities between the current control scheme and the human motor system and the implications for a biomimetic controller.

6.2 Overview

As explained previously the "hardware" for the experiment will be a simulated two-link planar arm. The arm will offer a test bed upon which to examine the controller. The tasks chosen for the controller have a moderate level of difficulty in that they encompas's most of the arm's range of motion, thereby hindering any potential linearizations the controller might employ (e.g. in learning the internal models). Eight targets equally spaced at 45^0 around a central home position will be used for the often duplicated center-out reaching movements. Each target sits on the circumference of a $20cm$ radius circle, with its center situated at $(0.0,\ 15.0)cm$ in the arm's reference frame. Beginning with the top most target and continuing in a clockwise fashion

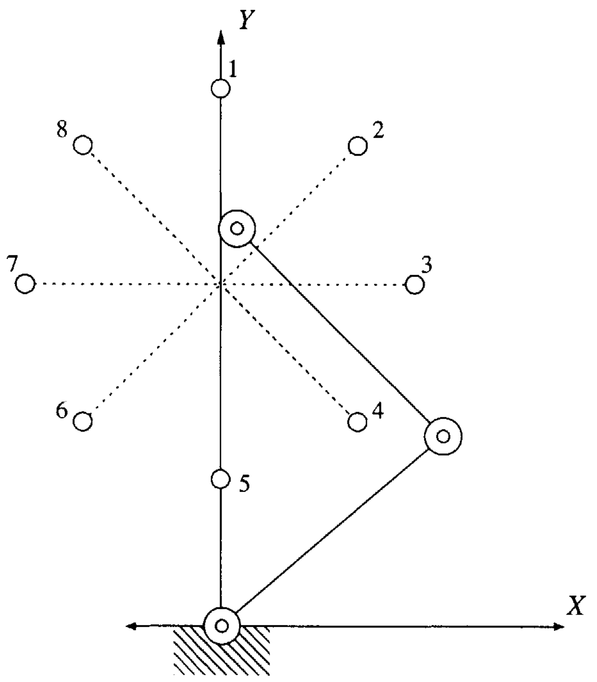

Figure 6-1: Diagram of target geometry

they are labelled targets 1 through 8 (figure 6-1). This archetype for human motor control experiments has been used on numerous occasions, thus readily allowing for a comparison of the results found here with previous experiments. The first task for the controller to undertake will be to learn to generate the necessary commands for making reaching movements to the eight targets. The time varying coordinates for the desired minimum jerk trajectory shall be provided to the controller. In this initial task the controller will only be equipped with a single module, allowing for a single behavior to be learned. The controller will be trained on-line to produce a minimum jerk profile movement. These movements will constitute the controllers first context, unperturbed reaching movements. Once the controller has been adequately trained the performance shall be evaluated.

Task two will consist of learning the same reaching movements, but now within a curl field. Again, only one module will be utilized for this task. A counter clockwise velocity dependent perturbation will be applied to the endpoint as described below.

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = \begin{bmatrix} 0 & B_c \\ -B_c & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{6.1}$$

The strength of the field, B_c is $40N \cdot s/m$. Training will proceed just as before on this, the second context, and again the resulting performance shall be evaluated.

The first two tasks of the experiment, while demonstrating the control architectures abilities for motor control and motor learning, do not examine the use of multiple paired models for learning and utilizing multiple behaviors. Endeavoring to do just that, the third task shall challenge the controller to learn the second context, the curl field, when equipped with two modules. The controller will be outfitted with the previously trained module for unperturbed reaching movements and another empty, or untrained, module. Training will take place intermittently just as before, but in this task the controller will be free to train either module as the responsibility signal dictates. The outcome of this task shall be compared with the results from the previous single module tasks.

A final task will utilize both previously trained modules for contexts one and two (unperturbed and curl field) to observe the controller's ability to generalize previously learned behaviors for a newly encountered context. The third context shall be a curl field, in the same orientation, but of half the strength of the previous one. Therefore, neither context one nor two shall be sufficient to generate the necessary motor commands and the controller will have to adapt. The assumption to be tested is that the controller can blend the unperturbed and curl field modules to achieve acceptable performance with minimal alterations to the previously learned motor behaviors.

6.3 Procedure

Here we shall present the steps that lead to the completion of the experiment. Although, in theory, it is possible to have all the network models (forward and inverse, kinematic and dynamic) adapt and train at once, it is definitely not the most effective method; the training process couples the models (feedback error and distal supervisor learning) and convergence of a large untrained architecture such as this would likely take a long time, if it would happen at all. In fact, the dynamic models, though included to compensate for the dynamics of motion during a movement, are not restricted as such and might produce a bias in the kinematic models, if the kinematic models were not well trained before the dynamic models. For these reasons the models will be, at least in part, trained sequentially.

The kinematic models are first in line to be trained, either forward and then inverse, or both at the same time. With both kinematic models trained to satisfaction they shall be held constant for the remainder of the experiment. With the kinematic models training complete, they can be employed while training the dynamic models. Again, it is likely more effective (assuming a distal supervised learning scheme) to train the forward dynamic model at least partially before attempting to train the inverse model. Training of the forward models and inverse models shall be conducted during the completion of tasks one and two, training in the unperturbed and the curl contexts. The results of these two tasks shall be examined for trajectory following performance.

For the third task of the experiment the initial variances shall be set according to the results of the previous two tasks. The responsibility signal will be initialized to [.9, .1] for the unperturbed and empty module respectively. These conditions will describe a situation in which the controller has previously been making unperturbed reaching movements using 90% of the unperturbed module, and 10% of the empty module. The controller will train on-line adapting the modules as dictated by their responsibility signals. This task shall examine if the controller is capable of retaining multiple motor behaviors.

During the fourth and final task, the controller will be outfitted with the modules previously trained in tasks one and two. The controller will then be trained in the half-strength curl field. The results will be examined to determine how the controller adapted and if it was able to blend the two modules without retraining them or if further training was necessary.

Chapter 7

Results

Here the results of the experiments are presented, not simply the numbers, but the story as it unfolded in an attempt to better elucidate what was done and why. First the physical parameters chosen for the simulated arm-like manipulator will be discussed. Then some preliminary work done, designing and training the networks, will be discussed. These preliminary trials, were a means to determine appropriate guidelines while undertaking the four tasks of the experiment, which will be covered last.

7.1 Physical Parameters

Keeping with the theme of an arm-like manipulator the dimensions and inertial parameters were chosen from anthropomorphic tables based on a person of approximately $70kg$. The upper arm length, l_1, was set to $.305m$, with a center of mass location l_{1cm}= $.131m$. The forearm, though typically shorter in length, was also set to $.305m$ to account for the added distance from the wrist to the hand. The center of mass, l_{2cm}, for the forearm, was also set to $.131m$. The mass of the upper arm and forearm was modeled as 2.244 and $1.428kg$. The mass moments of inertia were modeled as cylinders, $\frac{1}{12}ml^2$; I_1 and I_2 were 17.396×10^{-3} and $11.070\times10^{-}3$ respectively.

In Flash's [9] measurements of human arm stiffness during reaching movements values as high as $90Nm/rad$ were reported. Other studies however, have reported much lower values [12, 26]. Here we take the middle ground and use intermediate values,

$$\boldsymbol{K} = \begin{bmatrix} 40 & 20 \\ 20 & 50 \end{bmatrix} (N \cdot m/rad) \tag{7.1}$$

$$\boldsymbol{B} = \begin{bmatrix} 4 & 2 \\ 2 & 5 \end{bmatrix} (N \cdot s/rad) \tag{7.2}$$

7.2 Network Tuning

One advantage of the GRBF network is that the response for any given input is local; if the input vector is more than approximately three standard deviations away from a node's center (see Chapter 3) the node will not respond. Thus for the sake of computational efficiency, one would be inclined to "grab" only the pertinent nodes when computing the network's response, or when training, rather than blindly compute each node's response. This can be done by finding a hypercube (or hypersphere) around the input vector. In order to minimize the number of nodes within this hypercube the "width" of each node's response, or the variance σ^2 (see equation 3.3), should be minimal. Yet there are competing factors as well, for as the variance decreases, the overlap between neighboring nodes decreases, possibly hindering the networks ability to approximate the desired function. Consider the extreme case of a vanishingly small variance. Each node's response would degenerate to a spike at it's center. The network might achieve acceptable values at inputs that precisely corresponded with the network's centers, but clearly the network's ability to interpolate between nodes would be ineffective. At the other extreme, when the variance is infinitely large, each node is simply a hyperplane, and the network is then a linear sum of all the weights, regardless of the input. Further complicating the issue is the number of nodes that are used in the network, for all things being equal, more nodes will generally translate to better accuracy in the estimate.

To arrive at a workable solution several models were created with a fixed number of nodes but varying ratios of variance to mesh size, σ^2/δ. The mesh size, δ is the distance between two neighboring nodes. For simplicity δ was a constant value, the input dimension divided by the number of nodes within that space. These models were all trained on the forward kinematic mapping from joint angles to endpoint coordinates. As the ratio decreased from three to one the error in the model's estimate decreased as well. When the ratio was further decreased to 0.9 the error worsened. Based on these results and suggestions in previous GRBF network research [30], the ratio was set to 1 for the kinematic models as well as the dynamic models. Notice though, that with this ratio the radius of a hypersphere would have to be at least three nodes. For all of the networks used in this work this radius either enveloped the entire network, or the overhead involved in computing the hypercube introduced more computation than had it not been used at all.

7.3 Kinematic Models

In principle, the kinematic models could be trained by moving the arm slowly (such that the arm was in a quasi-static state), or by repeatedly moving the arm and allowing it come to rest before training. But in the interest of time, the forward and the inverse kinematic models were trained using the forward kinematic relation, $\boldsymbol{x} = L(\boldsymbol{\theta})$, equation 4.1. In addition, although the kinematic models have relatively low dimensionality ($\mathcal{R}^2 \Rightarrow \mathcal{R}^2$), and a least squares solution could be computed for the network by collecting a large data set, an on-line training algorithm was used such that the training of the networks would remain consistent throughout. All network's weights were initialized to zero.

7.3.1 Forward Kinematic Model

Before training could begin, the domain of the forward model had to be clearly defined. Unlike the classic feed-forward neural network, the nodes of the GRBF network are explicitly tied to their domain. If a node's center falls outside the domain of interest, then it will always show little if any response. To map out an appropriate domain the arm's workspace was finely meshed ($5mm$) from $(-20, 15)cm$ to $(20, 55)cm$. The joint angles for each end point position were determined and taking the maximum and minimum joint angles (and a small safety factor) the domain for the forward kinematic model, $\mathcal{D}_{fk}$ was defined to be,

$$\mathcal{D}_{fk} = \{\theta_1|_{rad} \; -1.0 \leq \theta_1 \leq 2.0; \;\; \theta_2|_{rad} 0.0 \leq \theta_2 \leq 3.0\}$$

The centers were then computed by evenly discretizing the domain amongst the nodes of the network.

To train the networks the workspace was again meshed in evenly spaced Cartesian coordinates, in increments of $\Delta x = \Delta y = 2mm$ with endpoint coordinates $\{\boldsymbol{x}\}_i = \{x, y\}_i$. These coordinates were then transformed into joint coordinates $\{\theta_1, \theta_2\}_i$, and used to train the network. Using a supervised learning scheme, the network was trained to minimize the cost function,

$$\mathcal{L}_k = \frac{1}{2}||\boldsymbol{x} - \hat{\boldsymbol{x}}||_i^2 \tag{7.3}$$

where the subscript refers to the i^{th} sample while training on-line. Initial training revealed that great accuracy was possible with these models. Therefore for practical purposes training was halted when the error was within what was judged to be reasonable bounds, rather than when the models had converged. Using a step size α_{fk} of 1 (equation 3.6) the network was trained until the average RMS error from one training session to the next was less than a determined tolerance ($\approx .5mm$). After training several networks of various sizes, using a step size α_{fk} of 1 it was decided that a network with 49 nodes (7 wide in each dimension, θ_1 and θ_2) provided adequate precision. Although the model's error dropped quickly within the first 50 passes over the data, the error continued to decrease in a manner reminiscent of an exponential decay. After approximately 300 passes over the workspace, validating the network reveals RMS error in $\hat{x}$ and $\hat{y}$ was $1.5mm$ and $0.4mm$, respectively. Figure 7-1 is a plot of the error vectors, in end point coordinates, over the workspace. The error vectors originate at desired coordinates and terminate at estimates of said coordinates (note that this is plotted with respect to the same coordinate frame as figure 4-1).

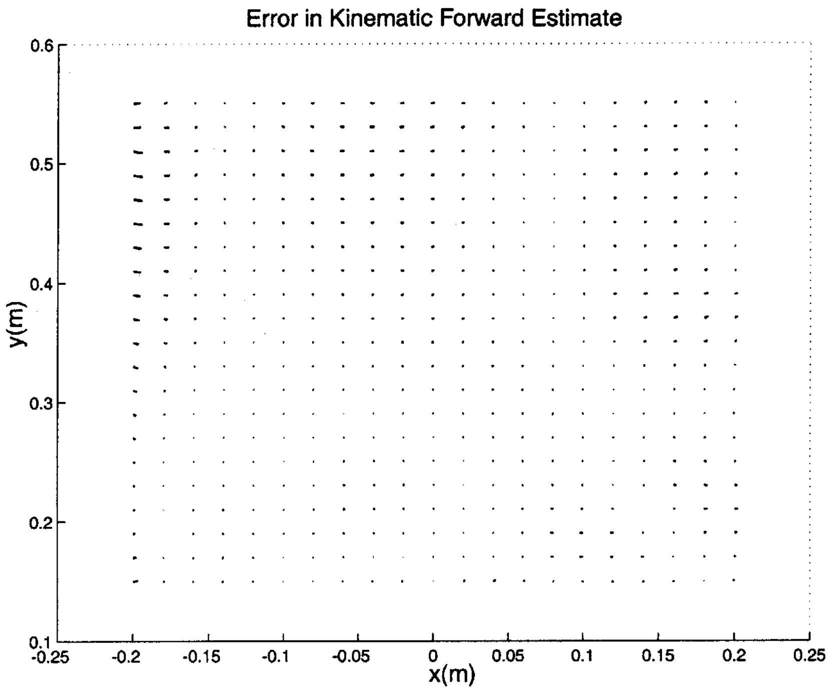

Figure 7-1: Error in forward kinematic estimate of endpoint coordinates. Error vectors originate at desired coordinates and terminate at estimates of said coordinates.

7.3.2 Inverse Kinematic Model

As with the forward kinematic model, the inverse model's domain had to be clearly defined before training could proceed. In this instance it was less difficult though, the domain was chosen to be a square, $10cm$ wider on each side than need be, to encompass all the targets,

$$\mathcal{D}_{ik} = \{x|_m \ -0.25 \leq x \leq 0.25;\ y|_m 0.10 \leq y \leq 0.60\}$$

Then, just as before, the domain was evenly spaced with 49 nodes. The distal supervised learning scheme was used to overcome the problems outlined in Chapter 3. The cost function to be minimized now related the actual endpoint coordinates (mapped forward from the inverse kinematic model's estimate) and the desired endpoint coordinate, $\boldsymbol{x}^d$,

$$\mathcal{J}_k = \frac{1}{2}\|\boldsymbol{x}^d - \boldsymbol{x}\|_i^2 \tag{7.4}$$

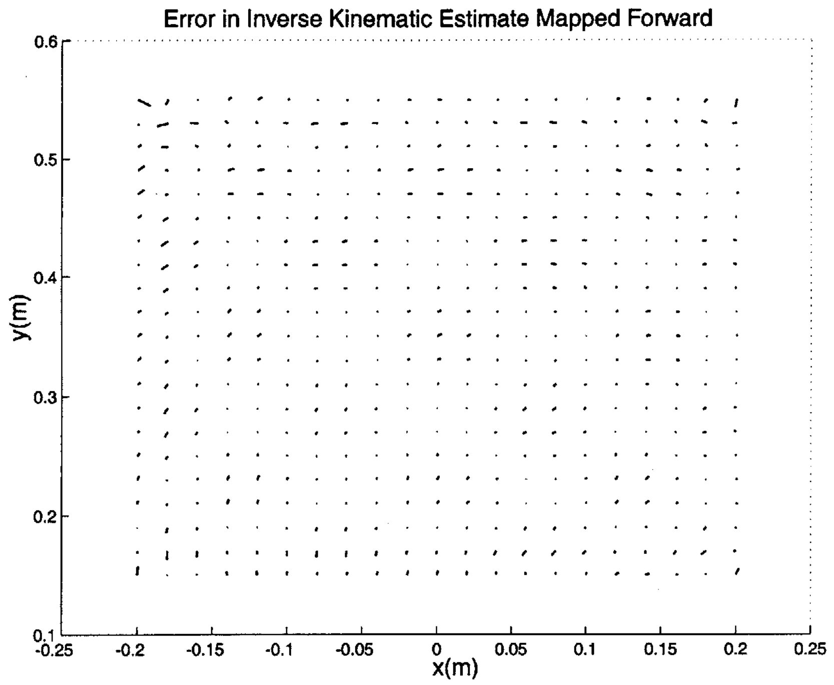

Figure 7-2: Error in inverse estimate of endpoint coordinates. Error vectors originate at desired coordinates and terminate at resulting coordinates when mapped forward from the inverse estimates.

First the training proceeded with the forward model already learned. Similar to the forward models, training stopped when the average RMS over several sessions dropped below tolerance ($\approx .5mm$). After 250 training sessions, the RMS error in radians was 5.6×10^{-3} and 2.7×10^{-4} in $\hat{\theta}_1$ and $\hat{\theta}_2$. With the inverse models, the effect the error in radians would have on the forward mapped end point, which is what was ultimately desired, was not intuitive. Therefore the inverse model's estimates of joint coordinates were mapped forward using equation 4.1. The result of 300 training sessions is shown in figure7-2. The RMS error in endpoint coordinates is $1.8mm$ and $1.7mm$ for x and y. For the distal supervised approach, the forward model need not be trained beforehand, so another training session was attempted with both the forward and inverse kinematic models untrained. Training took approximately the same number of sessions and there was little difference in the final results.

7.4 Dynamic Models

With the dynamic models the problem of dimensionality, and its impact on the GRBF network truly becomes apparent. The kinematic models, for instance, had relatively few nodes, 49, by discretizing their domain with 7 centers in each dimension ($7^2 = 49$). The dynamic models on the other hand, have a larger input space; each network requires 6 inputs. The forward dynamic model has position, velocity, and equilibrium coordinates as an input, and the inverse dynamic model has position, velocity and desired position as an input. If it were to have it's domain discretized with the same number of centers per dimension it would require $7^6 \approx 118,000$ nodes. For the task at hand this is an impractical amount of memory to work with on a personal computer. For this reason the dynamic networks were restricted to smaller sizes, 3 or 4 centers per dimension.

