LEARNING CONTROLLERS FOR COMPLEX BEHAVIORAL SYSTEMS

by

Lara S. Crawford and S. Shankar Sastry

Memorandum No. UCB/ERL M96/73

3 December 1996

LEARNING CONTROLLERS FOR COMPLEX BEHAVIORAL SYSTEMS

by

Lara S. Crawford and S. Shankar Sastry

Memorandum No. UCB/ERL M96/73

3 December 1996

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Learning Controllers for Complex Behavioral Systems*

Lara S. Crawford
Graduate Group in Biophysics
and
S. Shankar Sastry
Department of Electrical Engineering and Computer Sciences

University of California at Berkeley

Abstract

Biological control systems routinely guide complex dynamical systems through complicated tasks such as running or diving. Conventional control techniques, however, stumble with these problems, which have complex dynamics, many degrees of freedom, and a task which is often only partially specified (e.g., "move forward fast," or "execute a one-and-one-half-somersault dive"). To address problems like these, we are using a biologically-inspired, hierarchical control structure, in which controllers composed of radial basis function networks learn the controls required at each level of the hierarchy. Through learning and proper encoding of behaviors and controls, some of these difficulties in controlling complex systems can be overcome.

*This research was supported in part by ARO under grants DAAL03-91-G0171, DAAH04-94-G0211. DAAH04-95-05888, and MURI DAAH04-96-1-0341.

1 Introduction

Biological control systems do amazing things. In particular, biological organisms can coordinate the motions of many degrees of freedom to accomplish a complex task with apparent ease and can readily learn new patterns of coordination. The inherent complexity of the dynamics of a system such as a humanoid performing a typical behavior, as well as the high dimensionality, are prohibitive for conventional engineering control schemes. These methods generally rely on having a detailed model of the system, which is an inconvenient assumption in complex systems like a humanoid. Also, traditional methods generally assume an *a priori* desired trajectory, which then raises the issue of path planning. The goal of our work is to develop new control approaches for complex systems like humanoids to produce a desired behavior. The desired behavior is specified as a natural verbal criterion such as "move forward without falling over" for walking, or "execute a full twisting one-and-a-half somersault dive," for platform diving. As the equations of motion for these systems are prohibitively long and complex, we have used the SD/FAST software package (Symbolic Dynamics, Inc. [19]), which uses Kane's formulation, for our simulations.

Our approach for dealing with the complexity of the control problem is inspired by biological systems. First, our controller design has a hierarchical structure which simplifies the control task at each level. The control structure is hybrid in nature; the controllers at each level of the hierarchy compute continuous functions, but their output is decoded into discrete control actions, which then act on the continuous dynamical system. As typically studied, hybrid systems generally have one discrete part and one continuous part, whereas here we have continuous controllers producing discretized controls which act on continuous systems. Finally, the controllers themselves are not preconstructed, but rather learn which controls should be applied to produce the desired actions.

Currently, our control system is open-loop. The system can, through experience and repetition, build up an internal model relating the desired movements to the appropriate control signals. The learning controller can take on many forms; here, we have chosen radial basis function networks. The lower hierarchical levels can be trained by a modified supervised learning algorithm, while the complexity of the task facing the higher levels will require a reinforcement learning approach. For a more robust (and more realistic from a biological point of view) system, though, feedback will be required. In biological systems learning complex new tasks, a progression from a tightly-regulated, closed-loop form of control to open-loop control is often seen as the feed-forward controller becomes more accurate. Understanding control systems of this type is part of an ongoing project; currently, our aim is to understand the feed-forward portions of the controls and how they can be learned.

This paper is organized as follows. In Section 2, we present the diving problem as an example of the type of problem discussed above. We describe our controller design in Section 3, and present our learning algorithms and some preliminary results in Section 4.

2 The Diving Problem

The problem on which we are testing our control designs is that of a human platform diver. The control problem for the diver is as follows: given fixed initial conditions (after leaving the board), execute a certain maneuver (a full twisting one-and-a-half somersault, for example), and then enter the water in a fully extended, vertical position (see [2]). There is no particular desired trajectory specified. This problem has several interesting features. After the diver has left the board, he is subject to angular momentum conservation, which creates a nonholonomic constraint. The diver leaves the board with some initial (non-zero) angular momentum, however, so the system has drift. The drift velocity depends on the configuration of the diver. Since the diver is falling while executing the maneuver, there is a predetermined length of time in which the controls can act. Since the diver generally starts with his momentum totally in the somersault direction, he needs to execute a "throwing" maneuver with his arms to initiate twisting (see [9]). We are currently simulating a diver with ten degrees of freedom in the joints: three in each shoulder, one at each elbow, and one at each hip.

Although much work has been done recently on the control and steering of nonholonomic systems, most of it has been for drift-free systems (for a survey, see [23]). Some specific cases with drift have been addressed ([6]; [24]; [10], e.g.), but very little work exists concerning general systems with drift. Other control approaches to problems like this include that of Hodgins and Raibert [18] and Wooten and Hodgins [32], who divide complicated movements into states of a finite state machine; within each state, motions are regulated by PD controllers. There are some learning approaches to problems of this type in the literature as well: Gorinevsky, Kapitanovsky, and Goldenberg [13] use radial basis functions to learn the controls for steering a space platform with an arm, and Bertsekas and Tsitsiklis [3] use neurodynamic programming to learn to control discrete systems.

We have tested some conventional techniques on a planar, two-joint simplification of the diver model, but these proved unsatisfactory even for the simplified system. A simple learning algorithm applied to the planar diver was more promising (see [8]), and led to our continued work on learning controllers described here. We are also involved in an effort to develop new path-planning methods for nonholonomic systems with drift [11].

3 Learning Controller Architecture

In designing a controller, we have taken some inspiration from biological systems, the original learning controllers. For example, biological systems deal with dynamic complexity through hierarchical organization, with different parts of the motor control system performing different functions (see Figure 1). Our learning controller, shown schematically in Figure 2, is also hierarchical in design. The coordinating controller and the single-degree-of-freedom (single-DOF) controllers all operate in the discrete-time domain, while the plant, a mechanical system, operates in continuous time. The links between the regimes are provided by the decoders, which convert the joint control signals into torques, and the encoder, which

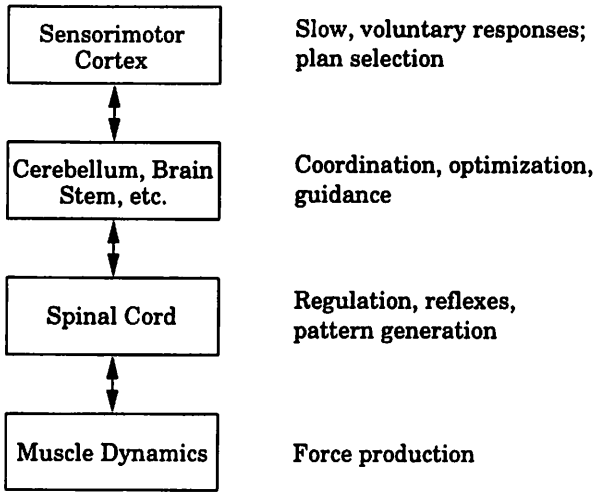

Figure 1: Hierarchy in vertebrate motor control.