To remain consistent, the dynamic models used the same ratio of variance to mesh size as did the kinematic models, and were also initialized to zero valued weights. Again the training was performed on-line.

7.4.1 Forward Dynamic Model

Just as with the kinematic models, before training could begin, the domain had to be properly determined. Providing the arm with a minimum jerk equilibrium coordinate trajectory (mapped to joint coordinates), for each target, the resulting motion was determined according to equation 4.10. Each center-out movement began with zero initial conditions and was simulated for 1.2 seconds in the unperturbed and perturbed (curl field) contexts. Figure 7-3 is a plot of the endpoint in the unperturbed context. Figure 7-4 shows the endpoint in the curl field context.

The resulting velocity and position profiles of these simulations were used to place bounds on the domain of the forward models $\mathcal{D}_{fd}$,

$$\mathcal{D}_{fd} = \left\{ \begin{array}{c} x|_m \; -0.25 \le x \le 0.25; \; y|_m 0.10 \le y \le 0.60; \\ \dot{x}|_{m/s} \; -0.50 \le \dot{x} \le 0.50; \; \dot{y}|_{m/s} \; -0.50 \le \dot{y} \le 0.50; \\ u_1|_{rad} \; -1.0 \le u_1 \le 2.0; \; u_2|_{rad} 0.0 \le u_2 \le 3.0; \end{array} \right\}$$

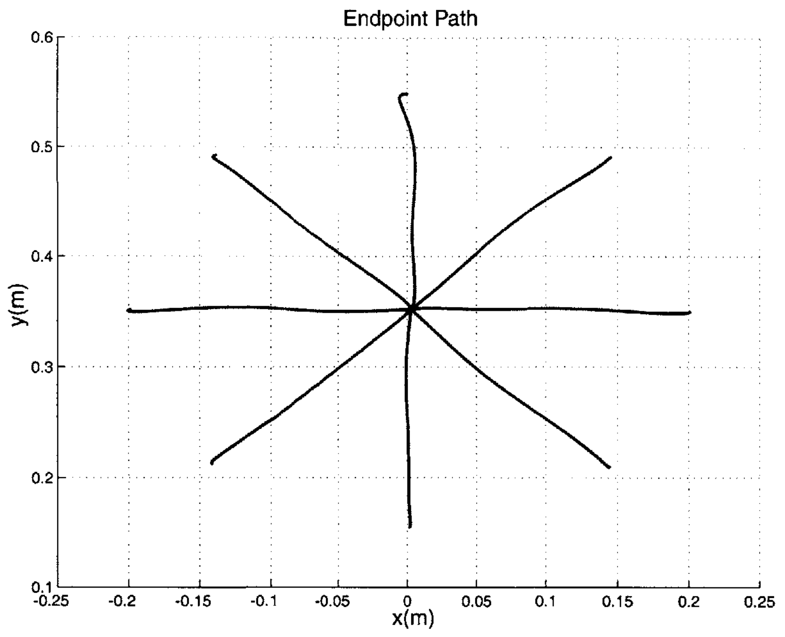

Figure 7-3: Resulting path of endpoint in unperturbed context when commanded with a minimum jerk trajectory. The RMS error from desired path is $2.0cm$.

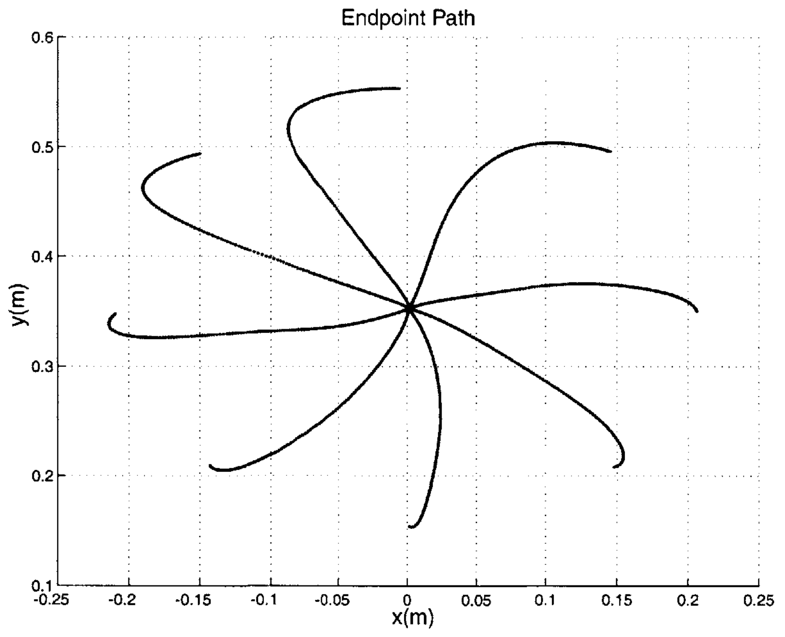

Figure 7-4: Resulting path of the endpoint in the perturbed, curl field context, when commanded with a minimum jerk trajectory. The RMS error from a desired path is $3.5cm$.

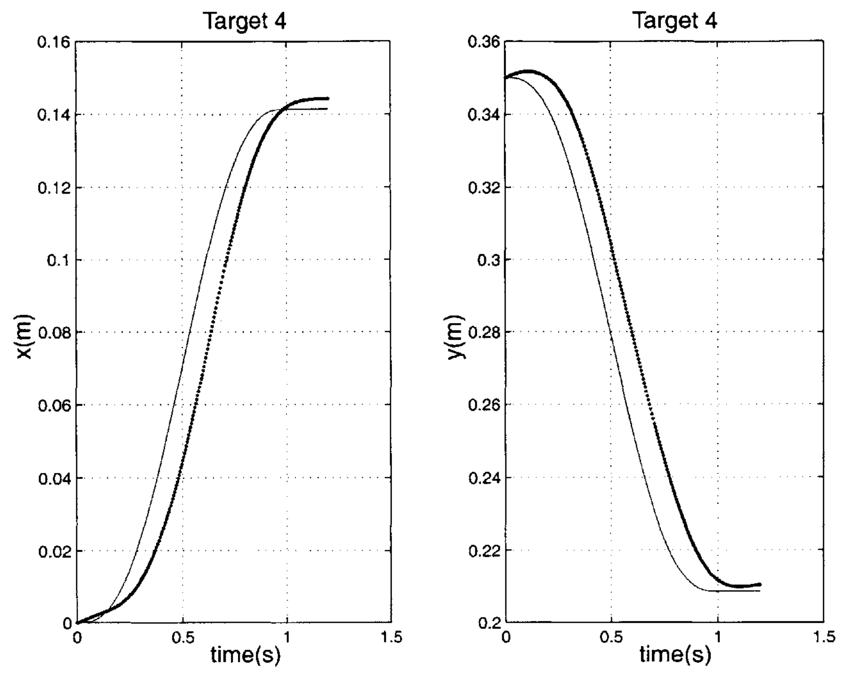

Figure 7-5: x and y components of the endpoint's path in the unperturbed movement to target 4. The solid lines represent the desired minimum jerk path, and the dotted lines are the actual paths.

As previously explained, the values chosen to represent muscle stiffness do not allow for good trajectory following. The top view of figures 7-3 and 7-4 showing the endpoint path with and without perturbations, respectively, are slightly deceiving in that the time component to the movements is not visible. Looking at a typical movement's x and y components of position and the desired trajectory (figure 7-5) the error is more obvious. The unperturbed RMS error in position was $2.0cm$, an error of 10% of the reaching movement. The curl field movements had an RMS error in position of $3.51cm$.

The forward dynamic model was initially trained by itself, so it could then be used to train the inverse model. Just as above, this was done by driving the arm with a minimum jerk profile. Using a fixed time interval of $5ms$, the forward model was intermittently presented with the pseudo-state and and the equilibrium joint coordinates, and then trained on the error between it's estimate and the actual endpoint acceleration at that time. Figure 7-6 is a schematic representation of how the con-

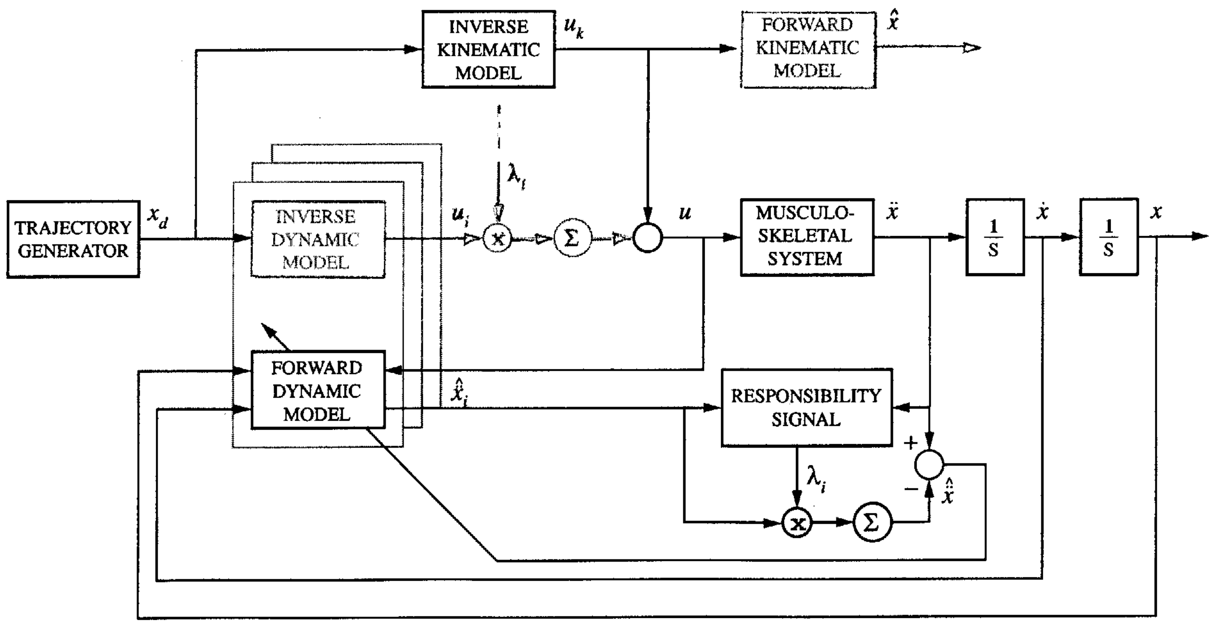

Figure 7-6: Diagram of control system while training the forward dynamic models (shaded portions are not used while training).

trol system was used to train the forward dynamic models. Using a gradient descent algorithm and a supervised learning scheme the cost function minimized was,

$$\mathcal{L}_d = \frac{1}{2}||\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}||_i^2 \tag{7.5}$$

In contrast to the smaller kinematic models, these larger dynamic models went unstable with a step size of 1. A line search method was tried, stepping the weights forward until the error was worse than the previous step, then halving the step and moving in the opposite direction. But in the end, a fixed step size was found to be the most practical method.

Training these models proved more difficult than the kinematic models in many respects. Unlike the kinematic models, with their robust behavior and apparent monotonic decay in error, the dynamic models required attentive care while training. When on-line training, the networks adapt their weights to decrease the error perceived locally; that is to say, there is no consideration for any adverse effects local training will have on global error. For instance, while training these dynamic models on target 2, in the upper right of the workspace, the network weights, though decreasing the error about this region of the domain, might be increasing the error in forward estimates while the arm is reaching for target 6 in the lower left half of the work space. The underlying cause of this dilemma is the inherent imprecision in the network model. Demanding too much precision from the model will force it to continually modulate its weights to meet the demands of training. This problem can be remedied with a boundary layer. A boundary layer on the error in the forward estimate of the models was enforced such that no training would take place when the forward model's estimate was within this magnitude of error. Progressively smaller boundary layers were used while training the forward dynamic models until validating the model revealed the overall error worsened. It is for these reasons, local training and boundary layers, that frequent validation of the models during the training process was a more prudent measure of progress than the error while training.

Validating the forward models was accomplished by observing a time history of the network's estimates during reaching movements to both the eight targets, and eight targets rotated 22.5^0 clockwise to untrained locations. The time history validation for a trained forward model with 729 nodes (3 centers per dimension) can be seen in figures 7-7 and 7-8. In both plots, the solid lines denote the actual acceleration, and the dotted lines denote the forward model's estimate. These models, trained approximately 100 times on each target reach, had an RMS error in their forward estimates of $\hat{\ddot{x}}$ and $\hat{\ddot{y}}$ of $2.68\times10^{-1} m/s^2$ and $2.39\times10^{-1} m/s^2$ respectively.

It was hoped that the forward models might attain a greater precision without adding an unreasonable number of nodes, and it was decided that 4 nodes per dimension, a total of 4,096 nodes in all, would be the largest acceptable network for the forward models. Therefore, rather than simply jump up to this larger model it was thought that a more prudent choice would be to sequentially add nodes in each dimension to determine if the network's resolution could be enhanced greatly by increasing the centers in any one dimension (e.g. 3 centers in $x, y, \dot{x}, \dot{y}$, but 4 centers in u_1 and u_2) or if the resolution increased proportionately in all dimensions. Training of these intermediate sized models revealed no clear advantages to increasing the nodes in any one dimension over another.

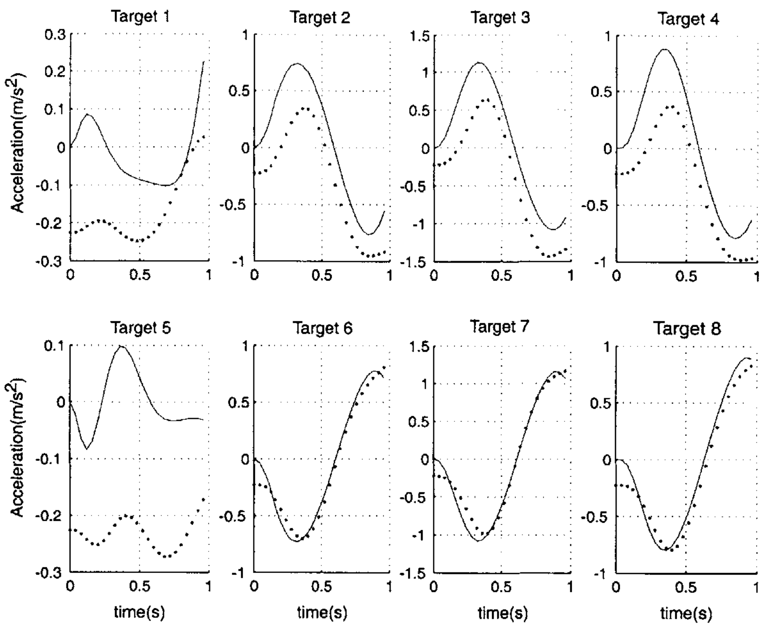

Figure 7-7: Forward estimate of the x component of the acceleration of the endpoint using a model with 729 nodes. The solid lines are $\ddot{x}$, and the dotted lines are $\hat{\ddot{x}}$. The RMS error in acceleration was $2.68\times10^{-1}m/s^2$.

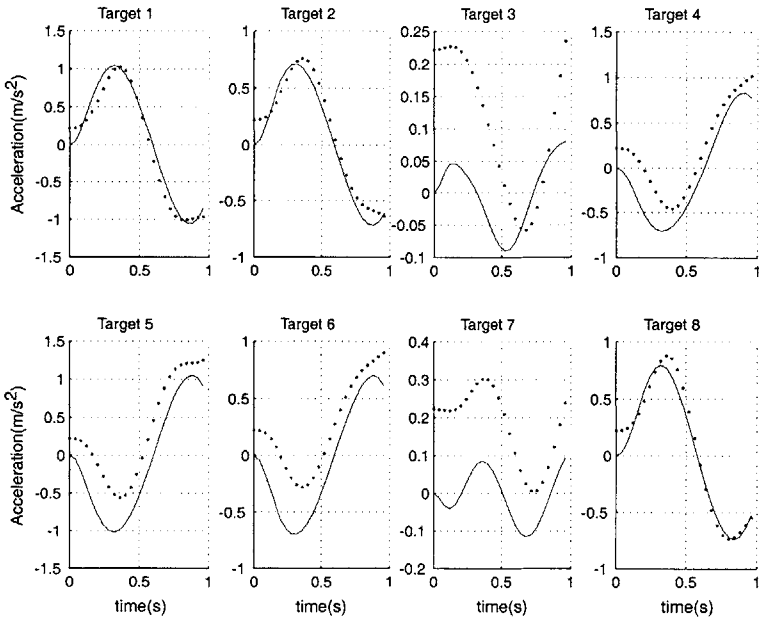

Figure 7-8: Forward estimate of the y component of the acceleration of the endpoint using a model with 729 nodes. The solid lines are $\ddot{y}$, and the dotted lines are $\hat{\ddot{y}}$. The RMS error in acceleration was $2.39\times10^{-1}m/s^2$.

Although larger networks resulted in better estimates, they took longer to train. In the end though, it was decided to go with the largest network, 4 nodes per dimension. This allowed for the greatest resolution that would have to be paid for in terms of number of nodes and commensurate computation time. This was justified for two reasons. For one, the forward models were originally intended to be used to train the inverse models using the distal supervised learner, so an accurate estimate would be helpful. Second, the switching scheme for multiple modules is dependent on the accuracy of each modules forward estimate. The more precise the forward estimate, the more clear the distinction between the modules. The trained forward model for the unperturbed context, with 4,096 nodes, can be seen in figures 7-9 and 7-10. These models were trained with a tolerance (boundary layer) of approximately $1.7\times10^{-1}m/s^2$, and a step size α_{fd} of 0.2, for over 100, 8 target sessions.