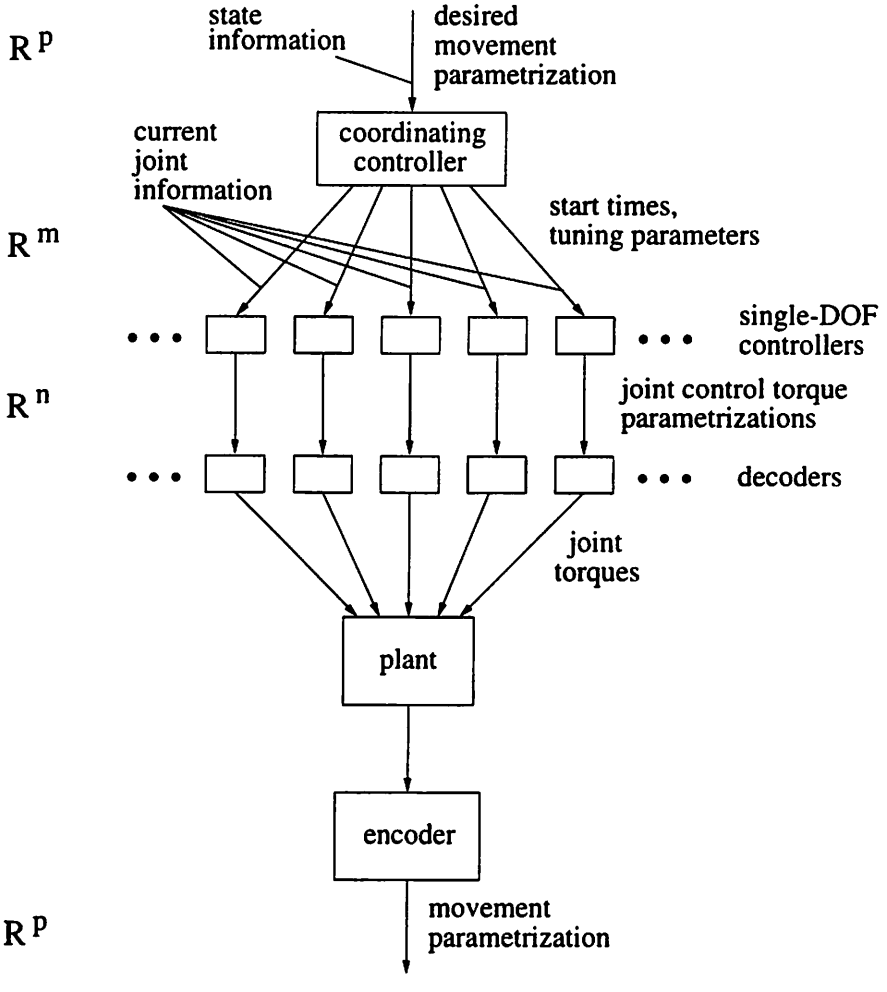

Figure 2: Schematic of a general hierarchical learning controller.

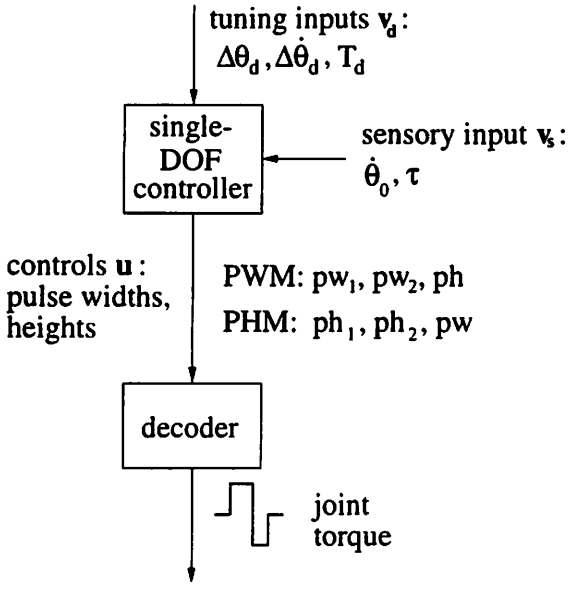

Figure 3: Closeup of a single-DOF controller.

converts the resulting motion into a movement parametrization. In general, this movement representation could include parameters such as the total distance moved forward, total number of rotations about some axis, or total movement time; the representation will depend on the important features of the type of movement being controlled. The controllers together with the decoder, the plant, and the encoder form a hybrid system.

3.1 Single-DOF Controllers

The single-DOF controllers take as input a vector in $\mathbb{R}^m$ from the higher-level coordinating controller, together with some current joint sensor information, and produce an output vector in $\mathbb{R}^n$ defining the joint torque profile. To design the joint control torque parametrization, we again turn to biology for direction. In a behavioral task such as diving, controls which produce a desired behavior are often nonunique, so some restriction of the allowed controls is needed. At low levels of the control hierarchy, biological systems accomplish this restriction through pattern generators. These relatively simple neural networks produce stereotypical bursting patterns which can be tuned by descending commands (i.e., signals from higher hierarchical levels). It has been known for some time that rhythmic movements like walking, swimming, breathing, and chewing are controlled in many animals by periodic pattern generators (for a review, see [15]). More recently, several investigators have found evidence for the existence of low-level controllers in fast, goal-directed, single-joint movements (see, for example, [14]). In the model proposed by Gottlieb, Corcos, and Agarwal in [14], the low-level controller is a pulse generator which produces square activation pulses as inputs to the motoneuron pool. Such a controller would produce the stereotypical patterns often seen in fast, single-joint movements, which are identifiable by their torque, velocity, and double-

or triple-burst EMG profiles. Thus, higher levels in the biological control hierarchy can choose, via tuning, only among the family of controls put out by the pattern generator. The parametrization of possible controls also provides a compact representation of the controls, which allows efficient storage and communication between hierarchical levels.

The single-DOF controllers and decoders we have chosen (see Figure 3) play the role of the pattern generators. They receive the desired change in joint angle and velocity and the desired movement time ($(\Delta\theta_d, \Delta\dot{\theta}_d, T_d) = \mathbf{v_d}$) as tuning parameters from the coordinating controller. The low-level controllers are required to compensate for some of the initial conditions on the joint, so the single-DOF controller takes as parameters the initial velocity of the DOF and the initial effective external torque acting on that DOF, $(\dot{\theta}_0, \tau) = \mathbf{v_s}$. We can then consider each single-DOF controller to be an element of a two-dimensional space of controllers indexed by $\mathbf{v_s}$. For each movement, the appropriate controller from this controller space is selected based on the sensory input. These sensory signals would be provided by information from joint sensors analogous to the stretch receptors and Golgi tendon organs of biological systems.

As there are three free inputs, the control family should have three specifiable parameters, to allow sufficient movement richness while maintaining uniqueness of the controls. Based on the model above, we have chosen torque profiles consisting of two square pulses, as indicated in Figure 3. As these torque profiles have four obvious parameters, namely the pulse heights and widths for the two pulses, we use another idea from [14] to restrict the control family. There, it is hypothesized that the motor control system uses two different control schemes in different conditions, pulse height modulation (PHM) and pulse width modulation (PWM). In our controller design, therefore, the PWM strategy, which we have chosen to apply for movement times larger than some critical time T_{crit}, requires the pulses' heights to be of equal magnitude and opposite direction, while in PHM ($T_d < T_{crit}$), the pulses' widths are equal. Thus there are three control parameters to be specified in either control strategy, one pulse height and two pulse widths in the PWM case, and one pulse width and two heights in the PHM case. Thus for PWM, $\mathbf{u} = (\mathrm{pw}_1, \mathrm{pw}_2, \mathrm{ph})$ (where the sign of the pulse widths indicates the pulse direction), and for PHM, $\mathbf{u} = (\mathrm{ph}_1, \mathrm{ph}_2, \mathrm{pw})$. We have currently implemented two separate controllers for each single-DOF controller, one for the PHM regime and one for the PWM regime, for ease of learning. A switch based on the value of T_d determines which controller is active. The decoder interface converts the output vector of the controller into the two-pulse pattern, which is reminiscent of both bang-bang control and the EMG profile mentioned above. In the future, we may apply a filter to these torques, in analogy to the filtering action of the motoneuron pool, which would produce more biologically realistic torque profiles. The output of a single DOF in the plant is represented by a single-joint encoder as a vector in $\mathbb{R}^m$, $\mathbf{v_p} = (\Delta\theta, \Delta\dot{\theta}, T)$. In between feed-forward movements, a PD controller is switched on to keep the joints from drifting. Joint limits are also enforced by stiff PD controllers which become active at the boundaries of the joint's angle range.