7.4.2 Inverse Dynamic Model

Using the values previously obtained for the forward dynamics model's domain (both the forward and inverse dynamics models have $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ as inputs) the domain for the inverse models was defined as,

$$\mathcal{D}_{id} = \left\{ \begin{array}{c} x|_m \; -0.25 \leq x \leq 0.25; \; y|_m 0.10 \leq y \leq 0.60; \\ \dot{x}|_{m/s} \; -0.50 \leq \dot{x} \leq 0.50; \; \dot{y}|_{m/s} \; -0.50 \leq \dot{y} \leq 0.50; \\ x_d|_m \; -0.20 \leq x_d 0.20; \; y_d|_m 0.15 \leq \; y_d \leq 0.55; \end{array} \right\}$$

The original goal was to train the inverse models using the distal teacher. Using the forward model's estimate of the dynamics, the inverse dynamic models would minimize a cost function relating the difference in the actual and desired acceleration,

$$\mathcal{J}_d = \frac{1}{2}||\ddot{\boldsymbol{x}}^d - \ddot{\boldsymbol{x}}||_i^2 \tag{7.6}$$

As already discussed in section 3.4, one of the drawbacks to the distal teacher arises with large networks such as the ones used here. To compute the Jacobian $(\partial\hat{\ddot{\boldsymbol{x}}}/\partial\boldsymbol{u})^T(\partial\boldsymbol{u}/\partial\boldsymbol{w})^T$, one must make something on the order of $N_{fd}\cdot N_{id}$ calculations. Where N_{fd} and N_{id} are the number of nodes in the forward and inverse dynamic

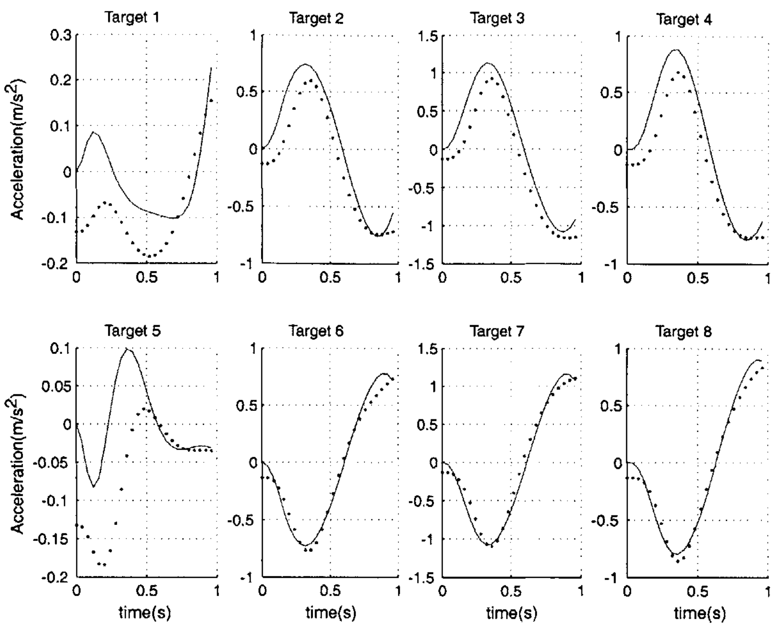

Figure 7-9: Forward estimate of the x component of the acceleration of the end point using a model with 4,096 nodes. The solid lines are $\ddot{x}$, and the dotted lines are $\hat{\ddot{x}}$. The RMS error in acceleration was $1.55 \times 10^{-1} m/s^2$.

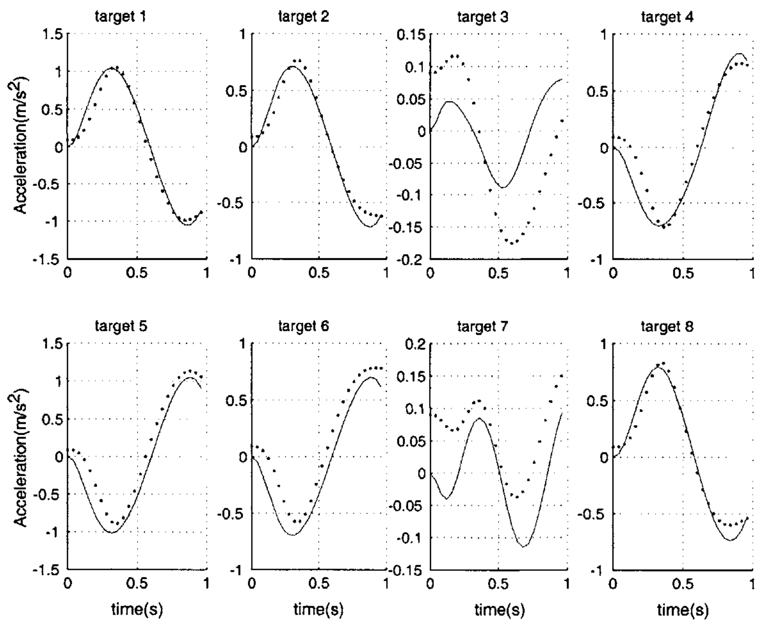

Figure 7-10: Forward estimate of the y component of the acceleration of the endpoint using a model with 4,096 nodes. The solid lines are $\ddot{y}$, and the dotted lines are $\hat{\ddot{y}}$. The RMS error in acceleration was $1.30 \times 10^{-1} m/s^2$.

models respectively. This is a time consuming process. Even when relatively small networks were used the computation time was unreasonably slow. One particular attempt to train an inverse dynamic model with a distal teacher, when both forward and inverse models had only 729 nodes, took over 8 hours on a Sun Ultra 10 personal work station and only completed 16 moves, a total of little more than 19 simulated seconds. Keeping in mind the number of training sessions required to sufficiently train a model, the distal supervised approach was not pursued further.

With the distal supervised learner paradigm ruled out, only one option remained, [1] the feedback error approach. To use this scheme an error signal would have to be computed in terms of the inverse model's output, joint coordinates. Using the inverse kinematic model we can obtain a signed error signal as follows,

$$\tilde{\boldsymbol{u}} = \phi_k(\boldsymbol{x}^d) - \phi_k(\boldsymbol{x}) \tag{7.7}$$

We should emphasize that this is still consistent with the biologically motivated control architecture. The feedback signal does not rely on an exact analytical expression but learned kinematic models. It is not unrealistic to imagine that the central nervous system can generate this error term by computing the direction in which the joints need to move.

To train the inverse models, randomly generated targets were selected and a time varying desired (minimum jerk) Cartesian trajectory was computed and provided to the control system. Again, the movements were intended to reach the target after 1 second. The arm dynamics (equation 4.10) employing the feed-forward kinematic component and the feedback inverse dynamic component for control were integrated in Matlab. At time intervals of $5ms$ the inverse model was trained on the error in position determined through the feedback signal equation 7.7. Figure 7-11 is a schematic representation of how the control system was used to train the inverse dynamic models. This scheme worked much faster than the distal supervised approach. A typical

[1]direct inverse learning was not an option for it would require continuously inverting the dynamics to compute the desired equilibrium trajectory, a computation that this architecture is not equipped to implement

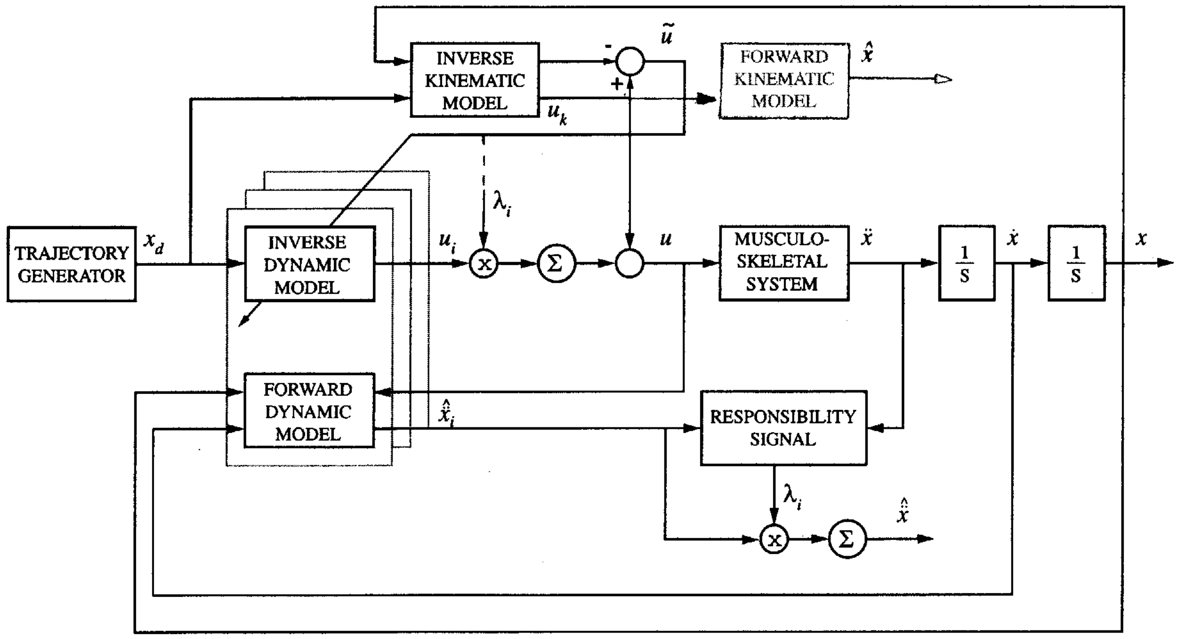

Figure 7-11: Diagram of control system while training the inverse dynamic models (shaded portions are not used during training).

simulation took approximately 1 minute per simulated second, as opposed to the 25 minutes per simulated second using the distal supervised teacher.

Considering the difficulties in training the forward models, boundary layers were an obvious option for the inverse dynamic models. While training the inverse models, various tolerances were placed on the system to halt the training process should the trajectory following error fall within acceptable bounds. The inverse models were likely not as precise in their commands as the kinematic models (3 centers per dimension as compared to the kinematic model's 7 centers per dimension) therefore tolerances close to those of the kinematic model's RMS error were used.

Various step sizes were used and their effects observed to determine how fast the inverse model would converge. It was found that not only did larger step sizes allow the models to converge faster, they would also yield ultimately smaller overall errors in the networks performance. In figure 7-12 we see the results of training for the unperturbed reaching movements with a step size of α_{id} of .1. The respective RMS position error (deviation from the desired minimum jerk trajectory calculated over the

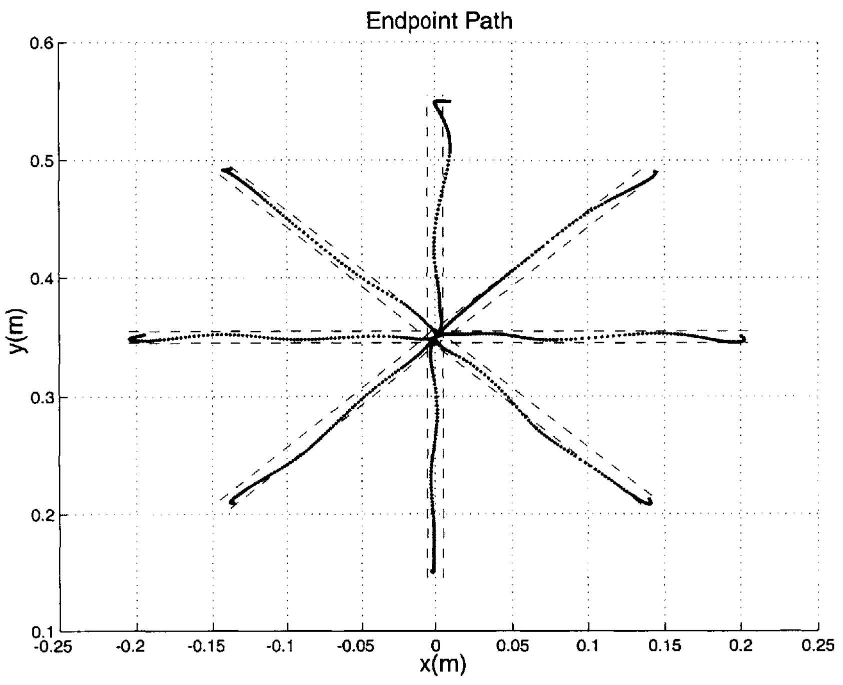

Figure 7-12: The endpoint path after approximately 150 training sessions. The RMS position error is $3.02mm$ in x and $1.81mm$ in y. Dotted lines denote the $4mm$ tolerance.

eight targets) was $3.02mm$ and $1.81mm$ in x and y respectively. This was completed after training approximately 150 times on each target, with a training tolerance of $4mm$.

Once again, it is likely that larger networks could produce better accuracy than these models with 729 nodes. Yet the inverse dynamic models must be integrated along with the arm while simulating. With an increase in number of nodes the simulations would greatly increase in time, making the experiments a highly time consuming venture.

7.5 Experimental Results

Now we present the results of tasks one through four. Though the individual steps have already been made, tasks one and two were not considered complete because they were trained under constrained conditions, that is, the forward and inverse models were not coupled and acting together in the controller. The findings above were viewed as preliminary results and used to help perform the four tasks of the experiment.

7.5.1 Tasks 1 and 2

For tasks one and two the controller was assembled as in figure 5-1. For these single behavior tasks, one and two, the controller was equipped with a single, empty, or untrained, module, so the responsibility signal need not be computed. The time step was again $5ms$ and training tolerances for the inverse and forward models were in the neighborhood of $.2m/s^2$ and $3mm$ respectively. Targets were generated at random and the time varying minimum jerk coordinates were presented to the controller. The inverse models learned quickly but not so for the forward models. Reviewing the data reveals a trend in the training of the forward models. Early during training, the forward model was capable of learning and estimating the acceleration components of the arm. As the error in trajectory following decreased and the inverse model was driven into its boundary layer the acceleration became more volatile, experiencing many sudden and large reversals in sign. The forward model's estimates during these conditions were limited. Viewing the data we see that this was often due to shifts in the control input due to training. When the endpoint fell outside the boundary layer, training would push it's equilibrium coordinate in the opposite direction, suddenly creating a large difference between the equilibrium and actual joint coordinates. This jump caused large accelerations due to the stiffness matrix (equation 4.9).

Figures 7-13 and 7-14 show the end point acceleration during one such movement to target 1. Figure 7-15 shows the corresponding endpoint path to target 1. Observe how the equilibrium coordinates of the path (dotted line) bounce back and forth across the desired path and how this in turn creates the large acceleration spikes in the plots (figures 7-13 and 7-14).

These large variations in the forward dynamics seem to have hindered the forward model's ability to learn accurate estimates. This had little effect on the inverse models, which could still achieve adequate performance in trajectory following. Yet

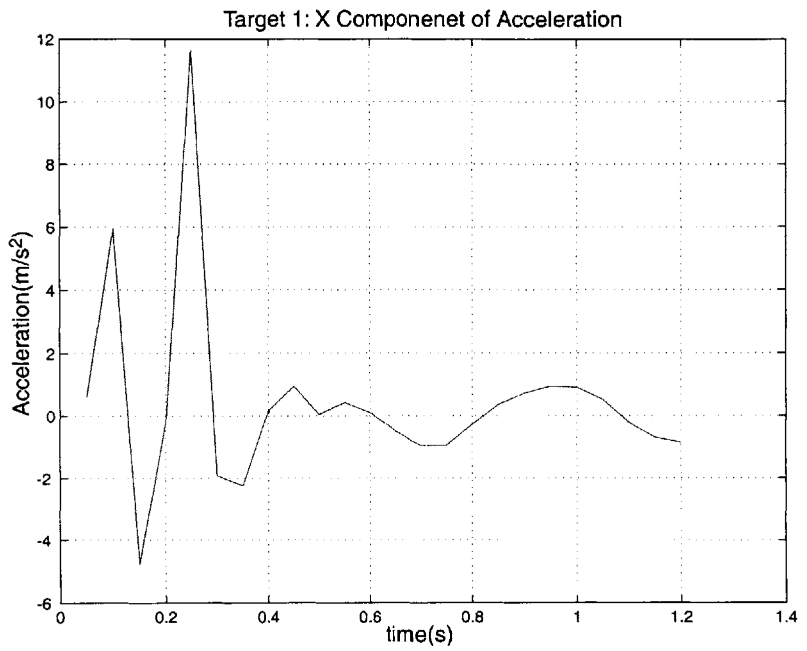

Figure 7-13: Acceleration due to boundary layer dynamics, x component.

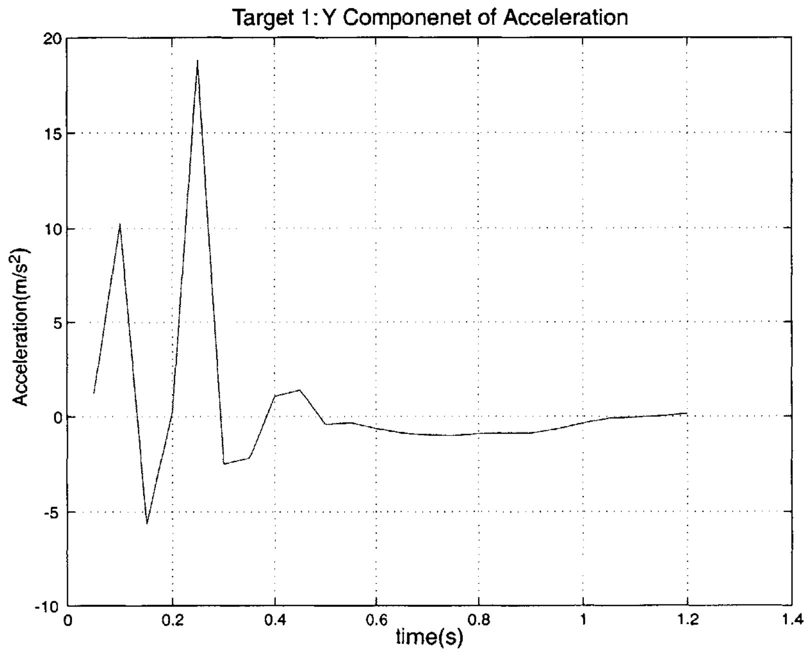

Figure 7-14: Acceleration due to boundary layer dynamics, y component.

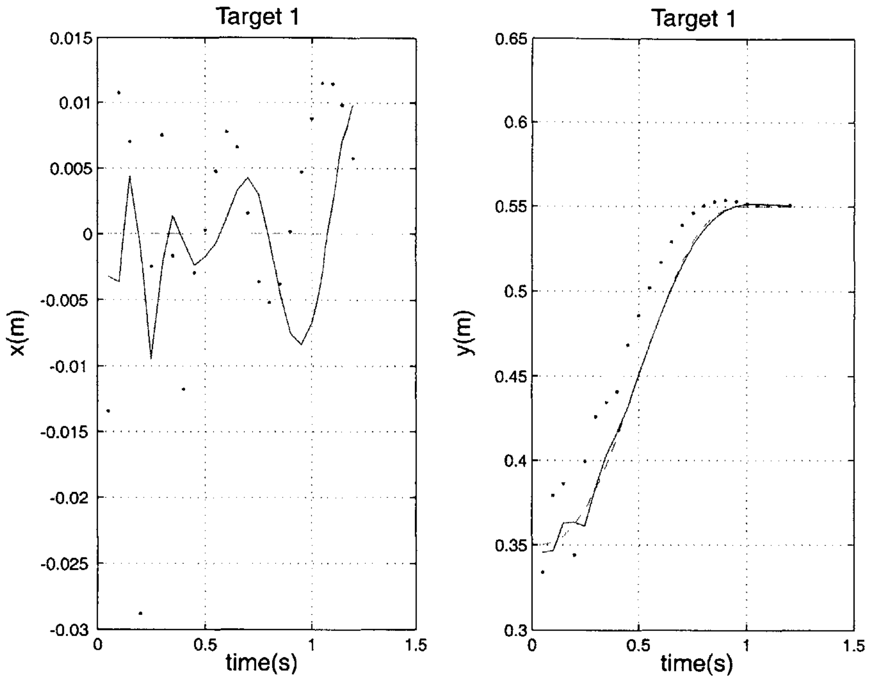

Figure 7-15: Endpoint path to target 1. Solid lines are the actual paths, dotted lines are the equilibrium coordinates during movement, and the dashed line is the desired path.

poor forward estimates degrade the controllers ability to switch between contexts. The architecture relies solely on the dynamic model's estimate of the actual endpoint acceleration while switching between contexts; if it is not accurate, then the controller won't reliably switch. Early attempts at using two modules were not very successful for these reasons. The dynamic model's estimate was not precise enough to distinguish between the two contexts and the responsibility for both modules tended to .5, disallowing the possibility for two distinct modules.