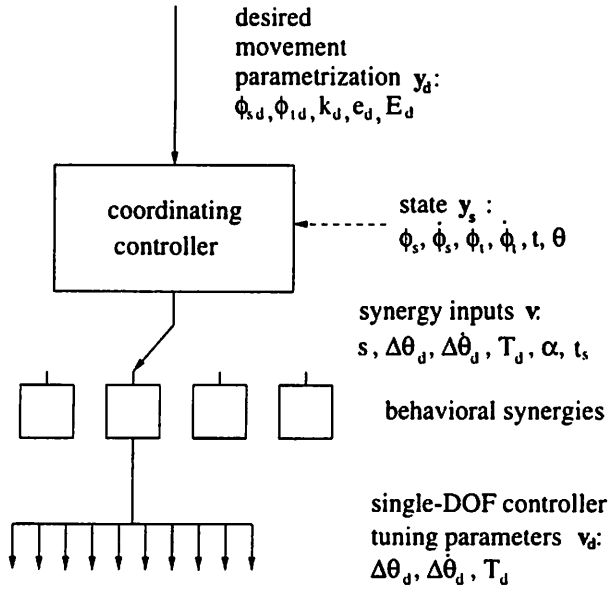

Figure 4: Closeup of the coordinating controller. Dotted arrow signifies sampled updating.

3.2 Coordinating Controller

The design of our proposed coordinating controller was also inspired by biology. Various researchers have shown that the most important piece of information for humans learning new, complex tasks is the relative timing between the different movement segments or the phasing between continuous movements (see, e.g., [31]; [28]; [21], Ch. 1). Thus, complex skills can be learned by combining more basic movement building blocks in an appropriate way. It is also the impression of athletes learning complex skills, like dives, that once they learn the basic building blocks, such as how to start dive rotation and how to pull out of a dive, they can learn different new skills by simply learning how to put the pieces together.

In our design, the coordinating controller takes as input the desired movement parametrization, a vector in $\mathbb{R}^p$, as well as some state information, and and is required to output the tuning inputs for each single-DOF controller (see Figure 4). To simplify the task of the controller, we define multi-DOF synergies, or behaviors, appropriate to the desired class of movements, such as "pike" or "throw" (the arm motion that initiates twisting) for the diving problem. The controller need only specify the synergy s to activate, the tuning parameters for one single-DOF controller in the synergetic group, coupling parameters α determining the relative amplitudes of motion of the other DOFs in the synergy, and the time to wait before executing the synergy t_s. To simplify learning (see Section 4), the controller must activate only one synergy at a time, and thus essentially acts like a state machine, with the states corresponding to the behaviors being executed. We have also assumed, for simplicity, that $\dot{\theta}_d$ will always equal zero for all degrees of freedom. This coupling reduces the number of degrees of freedom the coordinator needs to control directly. Biological systems show similar synergetic coupling. In pointing movements involving both the elbow and the shoulder, for

example, the velocity profiles are identical for movements in which the two joints are required to rotate in the same or opposite directions; only the signs and relative amplitudes change. In the future, the coordinating controller may also have some control over the diver's initial conditions.

We have initially chosen a movement representation (encoder) which specifies the total angle of rotation in the somersault and twist directions (these are unambiguous since we can assume that rotation in the "cartwheel" direction will be small), how tight a pike the diver executed, the squared error of the joint angles from the desired final entry position, and an estimate of the total energy expended in the dive ($\mathbf{y_p} = (\phi_s, \phi_t, k, e, E)$). Other variations on this type of parametrization are possible, of course (c.f. [7]). In the future, we may be required to add more parameters for stylistic considerations or to reduce the number of possible solutions. The state information supplied to the controller consists of the current somersault angle, somersault velocity, twist angle, twist velocity, time, and all ten joint angles ($\mathbf{y_s} = (\phi_s, \dot{\phi}_s, \phi_t, \dot{\phi}_t, t, \theta)$). (To reduce the state space, we have assumed that $\dot{\theta}$ is always near zero at the end of the movement, corresponding with our assumption $\dot{\theta}_d = 0$ above.) The state information is updated only at the completion of a synergetic motion. The infrequent update of state information is roughly similar to a diver's ability to "spot," or take his positional bearings by sighting the water or the board; the diver can only receive this information at most once per rotation.

The controllers described here have certain similarities to Brockett's (u,k,T) hybrid motor control system [5], but here all dynamics are encapsulated in the lower level pattern generators, so the controls are simply vectors rather than time trajectories. Also, the controllers are feed-forward (except for the limited use of PD controllers mentioned above) at this time, though we plan to add feedback in the future. Our control system design has several features similar to Pil and Asada's recursive structure redesign algorithm [25] as well.