To combat these problems and improve the forward estimate two approaches were taken: the boundary layer for the inverse model was increased, and the step size was decreased. The rationale for the first approach was that by decreasing the pressures on the controller in terms of trajectory following, the inverse model would push the position error within its boundary layer quickly and once inside would be sufficient to keep it there. Hence, there would be no more training, and fewer large variations in the forward dynamics. Invariably though, once inside the boundary layer, the position error would rebound occasionally. Now, however, when outside this larger boundary layer there would be an even larger sudden increase in motor command, for the training is proportional to the error. This would cause larger increases in end point acceleration, making it harder for the forward models to learn a proper estimate.

The second approach, to decrease the step size, α_{id}, was helpful in terms of reducing the magnitude of the end point acceleration, but as was learned in the preliminary experiments, smaller step sizes took longer to train and lead to ultimately larger errors in trajectory following (assuming the use of a boundary layer). Still more fine tuning was required though, as more two module experiments displayed difficulties in discriminating between the two contexts. Figure 7-16 a. is a representation of the error step size as it was initially used. The jump in step size at the tolerance value is due to the boundary layer, effectively zeroing the step size when the error was below tolerance. Figures 7-16 b. and c. show two variations that were investigated to "shape" the step size. These shaped step sizes were an attempt to more gently train the inverse models as the error rose above tolerance. Using a tolerance of $4mm$ and a thickness, α_δ of $4mm$, the results of using these three different steps were compared. Making reaching movements within the unperturbed context the controller trained on targets randomly generated eight at a time. The results of every fourth set of eight targets were recorded. A plot of the RMS acceleration of the x and y components of the end point is shown in figures 7-17 and 7-18. These plots were made to observe how the different step shapes changed the endpoint acceleration. Based on the general effect the step sizes had on the endpoint's acceleration the third step shape, 7-16 C. was used for the remainder of the work.

Tasks one and two were completed with this new shaped step for the inverse dynamic models. The resulting endpoint paths for the two tasks can be seen in figures 7-19 and 7-20. Again, training was accomplished by generating sets of eight targets in a random order. As before, the fourth set of eight targets were recorded. For both tasks the step sizes were set to $\alpha_{id} = .01$ and $\alpha_{fd} = .2$ with tolerances of $4mm$ and $0.2m/s^2$ for the inverse and forward dynamic models. Training took place

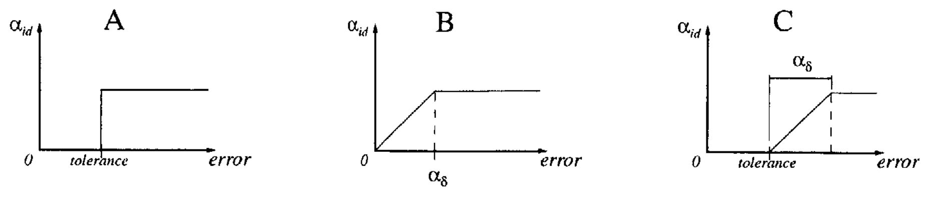

Figure 7-16: Diagram of shaped steps used for training the inverse dynamic models: A, original step, B, step size without a no training zone, C, step size with a no training zone and shaped step.

at $5ms$ intervals. Both tasks were trained for approximately 650 reaching movements. The RMS position error for task one was $4.56mm$ ($3.68mm$ and $2.70mm$ in x and y respectively). For task two the RMS position error was $6.36mm$ ($5.07mm$ and $2.81mm$ in x and y).

7.5.2 Task 3

To complete task three, the trained module from task one, the unperturbed context, was incorporated with another empty module. Several initial conditions would need to be established before task three was under way. Although it was not needed for the single behavior experiments, tasks one and two, the variance σ^2, equation 5.6, for both modules was computed at each training cycle. It was defined to be a filtered version of one half the mean squared error in the forward estimate $(\frac{1}{2}\|\ddot{x} - \hat{\ddot{x}}\|^2)$. That value for the variance of the unperturbed context, $0.30m^2/s^4$, was now used to initialize module one. The variance for module two, was set to a large value, $10m^2/s^4$, to reflect the fact that because it represented no behavior as of yet, its uncertainty in any estimate was relatively large.

The responsibilities would have to be initialized as well. Because task three is designed to test the control architecture's ability to autonomously perform in unknown contexts the responsibility signal was set to .9 and .1 for modules one and two respectively. This effectively initializes the system as if it had previously been making unperturbed reaching movements and was suddenly introduced into the curl field.

To filter the responsibility and variance signals appropriate time constants would

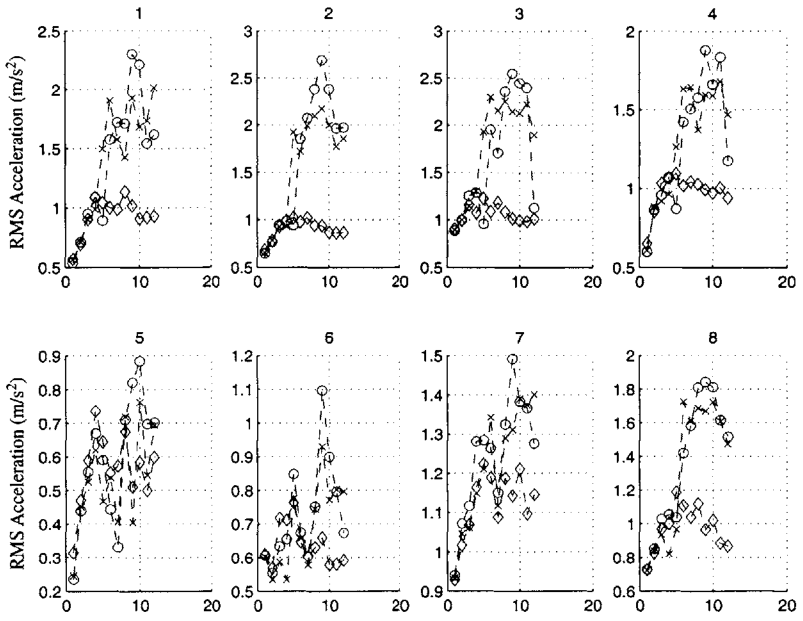

Figure 7-17: The x component of RMS acceleration of the endpoint for various shaped steps. X's, circles, and diamonds denote steps A, B, and C, respectively (see figure 7-16).

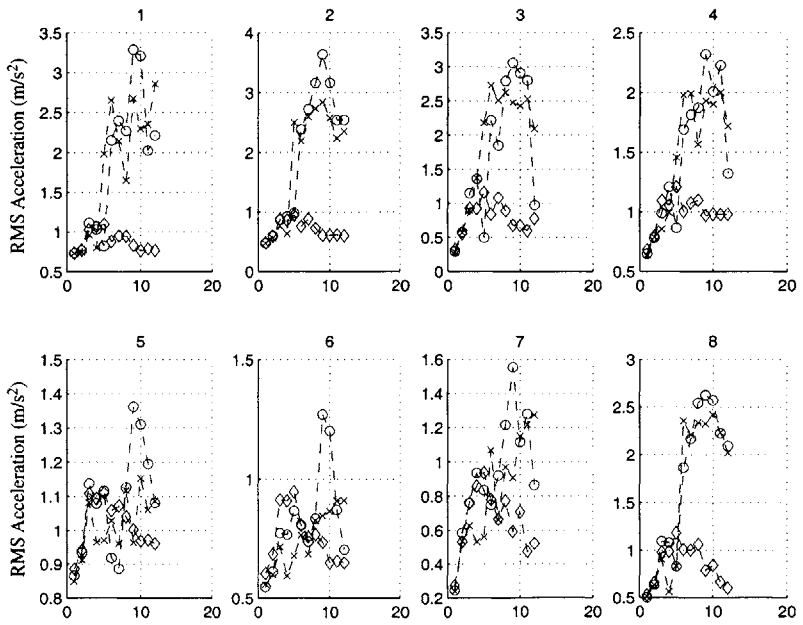

Figure 7-18: The y component of RMS acceleration of the endpoint for various shaped steps. X's, circles, and diamonds denote steps A, B, and C, respectively (see figure 7-16).

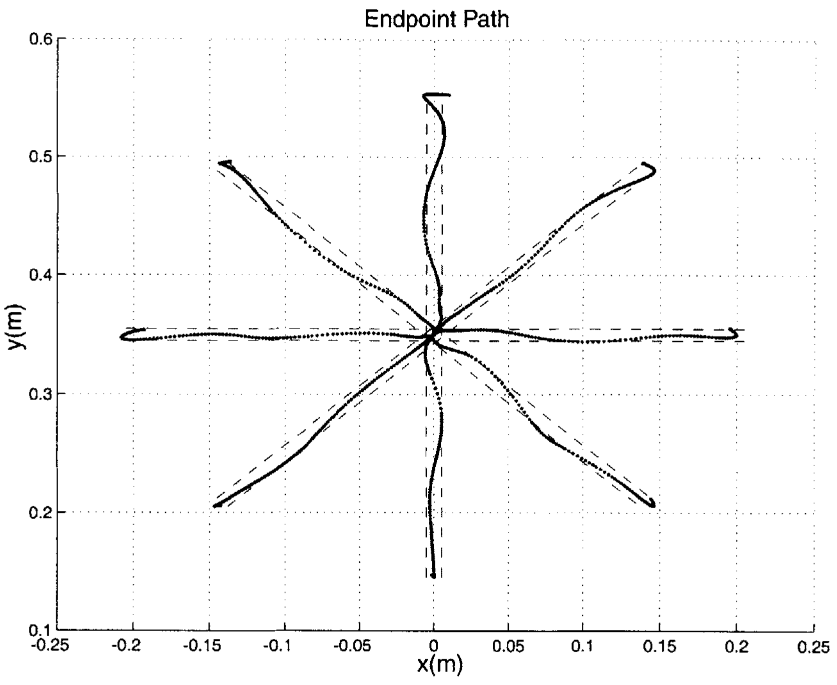

Figure 7-19: End point path for task1. RMS position error is $4.56mm$. Dotted lines denote the $4mm$ tolerance.

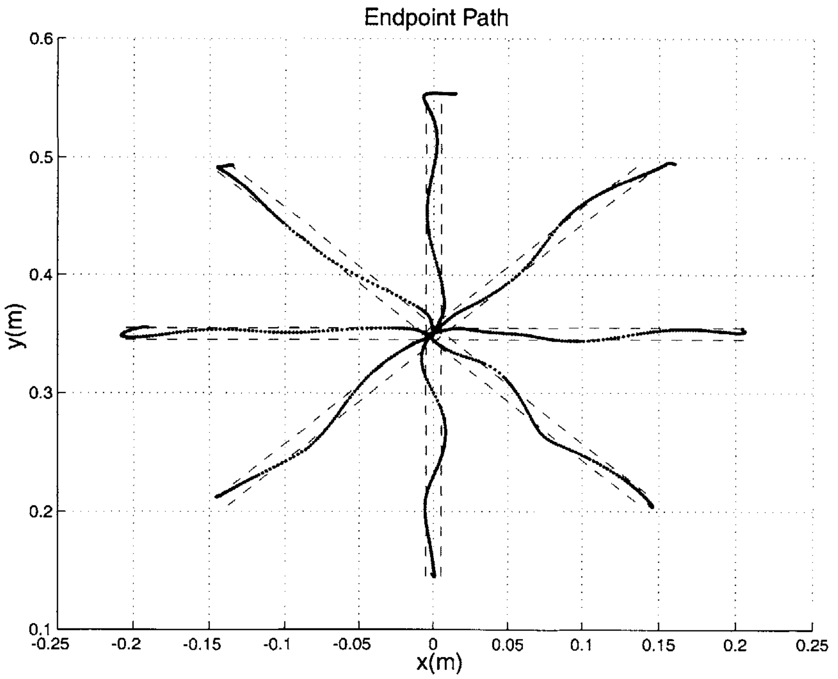

Figure 7-20: End point path for task2. RMS position error is $6.36mm$ Dotted lines denote the $4mm$ tolerance.

have to be determined. During the tasks the context would change abruptly, instantaneously in fact, so the responsibility signal should change relatively fast. On the other hand, if the signal changed too quickly, as preliminary tests showed, the responsibility bounced back and forth, favoring one module over another according to the forward model estimates for that time step. A more judicious signal would be representative of the forward estimates made over a larger time scale than a few update cycles. The time constant for the responsibility, τ_λ, was chosen to be 2 seconds, on the order of two moves. Unlike the time required to change contexts, learning was a slow process so the time constant for the variance, τ_{σ^2}, would have to reflect that. Considering tasks one and two were completed in 650 movements, approximately 780 seconds, τ_{σ^2} was chosen to be 100 seconds.

Training took place just as with tasks one and two; the results of every fourth set of eight randomly generated targets were recorded. Initially the responsibility signal grew to favor module two, but as time progressed the responsibility signal approached an even split between the two modules. Figure 7-21 is a time history of the mean responsibility signals for task three during each of the fourth set of eight targets.

Training was halted after approximately 500 movements. With a responsibility signal of .55 and .45, for modules one and two, the final RMS position error was $7.60mm$ ($6.76mm$ and $3.48mm$ in x and y). It is likely that more training would reduce the error in trajectory following, but task three was designed to investigate the controllers ability to incorporate multiple behaviors. The responsibility signals tending towards .5 as they did, suggested that this was not working. To determine how much of the previous behavior was retained in module one, and how much of the new behavior from the current task had been incorporated into module two, the modules were individually validated. Module one was used to control the arm in context one, and module two was used to control the arm in context two. Understandably so, with both responsibilities near .5, neither module retained a complete behavior. Module one's error had degraded to an RMS position error of $15.9mm$ in x and $9.56mm$ in y. Note also how training within the curl field of context two had caused the controller to move the arm in a direction opposite that of the curl (figure 7-22). Module two,

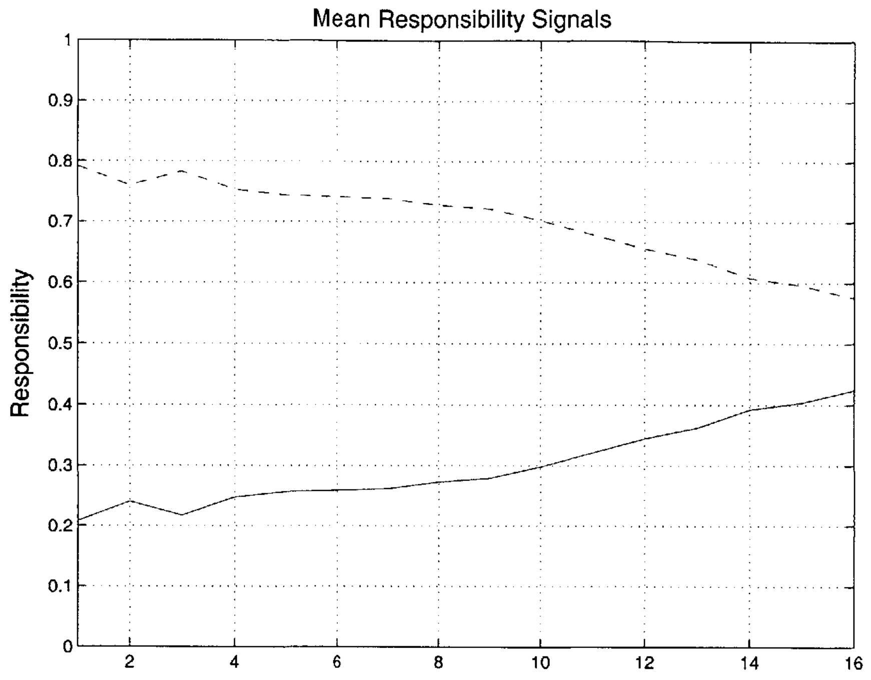

Figure 7-21: Mean responsibility signals for task three. λ_1 is denoted with a solid line, λ_2 with a dotted line.

though now capable of better performance than an untrained controller (7-4), with an RMS error of $7.87mm$ in x and $5.08mm$ in y (figure 7-23), was still less capable than the module of task two, and further training would be of little help for the responsibility signal would not favor it's training over that of module one.

7.5.3 Task 4

Task four, designed to investigate the controllers ability to generalize previously learned behaviors when interacting in new contexts, met with better success. Using the modules from task one and two, the responsibility was initialized to .9 and .1 as if the controller had just previously been making unperturbed reaching movements. τ_{σ^2} and τ_λ were set to the same values as before. The variances were initialized to the values obtained in tasks one and two, $\sigma_1^2 = 0.30m^2/s^4$ and $\sigma_2^2 = 0.72m^2/s^4$. The

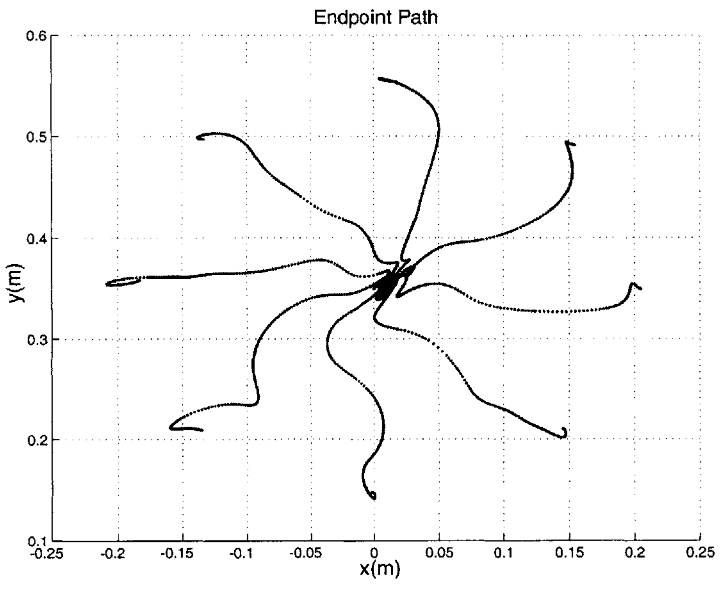

Figure 7-22: Validation of task three. Endpoint path in context one, using module one.

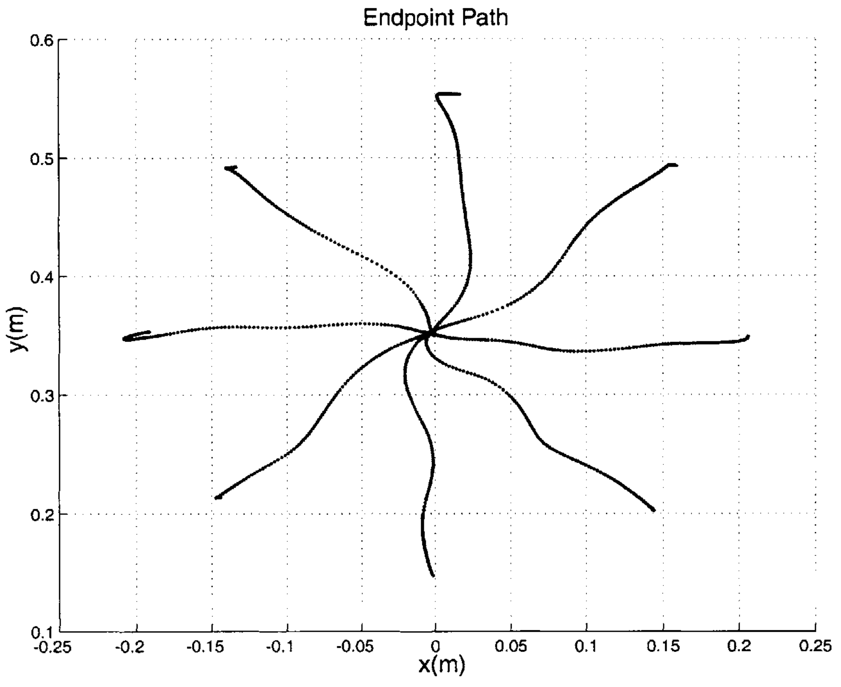

Figure 7-23: Validation of task three. Endpoint path in context two, using module two.

curl field strength $\boldsymbol{B}_c$ was halved

$$\boldsymbol{B}_c = \begin{bmatrix} 0 & -20 \\ 20 & 0 \end{bmatrix} \tag{7.8}$$

By decreasing the curl field strength and equipping the controller with trained modules from tasks one and two, task four would test if the controller could utilize the modules to make the desired minimum jerk movements with little training, and without losing much of the previously trained behavior.

Once the task began the position error immediately dropped to low values. After approximately 200 movements (with randomly generated targets) training was halted. A time history of the responsibility signal is shown in figure 7-24. The RMS position error was $5.15mm$ and $2.92mm$ in x and y. Validating the modules, we see that little of the previously learned behaviors was lost. Figure 7-25 shows the endpoint path when module one was validated in context one. The RMS position error was $5.28mm$ ($4.59mm$ and $2.62mm$ in x and y), compared with the $4.56mm$ of task 1, a difference in performance of 3.6% of the total reach length, evidence that module one remained essentially intact. Similarly when module two was validated in context two (the full strength curl field) the RMS position error was $6.71mm$ ($6.45mm$ and $3.45mm$ in x and y, figure 7-26), comparable to the RMS error of task two $6.36mm$, a difference of 1.8% of the reach length.

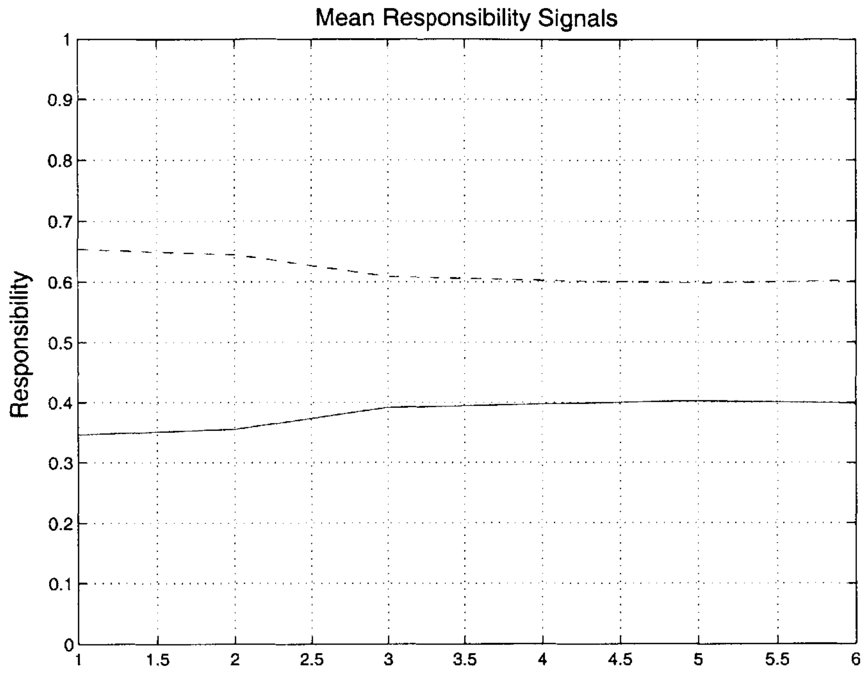

Figure 7-24: Mean responsibility signals for task three. λ_1 is denoted with a solid line, λ_2 with a dotted line.

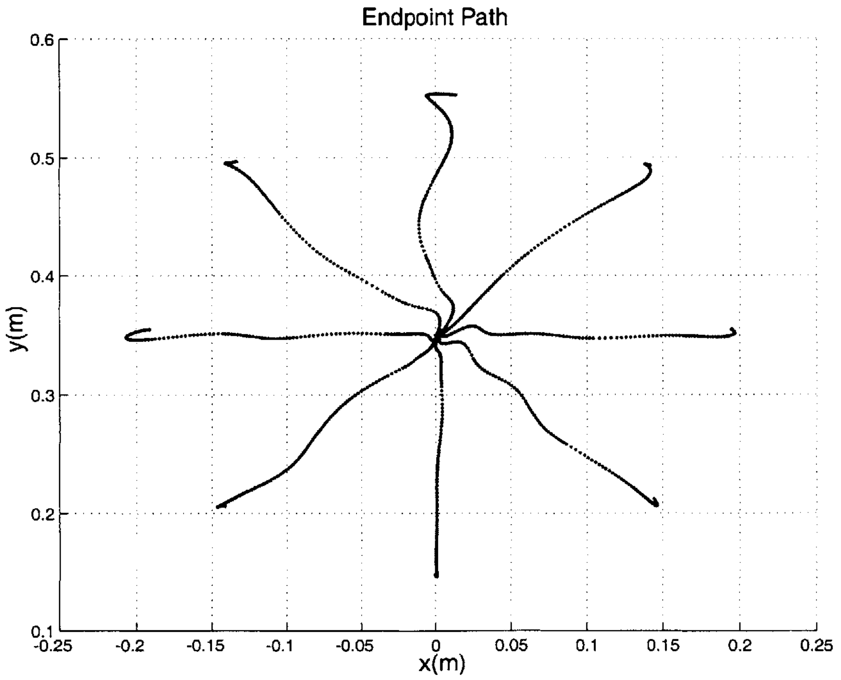

Figure 7-25: Validation of task four. Endpoint path in context one using module one.

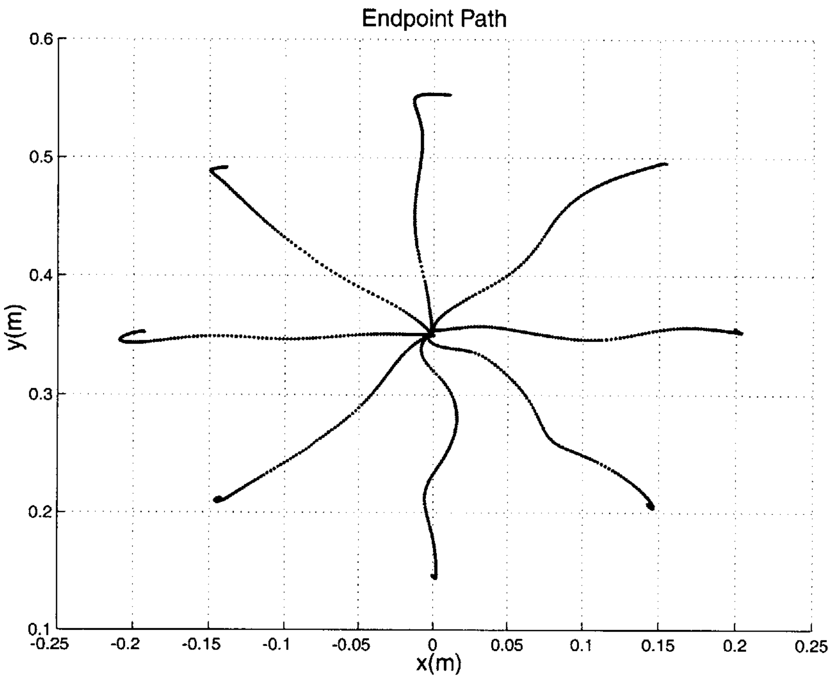

Figure 7-26: Validation of task four. Endpoint path in context two, using module two.

Chapter 8

Discussion

Task one was able to achieve $4.6mm$ in RMS error from the desired minimum jerk trajectory after 650 training trials. A similar experiment was made with human subjects. Participants were asked to make point-to-point, center-out movements, to randomly generated targets. The geometry of the human subject's targets was identical to that of this current experiment. The RMS error from a straight line path for the human subjects was found to be $1.6cm$. Hence, the controller can achieve better performance, in terms of trajectory following, than the human subjects.

Similar results were found for task two. The same experiment referred to above, consisted of subjects making point-to-point reaching movements in a curl field (the strength of which was, $B_c = 12N \cdot s/m$). The participants revealed an RMS error in the curl field of $1.7cm$. Again, this confirms that the controller, with $6.4mm$ trajectory following error, compared well.

Task three, although achieving $7.6mm$ in RMS error from a desired trajectory, was not able to achieve the ultimate goal of learning a second behavior while retaining the first behavior intact. In this sense it failed to meet the desired criteria for a successful completion of task three.

The results of task four proved more favorable. The controller was capable of generalizing knowledge of the previously learned behaviors, to a new unknown context, with little adaptation and little degradation of the previously learned behaviors.

The major shortcoming of the experiment is the failure of task three. The novelty of the controller lies in its attempts to incorporate multiple learned behaviors. Yet, during the completion of task three, it became apparent that the current implementation of this thesis' controller was not retaining the two behaviors of contexts one and two separately. In the process of learning the motor behavior for context two, the previously trained behavior for context one was being lost. Realizing that this was in great part due to the calculation of the responsibility signal and it's reliance on the forward dynamic model's estimates, two questions arose. Just how different are the two contexts? Are the two contexts similar enough that, due to inherent inaccuracies in the network models, the controller's ability to distinguish between them was hindered? And, does the control architecture have a stable equilibrium configuration that allows for multiple motor behaviors? The first question is specific to the implementation in this work and would not necessarily have a deep impact on the capabilities of the control architecture as a paradigm for motor control and motor learning in general. The second question, more fundamental to the controller, addresses the stability of the architecture, in terms of learned models and context dependent responsibility signals, while encompassing two or more distinct behaviors. The answer to this question would certainly have very strong implications for future use of this architecture.

To address the first question, the two contexts would have to be compared in terms of the controller's switching mechanism. A plot of the endpoint acceleration as the arm makes center-out reaching movements to each of the eight targets, in both contexts, is shown in figures 8-1 and 8-2. These plots were made using a minimum jerk trajectory, without the aid of an inverse dynamic controller. The responsibility signal is effectively a measure of which forward model has the least RMS error in the endpoint's acceleration. Knowing this, a few idealizations will be made. The first forward model, for task one, will be assumed perfect, that is it exactly matches the actual forward dynamic equations for context one. Further assume that the second forward model, for task two, is also perfect. That done, what will be referred to as the ideal responsibility update, $\hat{\lambda}$, is computed (equation 5.4). These are the values the responsibilities will converge to given enough time. Suppose the controller

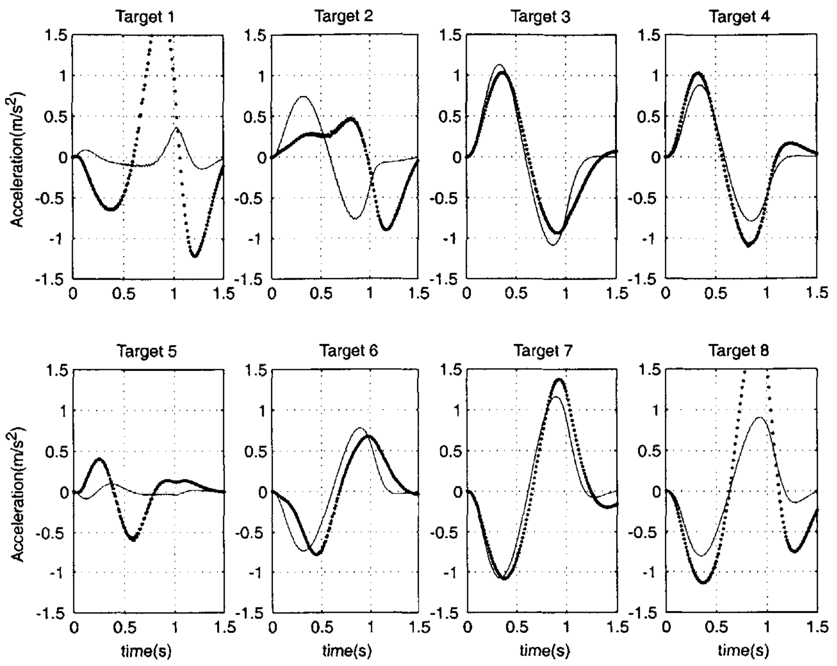

Figure 8-1: Endpoint acceleration, x component. Solid lines represent interaction in context one, dotted lines are context two.

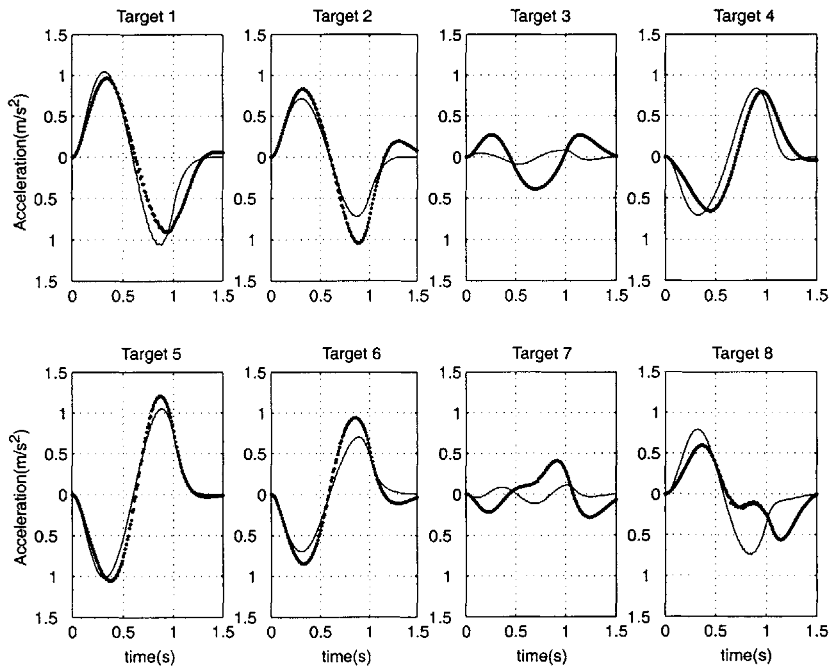

Figure 8-2: Endpoint acceleration, y component. Solid lines represent interaction in context one, dotted lines are context two.

is acting in context two, the curl field. The terms $e^{(-(\|\ddot{\boldsymbol{x}}-\hat{\ddot{\boldsymbol{x}}}_2\|)/\sigma_2^2)}$ in the soft max calculation will be 1, the difference between $\ddot{\boldsymbol{x}}$ and $\ddot{\boldsymbol{x}}_2$ being 0. To calculate the remaining exponential terms involving $\hat{\ddot{\boldsymbol{x}}}_1$ we substitute the numerator $\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_1\|$ with the RMS difference between the endpoint's acceleration for contexts one and two (figures 8-1 and 8-2), and use the variance found in task one, $\sigma_1^2 = 0.3m^2/s^4$. The resulting ideal responsibility updates are $\hat{\lambda}_1 = 0.315$ and $\hat{\lambda}_2 = 0.685$, or 31% of module one and 69% of module two. This represents the best responsibility signal possible, when both forward models are perfect in their estimates and the first model has the variance given above. If the forward models are perfect, then the variance for both will tend to zero according to equation 5.6, but the time constants for the variance calculation are two orders of magnitude slower than the responsibility calculation, so for the purposes of discussion this idealization will suffice. What this demonstrates is something that has previously been addressed several times. The forward models need accurate and precise estimates in order to properly distinguish between the contexts and allow the controller to parse several motor behaviors. With greater accuracy in the forward models, individual variances would shrink, and the distinction between two modules would be more pronounced, driving the responsibilities further apart, towards 1 and 0. In fact, inverting this responsibility update calculation, we see that in order to compute an ideal responsibility update signal of $\hat{\lambda}_1 = 0.1$ and $\hat{\lambda}_2 = 0.9$, values that would likely represent two fairly distinct behaviors, would require σ_1^2 to be $0.11m^2/s^4$. This value is below those actually achieved, $0.30m^2/s^4$ and $0.72m^2/s^4$ for tasks one and two respectively.

So it would seem that under ideal conditions (although not the conditions achieved here), the responsibility signals can obtain values that would enhance the controller's ability to partition motor behaviors between varying contexts. This being the case, another attempt at task three was made. This time, in contrast to the original task three, the controller was given a "nudge" in the right direction, initializing the responsibilities to favor the untrained module, instead of the previously trained unperturbed module. The responsibilities were initialized to $\lambda = [.01, .99]$. This large responsibility for module two, in addition to it's larger initial variance ($\sigma^2 = [.30, 10]$),

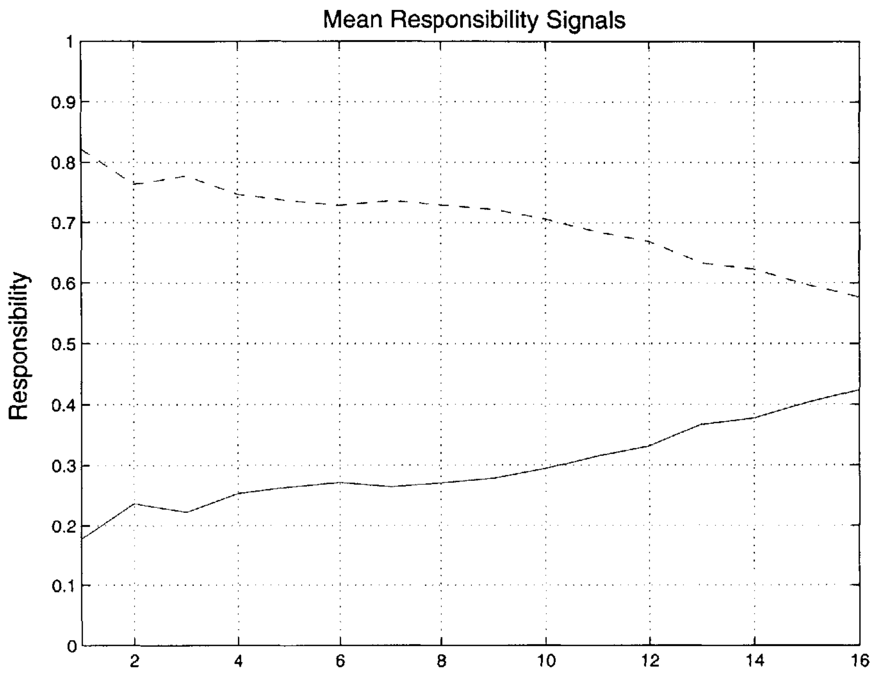

Figure 8-3: Mean responsibility signal during new task 3.λ_1 is denoted with a solid line, λ_2 with a dotted line.

would give module two a further advantage in learning the new unknown context. Interestingly, the results were not much different from those of the original task three. In figure 8-3 we see a plot of the mean responsibility signals over 512 randomly ordered reaching movements, calculated once every 24 moves, just as before. Comparing these results with the previous attempt at task three, figure 7-21, we see there was little, if any, improvement in partitioning the modules. In fact, the responsibility signals were remarkably similar. It seems as though the initial conditions of λ have little effect on the ultimate outcome of the task.

This leads us to the second question: is there a stable configuration for the controller in which it has retained multiple behaviors? Such a stable configuration would require the responsibility signals to settle to values that were close to one and zero. Yet, the multiple module tasks, both the original task three and the second attempt just described, did not exhibit this characteristic. What was observed, was a seemingly stable tendency towards responsibilities of 0.5. To answer the question of stability, the pertinent equations need to be arranged in a form amenable for stability analysis. As implemented in this thesis, all the variables were adapted intermittently, but they can easily be represented in their continuous form, an ensemble of coupled differential equations. For the purposes of the following discussion, they have all been reproduced here:

the forward dynamics:

$$\ddot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}) \tag{8.1}$$

the motor command:

$$\boldsymbol{u} = \boldsymbol{u}_k + \lambda_1 \boldsymbol{u}_1 + \lambda_2 \boldsymbol{u}_2 \tag{8.2}$$

the individual module motor commands:

$$\begin{aligned} \boldsymbol{u}_1 &= \psi(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{x}_d, \boldsymbol{w}_1) \\ \boldsymbol{u}_2 &= \psi(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{x}_d, \boldsymbol{w}_2) \end{aligned} \tag{8.3}$$

the individual module inverse model learning rules:

$$\begin{aligned} \dot{\boldsymbol{w}}_1 &= \lambda_1 (\partial\psi/\partial\boldsymbol{w}_1)^T (\phi_k(\boldsymbol{x}_d) - \phi_k(\boldsymbol{x})) \\ \dot{\boldsymbol{w}}_2 &= \lambda_2 (\partial\psi/\partial\boldsymbol{w}_2)^T (\phi_k(\boldsymbol{x}_d) - \phi_k(\boldsymbol{x})) \end{aligned} \tag{8.4}$$

the forward estimate:

$$\hat{\ddot{\boldsymbol{x}}} = \lambda_1 \hat{\ddot{\boldsymbol{x}}}_1 + \lambda_2 \hat{\ddot{\boldsymbol{x}}}_2 \tag{8.5}$$

the individual module forward estimates:

$$\begin{aligned} \hat{\ddot{\boldsymbol{x}}}_1 &= \phi(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{v}_1) \\ \hat{\ddot{\boldsymbol{x}}}_2 &= \phi(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{v}_2) \end{aligned} \tag{8.6}$$

the individual module forward model learning rules:

$$\begin{aligned} \dot{\boldsymbol{v}}_1 &= \lambda_1 (\partial\phi/\partial\boldsymbol{v}_1)^T (\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}) \\ \dot{\boldsymbol{v}}_2 &= \lambda_2 (\partial\phi/\partial\boldsymbol{v}_2)^T (\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}) \end{aligned} \tag{8.7}$$

the individual module responsibilities:

$$\dot{\lambda}_1 = \frac{1}{\tau_\lambda}\left[-\lambda_1 + \frac{\exp(-\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_1\|^2/\sigma_1^2)}{\exp(-\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_1\|^2/\sigma_1^2) + \exp(-\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_2\|^2/\sigma_2^2)}\right]$$

$$\dot{\lambda}_2 = \frac{1}{\tau_\lambda}\left[-\lambda_2 + \frac{\exp(-\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_2\|^2/\sigma_2^2)}{\exp(-\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_1\|^2/\sigma_1^2) + \exp(-\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_2\|^2/\sigma_2^2)}\right] \qquad (8.8)$$

the individual module variances:

$$\dot{\sigma}_1^2 = \frac{\lambda_1}{\tau_{\sigma^2}}\left[-\sigma_1^2 + \|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_1\|^2/2\right]$$

$$\dot{\sigma}_2^2 = \frac{\lambda_2}{\tau_{\sigma^2}}\left[-\sigma_2^2 + \|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_2\|^2/2\right] \qquad (8.9)$$

Though equations 8.1 to 8.9 are all used in the control architecture[1] several of them can be subsumed to compactly represent the *state* equations of the controller. The inverse dynamic model outputs (equations 8.3) can be combined in the motor command (equation 8.2), and placed in the arm's forward dynamics (equation 8.1). Similarly, the forward dynamic model estimates (equations 8.6) can be combined in the controller's forward estimates (equation 8.5), and used to define the forward model's learning rule (equations 8.7). The remaining time rate equations, 8.1, 8.4, 8.7, 8.8, and 8.9 dictate evolution of the state,

$$\mathcal{X} = \left[\boldsymbol{x}^T, \dot{\boldsymbol{x}}^T, \boldsymbol{w}_1{}^T, \boldsymbol{w}_2{}^T, \boldsymbol{v}_1{}^T, \boldsymbol{v}_2{}^T, \lambda_1, \lambda_2, \sigma_1^2, \sigma_2^2\right]^T$$

such that,

$$\dot{\mathcal{X}} = \boldsymbol{F}(\mathcal{X})$$

An equilibrium state, $\mathcal{X}_o$, is defined to be any state such that $\boldsymbol{F}(\mathcal{X}_o) = 0$. Setting all the rate equations to zero, we find that equilibria is satisfied when, $\ddot{\boldsymbol{x}} = 0$, $\hat{\ddot{\boldsymbol{x}}} = \ddot{\boldsymbol{x}}$, $\phi_k(\boldsymbol{x}_d) = \phi_k(\boldsymbol{x})$, and λ_1 and λ_2 are equal to their update values, the soft max calculation of equations 8.8. Note how this admits many solutions. Assume an optimum weight vector $\boldsymbol{v}^*$ exists, such that

$$\ddot{\boldsymbol{x}} = \hat{\ddot{\boldsymbol{x}}} = \phi(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{v}^*)$$

[1]The equations representing the learning rules for the kinematic module have been omitted, for they were not in use during the tasks.

Because the network models are linear, the above can be satisfied with any combination of $\lambda_1\hat{\ddot{\boldsymbol{x}}}_1 + \lambda_2\hat{\ddot{\boldsymbol{x}}}_2$, such that $\lambda_1\boldsymbol{v}_1 + \lambda_2\boldsymbol{v}_2 = \boldsymbol{v}^*$ [2], provided the responsibilities equal their update values. These criteria are met by multiple solutions, but to establish that the controller can retain multiple behaviors, in the form of distinct weight sets, the stability analysis should not solely be concerned with equilibrium of the entire state. For the controller to demonstrate the ability to utilize these distinct weight sets in a stable fashion, it would have to do so, not while the physical system was at rest, but while it was active and interacting within a context. As such, our stability analysis should concentrate on those equilibrium conditions that provide for stable sets of weights, regardless of the state of the physical system ($[\boldsymbol{x}^T, \dot{\boldsymbol{x}}^T]$).

There is a further point of clarification. Although the inverse dynamic models do indirectly affect the forward models, by driving the forward dynamics of the physical system, they have no direct affect on the responsibility signal. In fact, once the forward dynamic models and responsibility signals have come to equilibrium, the inverse dynamic models are decoupled from them, and can settle to their own equilibrium without perturbing the controller[3]. Taking into account these new qualifications, the equilibrium condition of concern is when, $\lambda_1\boldsymbol{v}_1 + \lambda_2\boldsymbol{v}_2 = \boldsymbol{v}^*$, and λ_1 and λ_2 are equal to their update values, e.g. for λ_1,

$$soft\ max = \frac{\exp(-\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_1\|^2/\sigma_1^2)}{\exp(-\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_1\|^2/\sigma_1^2) + \exp(-\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}_2\|^2/\sigma_2^2)} \tag{8.10}$$

To continue on with the analysis, these equilibrium conditions will have to be identified and characterized for stability. To determine if these equilibria are stable, Lyapunov's indirect method shall be used. The rate equations must be linearized about the equilibrium states in question and the resulting eigenvalues inspected. If the linearized system is stable (all real eigenvalues lie strictly on the left half of the jw axis), then the nonlinear system is stable about that equilibrium point. If the linearized system is unstable (at least one eigenvalue has a real part in the right half

[2]Adding a tolerance for training, a boundary layer, allows for inaccuracies in the forward estimates, such that $\|\ddot{\boldsymbol{x}} - \hat{\ddot{\boldsymbol{x}}}\| < \epsilon$, essentially conceding that $\boldsymbol{v}^*$ need not exist, or be achieved.

[3]This is true assuming the inverse dynamic models do not drive the forward dynamics of the physical system outside a region the forward dynamic models have already appropriately learned to estimate

plane), then the nonlinear system is unstable about that equilibrium point. And if the linearized system is marginally stable (all eigenvalues lie in the left half plane, but at least one lies on the jw axis), then no conclusions can be drawn about the stability of the nonlinear system.

To proceed any further with the stability analysis of the system becomes quite difficult. First, equation 8.1 represents a number of transformations from joint coordinates to endpoint coordinates (equations 4.1, 4.2 and 4.3) that make a closed form solution, and subsequent linearization extremely difficult. Second, though the equilibrium conditions have been defined above implicitly, explicit knowledge of $\boldsymbol{v}^*$ is unknown. Therefore, a simplified version of our current control system shall be analyzed. The physical dynamics of the arm shall be reduced to a second order system that ensures an exact forward estimate. To do this, it will be composed of Gaussians as follows,

$$\ddot{x} = f(x, \dot{x}) = e^{-(x-1)^2} - e^{-(x+1)^2} + e^{-(\dot{x}-1)^2} - e^{-(\dot{x}+1)^2} \tag{8.11}$$

As can be observed, the motor commands will be neglected altogether, somewhat analogous to the uncoupled motor commands of the actual system while the controller is in equilibrium. The above equation has multiple equilibrium states, corresponding to the system at rest, and values sufficiently removed from the centers of the Gaussians such that $f(x, \dot{x}) = 0$ (actually, x and $\dot{x}$ must tend to ∞). To counter this unusual circumstance, the analysis will be confined to a region (between the centers of the Gaussians) containing a unique stable equilibrium point, $\{x = 0; \dot{x} = 0\}$. The simplified forward models shall be similarly constructed,

$$\begin{aligned}
\hat{\ddot{x}}_1 &= \phi(x, \dot{x}, \boldsymbol{v}_1) = v_{11}e^{-(x-1)^2} + v_{12}e^{-(x+1)^2} + v_{13}e^{-(\dot{x}-1)^2} + v_{14}e^{-(\dot{x}+1)^2} \\
\hat{\ddot{x}}_2 &= \phi(x, \dot{x}, \boldsymbol{v}_2) = v_{21}e^{-(x-1)^2} + v_{22}e^{-(x+1)^2} + v_{23}e^{-(\dot{x}-1)^2} + v_{24}e^{-(\dot{x}+1)^2}
\end{aligned} \tag{8.12}$$

Note that now $\boldsymbol{v}^*$ exists and is known.

The responsibility signal shall be computed just as in equation 8.8, only now the variance shall be omitted (this will only alter the rate at which the responsibility changes). The new simplified, twelve element state, $\mathcal{X}$, is,

$$\mathcal{X} = \left[x, \dot{x}, \boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \lambda_1, \lambda_2\right]^T \tag{8.13}$$

representing the physical system, the weights, and the responsibilities. The complete simplified state equations are as follows,

$$\begin{aligned}
\ddot{x} &= e^{-(x-1)^2} - e^{-(x+1)^2} + e^{-(\dot{x}-1)^2} - e^{-(\dot{x}+1)^2} && (8.14)\\
\dot{\boldsymbol{v}}_1 &= \lambda_1(\partial\phi/\partial\boldsymbol{v})^T(\ddot{x} - \hat{\ddot{x}}) && (8.15)\\
\dot{\boldsymbol{v}}_2 &= \lambda_2(\partial\phi/\partial\boldsymbol{v})^T(\ddot{x} - \hat{\ddot{x}}) && (8.16)\\
\dot{\lambda}_1 &= \frac{1}{\tau_\lambda}\left[-\lambda_1 + \frac{e^{(-(\ddot{x}-\hat{\ddot{x}}_1)^2)}}{e^{(-(\ddot{x}-\hat{\ddot{x}}_1)^2)} + e^{(-\ddot{x}-\hat{\ddot{x}}_2)^2)}}\right] && (8.17)\\
\dot{\lambda}_2 &= \frac{1}{\tau_\lambda}\left[-\lambda_2 + \frac{e^{(-(\ddot{x}-\hat{\ddot{x}}_2)^2)}}{e^{(-(\ddot{x}-\hat{\ddot{x}}_1)^2)} + e^{(-\ddot{x}-\hat{\ddot{x}}_2)^2)}}\right] && (8.18)
\end{aligned}$$

With this new system, the equilibria states of interest are defined by, $\lambda_1\boldsymbol{v}_1 + \lambda_2\boldsymbol{v}_2 = \boldsymbol{v}^*$, where $\boldsymbol{v}^* = [-1, 1, -1, 1]^T$, and λ_1 and λ_2 are equal to their update values,

$$soft\ max = \frac{e^{(-(\ddot{x}-\hat{\ddot{x}}_i)^2)}}{e^{(-(\ddot{x}-\hat{\ddot{x}}_i)^2)} + e^{(-\ddot{x}-\hat{\ddot{x}}_j)^2)}} \tag{8.19}$$

At first glance it might appear that these criteria can be met by multiple solutions, but recall that for equilibrium, these criteria must hold regardless of the physical state, $[x, \dot{x}]^T$. Yet the update values, from equation 8.19, are inherently a function of the physical state. To help explain this, a plot of the responsibility satisfying the equilibrium conditions for varying x has been made ($\dot{x}$ was held constant at 0). The plot, figure 8-4, was made by choosing values for $\boldsymbol{v}_1$ and x, and numerically solving for λ_1 and λ_2 such that $\lambda_1\boldsymbol{v}_1 + \lambda_2\boldsymbol{v}_2 = \boldsymbol{v}^*$, and both responsibilities equaled their update values. The responsibility for module one is shown.

As can be seen from the plot, all weight vectors correctly estimate the forward state, $\ddot{x}$, when the system is at rest. This is due to the choice of weight vectors, symmetric about zero. Yet, the only combination of weight vectors and responsibilities that allows for an equilibrium condition, regardless of the physical state, is when both weight vectors are equal to the optimum weight vector (in this case the actual weight vector), $\boldsymbol{v}_1 = \boldsymbol{v}_2 = \boldsymbol{v}^*$, and both responsibilities are one half, $\lambda_1 = \lambda_2 = .5$. For all

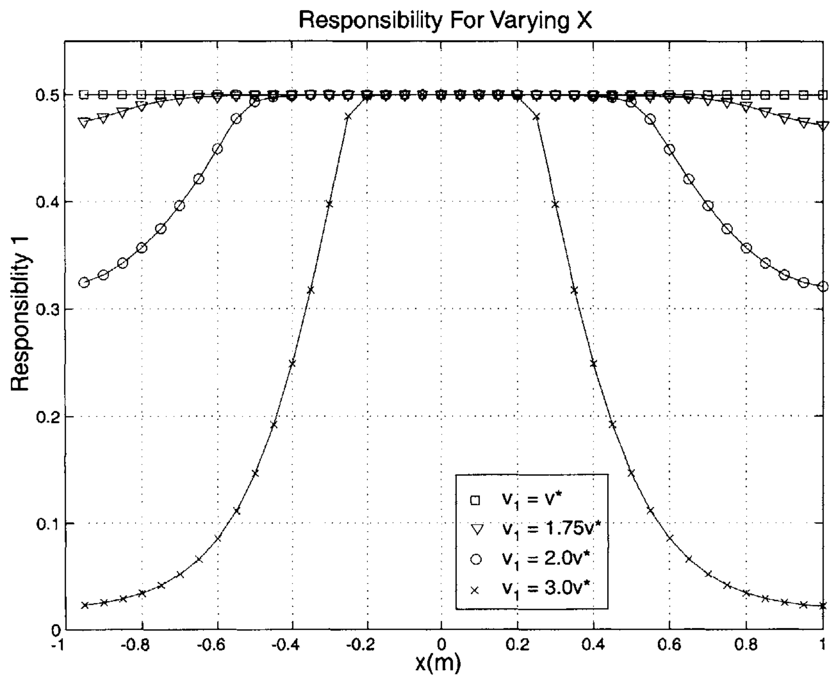

Figure 8-4: Varying responsibility for varying x.

other configurations the responsibility varies with the physical state and is therefore not in equilibrium.

Mathematically this is confirmed by finding the conditions necessary for which the responsibility is constant with respect to x and $\dot{x}$. By taking the partial derivative of equation 8.19 with respect to x and setting the result to zero, the only solution, for varying x, is when $\boldsymbol{v}_1 = \boldsymbol{v}_2 = \boldsymbol{v}^*$. Taking the partial of equation 8.19 with respect to $\dot{x}$ the same result is obtained (see Appendix C).

What has been demonstrated, is that the *only* equilibrium state for the controller (not the complete state of the system), is when all the forward dynamic weight vectors are equal to the optimum weight vector, and the responsibility is split evenly amongst the modules. Although the tasks of the experiment seem to suggest this is a stable equilibrium configuration, the question of stability still remains. Therefore, the equilibrium state, $\mathcal{X}_o$, shall be defined as,

$$\mathcal{X}_o = [0, 0, (-1, 1, -1, 1), (-1, 1, -1, 1), .5, .5]^T \tag{8.20}$$

At this state, the system is in equilibrium; the physical system is at rest and no model will train, for both produce exact estimates of the forward acceleration. The next step is to linearize the state equations about this equilibrium state and consider the stability of the linearized system.

Regrettably, linearizing the system about this equilibrium state, $\mathcal{X}_o$, zero eigenvalues are obtained (see Appendix C). As such, no conclusions can be drawn yet. Inspecting the linearized system matrix, $\boldsymbol{A}$, reveals a decoupling of the state. The $\boldsymbol{A}$ matrix is composed of smaller matrices representing the physical state variables, the weights and the responsibilities. $\boldsymbol{A}$ is of the form,

$$\boldsymbol{A} = \begin{bmatrix} [A]_{\ddot{x}} & 0 & 0 \\ 0 & [A]_v & 0 \\ 0 & 0 & [A]_\lambda \end{bmatrix} \tag{8.21}$$

This reveals that, locally about equilibrium, there is no dynamic interaction between the physical states, the weights and the responsibilities. Moreover, the eigenvalues and the eigenvectors, V, are also decoupled, dependent only on the parameters of these distinct groups of state variables.

$$eigen(\boldsymbol{A}) = eigen([A]_{\ddot{x}}),\ eigen([A]_{\boldsymbol{v}}),\ eigen([A]_\lambda) \tag{8.22}$$

$$[V] = \begin{bmatrix} v_{1,1} \\ v_{1,2} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} v_{1,2} \\ v_{2,2} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ v_{i,3} \\ \vdots \\ 0 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ \vdots \\ v_{i,10} \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ v_{11,11} \\ v_{12,11} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ v_{11,12} \\ v_{12,12} \end{bmatrix} \tag{8.23}$$

The importance of this lies in the interpretation of eigenvalues and eigenvectors. Any trajectory for this system, represented in state space, follows a path whose shape, or mode, is composed of the eigenvectors. The rate at which these modes travel is a function of the eigenvalue associated with a particular vector. The zero eigenvalues of the $\boldsymbol{A}$ matrices originate in the weight matrices, $[A]_{\boldsymbol{v}}$, all other eigenvalues being strictly stable. This means that the physical states, with negative eigenvalues, are stable and uncoupled from the rest of the state and tend toward equilibrium (as is witnessed by the lack of other state terms in equation 8.11). The responsibilities,

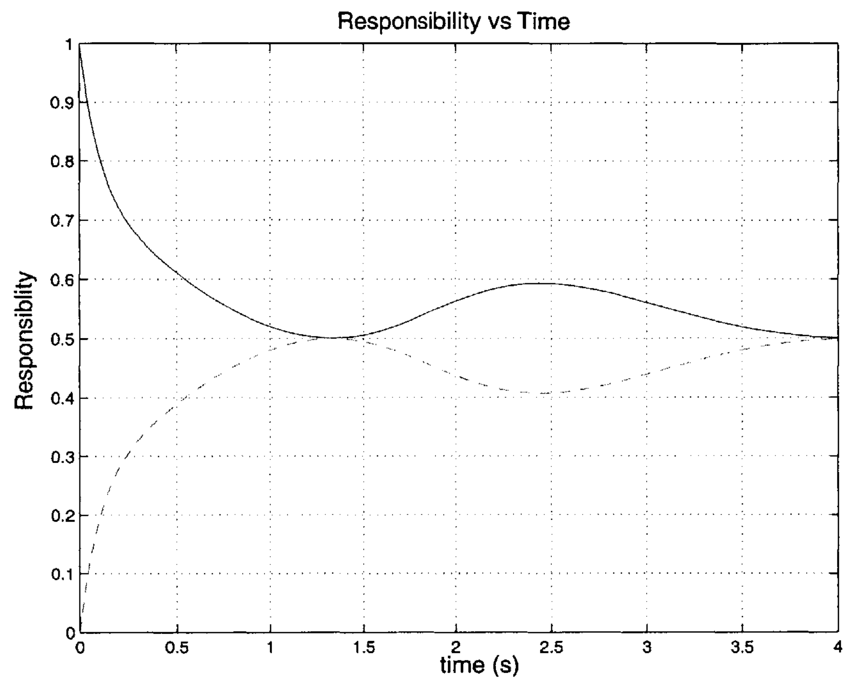

Figure 8-5: The resulting time history of the responsibilities for the simplified simulated system.

too, have negative eigenvalues and uncoupled eigenvectors and are stable about this equilibrium point. The weights, with their zero eigenvalues, are neutrally stable, confirming the plot of figure 8-4. The weights have freedom to move about this equilibrium state (where the system is at rest), provided they satisfy $\lambda_1 \boldsymbol{v}_1 + \lambda_2 \boldsymbol{v}_2 = \boldsymbol{v}^*$. But it has already been shown that the only equilibrium state for the controller, when the physical system is not at rest, is when both weight vectors are equal to the optimum weight vector.

To confirm the results of the stability analysis several simulations were performed. The nonlinear system was initialized to the equilibrium state considered (except x_o which was given a nonzero initial value, .9, to set the system in motion). As expected, $\mathcal{X}_o$, was a stable equilibrium point. The weights, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, and the responsibilities, λ_1 and λ_2, all remained constant. When the system was initialized with responsibility values other than .5, the responsibility quickly settled back to $\lambda_1 = \lambda_2 = .5$ (figure 8-5).

With the stability of the simplified nonlinear system established, what has been learned of the actual system? In general, the actual system, with it's motor commands, and physical dynamics that are not composed of symmetric Gaussians, does not have an uncoupled state matrix $\boldsymbol{A}$. In addition, $\boldsymbol{v}^*$ most likely does not exist (the true form of the function is known, and is not decomposable as a set of Gaussians, although with a large enough boundary layer it could be estimated within tolerance). Yet when the actual system of this thesis is analyzed, one characteristic remains unchanged: the responsibility is only constant when $\boldsymbol{v}_1 = \boldsymbol{v}_2$. The soft max function still remains, implicitly, a function of the state of the physical system. As such, it will only yield a constant response, for an active arm, when both forward dynamic arms are identical. This then implies that the responsibility will only be in equilibrium when both responsibilities are one half. Just as the simplified system, lacking motor commands and having forward dynamics that conveniently allow for an exact forward estimate, cannot retain multiple memories in the form of motor behaviors, neither can the actual, more complex, system.

Chapter 9

Conclusions

9.1 Review

Before drawing any conclusions about the work done here, it is appropriate to present a brief review of some of the options considered and taken, in this thesis, and their effects on the results of this work. The physical model for example, was modeled simply, but was arguably a practical "low-order" approximation of the human arm. The "arm" could have been modeled in a more complicated fashion, e.g. configuration-dependent moment arms for the muscles, or nonlinear stiffness and damping, but this would not significantly alter the abilities of the controller.

For ease of computation and training, linear network models were chosen. Nonlinear models could have been chosen, but this would have added unnecessary complexity. Despite altering training (by including local minima), nonlinear networks would not change the abilities of the controller to produce correct motor commands, or the stability of controller while switching. In fact, it would likely just make training more difficult while learning motor behaviors.

Three possible learning paradigms were considered: direct inverse learning, distal supervised learning, and feedback error learning. Direct inverse learning was not chosen to train the models, for it was felt that it was not a plausible representation for a biologically inspired system. Primarily this is because it is not a goal oriented learning scheme. If used to train the inverse models for motor commands it would learn by comparing actual muscle lengths with estimated muscle lengths, rather than training with desired endpoint coordinates, which was the goal. Another difficulty is the problem of non-convex sets, a problem direct inverse learning is not equipped to deal with. The distal supervised and feedback error learning schemes however, are biologically plausible. The inverse kinematic model was trained using the distal supervisor, and the inverse dynamic models were trained using the feedback error scheme. Although computationally more demanding, the distal supervised teacher could be acquired through internal models -mathematically, it's just another implementation of a gradient descent. Similarly, it could be argued that sensory feedback and an appropriate internal model are all that is required for an error signal, when using a feedback error learning scheme.

The gating, or switching, mechanism was the source of the controller's inability to retain multiple motor behaviors in the form of learned weights. The soft max calculation is implicitly a function of the physical state of the arm and only yields constant responsibility updates when both forward dynamic weight vectors are identical. In summary, the modules were stable about the current context, but the switching mechanism could not discriminate between contexts in a stable manner. Similar schemes proposed in the literature [38], will necessarily face the same issues.

9.2 Recommendations For Future Work

Having identified the problem, the next step should be to seek out a solution. If switching is to remain a function of the forward model's estimates, two options exist for stabilizing the responsibility calculation. Multiple, learned, stable equilibria will have to be incorporated into the switching mechanism. Under such a circumstance, the responsibility signal would migrate to learned sets of responsibility values for different contexts. Perhaps with the use of another neural network this might be accomplished; it might prove sufficient for the controller to retain multiple motor behaviors. Another alteration would be to devise a switching mechanism that relies not on the physical state of the arm, as the current one does, but on an implicit property of the forward models themselves. That is, rather than determining a responsibility signal based on a state dependent calculation, the responsibility should be determined through some means that does not change as the arm moves about in it's workspace. This too, might prove sufficient. If instead, switching between modules is not a function of the forward estimates, a more effective means might be based entirely on sensory feedback. Perhaps another network could be used to select modules based on sensory information that would be indicative of a context [38]. If a feedback error learning scheme is still used, the forward dynamic models are not needed and may be neglected altogether.

Once the switching mechanism is stabilized, there is still the issue of training the modules. For if the responsibility signal is to remain a smoothly varying, rather than an "all-or-none", value, then learning will drive *all* modules to converge on the current context parameters. This work did address this issue with the use of boundary layers. By defining an arbitrary boundary layer for both the forward and inverse dynamic models, training would eventually halt for all modules. Some would halt sooner than others, based on their responsibility signal, regardless of a non-zero training error. Although there is no reason to suspect that this would not work with a stable responsibility signal, another approach would be to add a time component to learned modules, such that they are more difficult to adapt after some arbitrary amount of time following training. There is evidence to support such a temporal component for the consolidation of memory [5]. This would further slow the convergence of modules to the current context.

The proposed controller's ability to exhibit motor learning, in particular, motor learning of multiple behaviors, was investigated in an effort to examine the two broad goals of this thesis: one, research useful control architectures and, two, better the understanding of the human motor system. With respect to the first goal, it can be said, that while it was shown that good performance was possible in terms of trajectory following, it has been demonstrated through simulations, and proven mathematically, that the control system presented in this thesis is not capable of retaining multiple motor behaviors. Any further investigations into learning multiple motor memories must devise a stable means of switching between them. Nonetheless, a controller equipped with a neural network model to produce equilibrium joint coordinates for dynamic motion of a two link arm, using feedback error learning, has displayed itself to be a viable option for heuristic motor control.

With respect to the second goal, the conclusions are less obvious. The controller's inability to demonstrate motor learning of several behaviors limits its utility in revealing hidden processes of the human system. What can be postulated based on the results here, is what the human motor system does *not* do. It is likely that the controller in this thesis was equipped with better forward estimates than the human motor system is. Given a position, velocity and instantaneous equilibrium muscle lengths (or some similar motor command) it is unlikely that the human motor system produces a more precise estimate of the forward dynamics of the endpoint than the forward models used here. Yet for a normally-functioning human, the two reaching movements in the two contexts represent extremely different environments, which are easily discernible, whereas the controller has displayed inability, to discriminate between the two (recall the ideal responsibility signal). This would lead one to believe the human motor system uses more information than a forward estimate of endpoint acceleration, and/or a different means of updating responsibilities.

The complete motor system (controller and arm) are a reasonable first pass at an architecture that emulates the human motor system. Therefore, in terms of human motor learning, the feedback error paradigm, a biologically plausible option, has proven itself to be a practicable alternative and can thus be considered a contender for use in the human motor system.

Recall that an important application for a working model of human motor control is in the field of rehabilitation and neuro-recovery. As previously mentioned, it is not clear whether or not motor recovery is the same, or a similar process, as motor learning. To investigate such a claim a similar architecture to the one presented here could be researched. If a motor impairment is a damaged module (e.g. altered weights) then recovery could represent one, or a combination of the following hypothetical scenarios. A new module could replace the damaged module's behavior, which would require a learning period, or a new module could be learning to supplement the damaged module's motor commands. Research into these hypotheses represents just the beginnings of what could be accomplished in the field nero-recovery with a clearer understanding of the human motor system.

Appendix A

Two-Link Dynamics

The equations of motion for a two-link manipulator moving in the plane, with link masses, m_1 and m_2, link inertias, I_1 and I_2, link lengths, l_1 and l_2, and center of mass locations, l_{1cm} and l_{2cm}, are shown below,

$$\begin{bmatrix} H_{11} H_{12} \\ H_{21} H_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} C_{11} C_{12} \\ C_{21} C_{22} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{A.1}$$

H, is the inertia matrix, composed of the configuration dependent inertia terms, ***C***, is a matrix of velocity induced, Coriolis and centripetal acceration terms, and τ is a vector of joint torques. The terms are computed as follows:

$$\begin{aligned} H_{11} &= m_1 l_{1cm}^2 + m_2(l_1^2 + l_{2cm}^2 + 2 l_1 l_{2cm} cos(\theta_2)) + I_1 + I_2 \\ H_{12} &= m_2(l_{2cm}^2 + l_1 l_{2cm} cos(\theta_2)) + I_2 \\ H_{22} &= m_2 l_{2cm}^2 + I_2 \\ C_{11} &= -m_2 l_1 l_{2cm} sin(\theta_2)\dot{\theta}_2 \\ C_{12} &= -m_2 l_1 l_{2cm} sin(\theta_2)(\dot{\theta}_1 + \dot{\theta}_2) \\ C_{21} &= m_2 l_1 l_{2cm} sin(\theta_2)\dot{\theta}_1 \\ C_{22} &= 0 \end{aligned} \tag{A.2}$$

Appendix B

Gradient Descent

for a radial basis function network, defined as follows,

$$\hat{y} = \phi(x, \boldsymbol{w}) = \sum_{i=1}^{n} w_i g_i(x) = \sum_{i=1}^{n} w_i \exp\left(\frac{-(x-\xi_i)^2}{\sigma_i^2}\right) \tag{B.1}$$

and a cost function,

$$E = \frac{1}{2}(y - \hat{y})^2 \tag{B.2}$$

the derivative of E with respect to the i^{th} weight is,

$$\frac{\partial E}{\partial w_i} = -\frac{\partial \hat{y}}{\partial w_i}(y - \hat{y}) = -g_i(y - \hat{y}) \tag{B.3}$$

the subseqent gradient of E with respect to the weight vector $\boldsymbol{w}$ is

$$\nabla_{\boldsymbol{w}} E = -\frac{\partial \hat{y}}{\partial \boldsymbol{w}}^T (y - \hat{y}) \tag{B.4}$$

where $\frac{\partial \hat{y}}{\partial \boldsymbol{w}}$ is an $n \times 1$ array

$$[g_1 \; g_2 \; \cdots \; g_n] \tag{B.5}$$

In general, the Jacobian, $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{w}}$, for an m dimensional function $\hat{\boldsymbol{y}}$ is defined as,

$$\begin{bmatrix} \frac{\partial \hat{y}_1}{\partial w_1} & \frac{\partial \hat{y}_1}{\partial w_2} & \cdots & \frac{\partial \hat{y}_1}{\partial w_n} \\ \frac{\partial \hat{y}_2}{\partial w_1} & \frac{\partial \hat{y}_2}{\partial w_2} & \cdots & \frac{\partial \hat{y}_2}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_m}{\partial w_1} & \frac{\partial \hat{y}_m}{\partial w_2} & \cdots & \frac{\partial \hat{y}_m}{\partial w_n} \end{bmatrix} \tag{B.6}$$

Appendix C

Lyapunov Stability

Lyapunov's indirect or linearization method is concerned with local stability of a nonlinear system about some equilibrium point, $\mathcal{X}_o$, such that

$$\dot{\mathcal{X}} = \boldsymbol{F}(\mathcal{X}_o) = 0 \tag{C.1}$$

Assuming that $\boldsymbol{F}(\mathcal{X})$ is continuously differentiable the system dynamics can be expanded as follows,

$$\dot{\mathcal{X}} = \left[\frac{\partial \boldsymbol{F}}{\partial \mathcal{X}}\right]_{\mathcal{X}_o} (\mathcal{X} - \mathcal{X}_o) + h.o.t. \tag{C.2}$$

Denoting the first term in the expansion, the Jacobian matrix of $\boldsymbol{F}$ as, $\boldsymbol{A}$,

$$\boldsymbol{A} = \left[\frac{\partial \boldsymbol{F}}{\partial \mathcal{X}}\right]_{\mathcal{X}_o} \tag{C.3}$$

the linarized system is

$$\dot{\mathcal{X}} = \boldsymbol{A}\,(\mathcal{X} - \mathcal{X}_o) \tag{C.4}$$

Conclusions about the local stability of the nonlinear system may be then be drawn based on analysis of this linearized system.

For the simplified system of Chapter 8, the actual nonlinear system is,

$$\dot{x} = \dot{x} \tag{C.5}$$

$$\ddot{x} = e^{-(x-1)^2} - e^{-(x+1)^2} + e^{-(\dot{x}-1)^2} - e^{-(\dot{x}+1)^2} \tag{C.6}$$

$$\dot{\boldsymbol{v}}_1 = (\partial\phi/\partial\boldsymbol{v})^T(\ddot{x} - \hat{\ddot{x}}) \tag{C.7}$$

$$\dot{\boldsymbol{v}}_2 = (\partial\phi/\partial\boldsymbol{v})^T(\ddot{x} - \hat{\ddot{x}}) \tag{C.8}$$

$$\dot{\lambda}_1 = \frac{1}{\tau_\lambda}\left[-\lambda_1 + \frac{e^{(-(\ddot{x}-\hat{\ddot{x}}_1)^2)}}{e^{(-(\ddot{x}-\hat{\ddot{x}}_1)^2)} + e^{(-\ddot{x}-\hat{\ddot{x}}_2)^2)}}\right] \tag{C.9}$$

$$\dot{\lambda}_2 = \frac{1}{\tau_\lambda}\left[-\lambda_2 + \frac{e^{(-(\ddot{x}-\hat{\ddot{x}}_2)^2)}}{e^{(-(\ddot{x}-\hat{\ddot{x}}_1)^2)} + e^{(-\ddot{x}-\hat{\ddot{x}}_2)^2)}}\right] \tag{C.10}$$

Equations C.7 and C.8 represent the following rate equations,

$$\dot{v}_{ij} = \lambda_i g_j(\ddot{x} - \hat{\ddot{x}}) \tag{C.11}$$

where g_j is,

$$g_1 = e^{(-(x-1)^2)} \tag{C.12}$$

$$g_2 = e^{(-(x+1)^2)} \tag{C.13}$$

$$g_3 = e^{(-(\dot{x}-1)^2)} \tag{C.14}$$

$$g_4 = e^{(-(\dot{x}+1)^2)} \tag{C.15}$$

The complete state, $\mathcal{X}$, is

$$\mathcal{X} = [x, \dot{x}, v_{11}, v_{12}, v_{13}, v_{14}, v_{21}, v_{22}, v_{23}, v_{24}, \lambda_1, \lambda_2]^T \tag{C.16}$$

Proceeding along with the linearization we obtain, for the $\dot{x}$ terms,

$$\frac{\partial\dot{x}}{\partial\dot{x}} = 1 \tag{C.17}$$

and all other derivatives are 0. For the $\ddot{x}$ terms,

$$\frac{\partial\ddot{x}}{\partial x} = 2(x-1)g_1 - 2(x+1)g_2 \tag{C.18}$$

$$\frac{\partial\ddot{x}}{\partial\dot{x}} = 2(\dot{x}-1)g_3 - 2(\dot{x}+1)g_4 \tag{C.19}$$

and all other derivatives are 0. For the $\partial\dot{v}_{ij}/\partial x$ terms, where $i = j = 1, 2$, we have,

$$\frac{\partial\dot{v}_{ij}}{\partial x} = -2\lambda_i(x-1)g_j(\ddot{x} - \hat{\ddot{x}}) + \lambda_i g_j(\frac{\partial\ddot{x}}{\partial x} - \frac{\partial\hat{\ddot{x}}}{\partial x}) \tag{C.20}$$

for $i = 1, 2; j = 3, 4$ we have,

$$\frac{\partial\dot{v}_{ij}}{\partial x} = \lambda_i g_j(\frac{\partial\ddot{x}}{\partial x} - \frac{\partial\hat{\ddot{x}}}{\partial x}) \tag{C.21}$$

For the terms $\partial \dot{v}_{ij}/\partial \dot{x}$, where $i = j = 1, 2$, we have,

$$\frac{\partial \dot{v}_{ij}}{\partial \dot{x}} = \lambda_i g_j \left(\frac{\partial \ddot{x}}{\partial \dot{x}} - \frac{\partial \hat{\ddot{x}}}{\partial \dot{x}}\right) \tag{C.22}$$

where $i = 1, 2; j = 3, 4$ we have,

$$\frac{\partial \dot{v}_{ij}}{\partial \dot{x}} = -2\lambda_i (x-1) g_j (\ddot{x} - \hat{\ddot{x}}) + \lambda_i g_j \left(\frac{\partial \ddot{x}}{\partial \dot{x}} - \frac{\partial \hat{\ddot{x}}}{\partial \dot{x}}\right) \tag{C.23}$$

For the terms $\partial \dot{v}_{ij}/\partial v_{kl}$, we have,

$$\frac{\partial \dot{v}_{ij}}{\partial v_{kl}} = -\lambda_i \lambda_k g_j g_l \tag{C.24}$$

For ther terms $\partial \dot{v}_{ij}/\partial \lambda_k$, where $i = k$, we have,

$$\frac{\partial \dot{v}_{ij}}{\partial \lambda_i} = g_j (\ddot{x} - \hat{\ddot{x}}) - \lambda_i g_j \hat{\ddot{x}}_i \tag{C.25}$$

for $i \neq k$, we have,

$$\frac{\partial \dot{v}_{ij}}{\partial \lambda_k} = -\lambda_i g_j \hat{\ddot{x}}_k \tag{C.26}$$

For the state terms involving $\dot{\lambda}_i$, the following notation shall be adopted,

$$\dot{\lambda}_i = \frac{1}{\tau_\lambda} \left[-\lambda_i + \frac{e^{(-(\ddot{x}-\hat{\ddot{x}}_i)^2)}}{e^{(-(\ddot{x}-\hat{\ddot{x}}_i)^2)} + e^{(-\ddot{x}-\hat{\ddot{x}}_j)^2)}} \right] = \frac{1}{\tau_\lambda} \left[-\lambda_i + f_i/g \right] \tag{C.27}$$

where $g = f_1 + f_2$, and $f_i = e^{(-(\ddot{x}-\hat{\ddot{x}}_i)^2)}$, and $\acute{g} = \acute{f}_1 + \acute{f}_2$. Then the derivatives of $\dot{\lambda}_i$ are of the following form,

$$\acute{\dot{\lambda}}_i = \frac{1}{\tau_\lambda} \left[(\acute{f}_i g - f_i \acute{g})/g^2 \right] \tag{C.28}$$

For the terms $\partial \dot{\lambda}_i/\partial x$, we have,

$$\begin{aligned} \acute{f}_1 &= -2(\ddot{x} - \hat{\ddot{x}}_1)\left(\frac{\partial \ddot{x}}{\partial x} - \frac{\partial \hat{\ddot{x}}_1}{\partial x}\right) f_1 \\ \acute{f}_2 &= -2(\ddot{x} - \hat{\ddot{x}}_2)\left(\frac{\partial \ddot{x}}{\partial x} - \frac{\partial \hat{\ddot{x}}_2}{\partial x}\right) f_2 \end{aligned} \tag{C.29}$$

For the terms $\partial \dot{\lambda}_i/\partial \dot{x}$, we have,

$$\begin{aligned} \acute{f}_1 &= -2(\ddot{x} - \hat{\ddot{x}}_1)\left(\frac{\partial \ddot{x}}{\partial \dot{x}} - \frac{\partial \hat{\ddot{x}}_1}{\partial \dot{x}}\right) f_1 \\ \acute{f}_2 &= -2(\ddot{x} - \hat{\ddot{x}}_2)\left(\frac{\partial \ddot{x}}{\partial \dot{x}} - \frac{\partial \hat{\ddot{x}}_2}{\partial \dot{x}}\right) f_2 \end{aligned} \tag{C.30}$$

For the terms $\partial\dot{\lambda}_i/\partial v_{jk}$, where $i \neq j$ we have,

$$\acute{f}_i = -2(\ddot{x} - \hat{\ddot{x}}_j)g_k f_i \tag{C.31}$$

and where $i = k$, we have,

$$\acute{f}_i = 0 \tag{C.32}$$

For the terms $\partial\dot{\lambda}_i/\partial\lambda_j$, we have,

$$\frac{\partial\dot{\lambda}_i}{\partial\lambda_j} = -\frac{1}{\tau_\lambda} \tag{C.33}$$

when $i = j$ and 0 otherwise.

The resulting matrix $\boldsymbol{A}$ evaluated about equilibrium is composed of three block matrices. $\boldsymbol{A}_x$, $\boldsymbol{A}_v$ and $\boldsymbol{A}_\lambda$ representing the physical dynamics, the weights and the responsibilites respectively.

$$\boldsymbol{A} = \begin{bmatrix} [A]_{\ddot{x}} & 0 & 0 \\ 0 & [A]_v & 0 \\ 0 & 0 & [A]_\lambda \end{bmatrix} \tag{C.34}$$

where,

$$\boldsymbol{A}_x = \begin{bmatrix} 0 & 1 \\ -4e^{-1} & -4e^{-1} \end{bmatrix} \tag{C.35}$$

$$\boldsymbol{A}_v = \begin{bmatrix} -\lambda_1\lambda_2 e^{-2} & \cdots & -\lambda_1\lambda_2 e^{-2} \\ \vdots & -\lambda_1\lambda_2 e^{-2} & \vdots \\ -\lambda_1\lambda_2 e^{-2} & \cdots & -\lambda_1\lambda_2 e^{-2} \end{bmatrix}_{8\times 8} \tag{C.36}$$

$$\boldsymbol{A}_\lambda = \begin{bmatrix} -1/\tau_\lambda & 0 \\ 0 & -1/\tau_\lambda \end{bmatrix} \tag{C.37}$$

and the eigen values of $\boldsymbol{A}$ are, $(-2e^{-1} \pm j2\sqrt{e^{-1} - e^{-2}})$, for $\boldsymbol{A}_x$, $(0, 0, 0, 0, 0, 0, 0, -8\lambda_1\lambda_2 e^{-2})$, for $\boldsymbol{A}_v$, and $(-1/\tau_\lambda, -1/\tau_\lambda)$, for $\boldsymbol{A}_\lambda$.

In Chapter 8 the circumstances necessary for a constant responsibility update, regardless of x and $\dot{x}$, are considered. They are found when the partial of the responsibility update $\hat{\lambda}$ with respect to x and $\dot{x}$ are set to zero. Refering to equation

C.28 the derivatives are zero when the numerator is zero. Using equations C.29 and C.30 the partial derivatives are (the derivatives of λ_1 shall be used, although either are valid),

$$\frac{\partial \hat{\lambda}_1}{\partial x} = -2(\ddot{x} - \hat{\ddot{x}}_1)\left(\frac{\partial \ddot{x}}{\partial x} - \frac{\partial \hat{\ddot{x}}_1}{\partial x}\right) f_1 \left[f_1 + f_2\right]$$
$$+ \left[2(\ddot{x} - \hat{\ddot{x}}_1)\left(\frac{\partial \ddot{x}}{\partial x} - \frac{\partial \hat{\ddot{x}}_1}{\partial x}\right) f_1 + 2(\ddot{x} - \hat{\ddot{x}}_2)\left(\frac{\partial \ddot{x}}{\partial x} - \frac{\partial \hat{\ddot{x}}_2}{\partial x}\right) f_2\right] f_1 \tag{C.38}$$

where,

$$\frac{\partial \ddot{x}}{\partial x} = 2(x-1)g_1 - 2(x+1)g_2$$

$$\frac{\partial \hat{\ddot{x}}_1}{\partial x} = -2v_{11}(x-1)g_1 - 2v_{12}(x+1)g_2$$

$$\frac{\partial \hat{\ddot{x}}_2}{\partial x} = -2v_{21}(x-1)g_1 - 2v_{22}(x+1)g_2$$

and,

$$\frac{\partial \hat{\lambda}_1}{\partial \dot{x}} = -2(\ddot{x} - \hat{\ddot{x}}_1)\left(\frac{\partial \ddot{x}}{\partial \dot{x}} - \frac{\partial \hat{\ddot{x}}_1}{\partial \dot{x}}\right) f_1 \left[f_1 + f_2\right]$$
$$+ \left[2(\ddot{x} - \hat{\ddot{x}}_1)\left(\frac{\partial \ddot{x}}{\partial \dot{x}} - \frac{\partial \hat{\ddot{x}}_1}{\partial \dot{x}}\right) f_1 + 2(\ddot{x} - \hat{\ddot{x}}_2)\left(\frac{\partial \ddot{x}}{\partial \dot{x}} - \frac{\partial \hat{\ddot{x}}_2}{\partial \dot{x}}\right) f_2\right] f_1 \tag{C.39}$$

where,

$$\frac{\partial \ddot{x}}{\partial \dot{x}} = 2(x-1)g_3 - 2(x+1)g_4$$

$$\frac{\partial \hat{\ddot{x}}_1}{\partial \dot{x}} = -2v_{13}(x-1)g_3 - 2v_{14}(x+1)g_4$$

$$\frac{\partial \hat{\ddot{x}}_2}{\partial \dot{x}} = -2v_{23}(x-1)g_3 - 2v_{24}(x+1)g_4$$

Grouping all terms together, for both $\partial \hat{\lambda}_1/\partial x$ and $\partial \hat{\lambda}_1/\partial \dot{x}$, the derivatives are identically zero when $\boldsymbol{v}_1 = \boldsymbol{v}_2 = \boldsymbol{v}^*$.

Bibliography

[1] Albus, J.S. (1975) A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Trans. ASME J. Dynamic Syst. Meas. Contr.* **97**:220-227.

[2] Bizzi, E., Accornero, N., Chapple, W. & Hogan, N. (1984) Posture Control and Trajectory Formation During Arm Movement. *The Journal of Neuroscience* **4**:2738-2744.

[3] Bizzi, E., Mussa-Ivaldi, F., & Giszter, S. (1991) Computations Underlying the Execution of Movement: A Biological Perspective. *Science* **253**:287-291.

[4] Bizzi, E. (1993) Intermediate representations in the formation of arm trajectories. *Current Opinion in Neurobiology*, **3**:925-931.

[5] Brashers-Krug, T., Shadmehr, R. & Bizzi, E. (1996) Consolidation in Human Motor Memory. *Nature* **382**:252-255.

[6] Chapman, C.E., Spidalieri, G. & Lamarre, Y. (1984) Discharge properties of area 5 neurons during arm reaching movements triggered by sensory stimuli in the monkey. *Brain Research* **309**:63-67.

[7] Feldman, A.G. (1966) Fucntional Tuning of the Nervous System During Control of Movement or Maintenance of a Steady Posture. III. Mechanographic Analysis of the Execution by Man of the Simplest Motor Tasks. *Biofizika* **11**:667-675.

[8] Flash, T. & Hogan, N. (1985) The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model. *J. Neuroscience*, **4**:1688-1703.

[9] Flash, T. (1987) The Control of Hand Equilibrium Trajectories in Multi-Joint Arm Movements. *57*:257-274.

[10] Flash, T., Hogan, N. & Richardson, M. (2000) Optimization Principles In Motor Control. in Arbib, M. A. (ed.) The Handbook of Brain Theory and Neural Networks, 2nd. Ed., MIT Press

[11] Gandolfo, F., Li, C.-S.R., Benda, B.J., Padoa Schioppa, C. & Bizzi, E. In Press.

[12] Gomi, H. & Kawato, M. (1997) Human arm stiffness and equilibrium-point trajectory during multi-joint movement. *Biological Cybernetics* **76**:163-171.

[13] Hogan, N. (1980) Mechanical Impedance Control in Assistive Devices and Manipulators. *Proceedings of the Joint Automatic Control Conference*, **1**:TA10-B.

[14] Hogan, N. (1984) An Organizing Principle for a Class of Voluntary Movements. *The Journal of Neuroscience*, **4**:2745-2754.

[15] Hogan, N. (1984) Adaptive Control of Mechanical Impedance by Coactivation of Antagonist Muscles. *Transactions on Automatic Control*, **29**:681-690.

[16] Hogan, N. (1985) The Mechanics of Multi-Joint Posture and Movement Control. *Biological Cybernetics*, **52**:315-331.

[17] Hogan, N. (1987) Planning and Execution of Multijoint Movements. *Can. J. Physiol. Pharmacol.*, **66**:508-517.

[18] Hogan, N. (1990) Mechanical Impedance of Single- and Multi-Articular Systems. *Multiple Muscle Systems: Biomechanics and Movement Organization*, Ch. **9**.

[19] Jordan, M.I. & Rumelhart, D.E. (1992) Forward Models: Supervised Learning with a Distal Teacher. *Cognitive Science*, **16**:307-354.

[20] Jordan, M.I. & Jacobs, R.A. (1994) Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, **6**:181-214.

[21] Kalaska, J.F., Cohen, D.A.D., Prud'homme, M. & Hyde, M.L. (1990) Parietal area 5 neuronal activity encodes movement kinematics, not movement dynamics. *Experimental Brain Research* **80**:351-364.

[22] Kawato, M., Furukawa, K. & Suzuki, R. (1987) A Hierarchical Neural-Network Model for Control and Learning of Voluntary Movement. *Biological Cybernetics* **57**:169-185.

[23] Kawato, M. (1990) Feedback-Error-Learning Neural Network for Supervised Motor Learning. Advanced Neural Computers

[24] Girosi, F. & Poggio, T. (1989) Networks and the Best Approximation Property. A.I. Memo No. 1164.

[25] Morasso, P. (1981) Spatial Control of Amr Movements. *Experimental Brain Research* **42**:223-227.

[26] Mussa-Ivaldi, F.A., Hogan, N. & Bizzi, E. (1985) Neural, Mechanical, and Geometric Factors Subserving Arm Posture in Humans. *The Journal of Neuroscience* **5**:2732-2743.

[27] Pascual-Leone, A., Dang, N., Cohen, L.G., Brasil-Neto, J.P., Cammarota, A. & Hallet, M. (1995) Modulation of Muscle Responses Evoked by Transcranial Magnetic Stimulation During the Acquisition of New Fine Motor Skills. *Journal of Neurophysiology* **74**:1037-1045.

[28] Polit, A. & Bizzi, E. (1979) Characteristics of Motor Programs Underlying Arm Movements in Monkeys. *Journal of Neurophysiology* **42**:183-194.

[29] Sanner, R.M., & Slotine, J.J.E. (1992) Gaussian Networks for Direct Adaptive Control. *IEEE Transactions of Neural Networks* **3**:837-863.

[30] Sanner, R.M., & Slotine, J.J.E. (1991) Stable Adaptive Control and Recursive Identification Using Radial Gaussian Networks. Proceedings of the 30th Conference on Decision and Control. 2115-2123.

[31] Shadmehr, R., Mussa-Ivaldi, F.E., & Bizzi, E. (1993) Postural Force Fields of the Human Arm and Their Role in Generating Multi-Joint Movements. *Journal of Neuroscience* **13**:45-62.

[32] Shadmehr, R. (1990) Learning Virtual Equilibrium Trajectories for Control of a Robot Arm. *Neural Computation* **2**:436-446.

[33] Uno, Y., Kawato, M., & Suzuki, R. (1989) Formation and Control of Optimal Trajectory in Human Multijoint Arm Movement: Minimum Torque-Change Model. *Biological Cybernetics* **61**:89-101.

[34] Williams, M. & Lissner, H.R. Biomechanics of Human Motion 1962, W. B. Saunders Company.

[35] Wolpert, D.M., Ghahramani, Z. & Jordan, M.I. (1995) Arm arm trajectories planned in kinematic or dynamic coordinates? An adaptation study. *Experimental Brain Research* **103**:460-470.

[36] Wolpert, D.M., Ghahramani & Jordan, M.I. (1995) An Internal Model for Sensorimotor Integration. *Science* **269**:1880-1882.

[37] Wolpert, D.M. (1997) Computational Approaches to Motor Control. *Trends in Cognitive Sciences* **1**:209-216.

[38] Wolpert, D.M. & Kawato, M. (1998) Multiple paired forward and inverse models for motor control. *Neural Networks II* 1317-1329.

22410 -69