4 Learning Algorithms

In biological systems, when the structure of the system is not known *a priori*, an internal model can be built up through learning. The learned model will allow the system to generalize from known tasks to new tasks. Similarly, in our control scheme, the controllers learn the required controls in the absence of a predefined model. In the behavioral literature, controllers like these might be viewed as schemas. The original definition of a schema is simply a learned relationship between the input and required output vectors of the controller [27]. Here, this would correspond to a learned model of the inverse relationship governing the lower levels of the system. Thus the single-DOF controller, for example, is a learned function $\hat{g}^{-1} : \mathbb{R}^m \longrightarrow \mathbb{R}^n$ approximating the inverse of the lumped system of the the decoders, the plant, and the single-DOF encoder, $g : \mathbb{R}^n \longrightarrow \mathbb{R}^m$. The single-DOF controllers were trained with only one degree of freedom in the plant free, and all others fixed.

4.1 Single-DOF Controllers

The implementation of the learning controllers can be done in several ways. Our current implementation uses networks of radial basis functions for both the single-DOF controllers and the coordinating controller (see [16] for an introduction). The output vector $\mathbf{u} = f(\mathbf{v_d}, \mathbf{v_s})$ of a single-DOF controller, for example, is given by

$$u_i = f_i(\mathbf{v_d}, \mathbf{v_s}) = \sum_{j=1}^{N} w_{ij}(\mathbf{v_s})\phi(\frac{||\mathbf{v_d} - \mathbf{v_j}||}{\sigma_j}) = \Phi^T(\mathbf{v_d})\mathbf{w_i}(\mathbf{v_s}),$$

where $\mathbf{v_j}$ is the center of the jth basis function, σ_j defines the spread of the jth basis function, ϕ is the standard basis function itself, and the $w_{ij}(\mathbf{v_s})$s are weights. For our system, we have chosen $\phi(s) = e^{-\frac{s^2}{2}}$. As discussed above, $\mathbf{v_s}$ defines a two-dimensional space of controllers, which appears here as the two-dimensional weight functions $w_{ij}(\mathbf{v_s})$. Our current approach is to train the controller for fixed values of $\mathbf{v_s}$ lying on a grid, and use functional interpolation to obtain controllers for $\mathbf{v_s}$ between the grid points. We are currently using radial basis functions with constant spread arranged in a dense, grid-centered sphere packing, so this interpolation is straightforward. Before being input to the plant, the controls are passed through a squashing function $h(u_i) = \frac{1}{1+e^{-\beta u_i}}$ so the controls are always within allowed ranges. With a radial basis function architecture, only one layer is required to approximate any function, whereas with a conventional neural network architecture, two are needed if the function is discontinuous (see [17] for a brief summary). Thus, if the centers and the functions themselves are fixed, a linear algorithm such as recursive least squares can be applied to the weights.

In our implementation, the situation is a bit more complicated. To use recursive least squares, one needs to obtain an error measure on the $\mathbf{u}$ produced by the controller. Here, all we have available is the error on the plant output. To get around this problem for the single-DOF controllers, we have adopted the scheme shown in Figure 5, with $\mathbf{v_s}$ fixed. The algorithm can be summarized as follows:

1. A random $\mathbf{v_d}$ within the controller's effective range is generated and passed to the controller.

2. The controller produces output $\mathbf{u}$ in response to its input, and this control is passed through the squashing function.

3. The decoder produces a torque profile corresponding to $\mathbf{u}$, and a single DOF of the plant (all others held fixed) is simulated with that control.

4. The single-DOF encoder converts the plant output into a vector in $\mathbb{R}^n$, $\mathbf{v_p} = (\Delta\theta, \Delta\dot{\theta}, T)$. $(\mathbf{v_p}, \mathbf{u})$ is a valid training pair for the network.

5. $\mathbf{v_p}$ is fed back into the controller as a new input, $\mathbf{v'_d}$. If $\mathbf{v'_d}$ is outside the range of the controller, the trial is aborted.

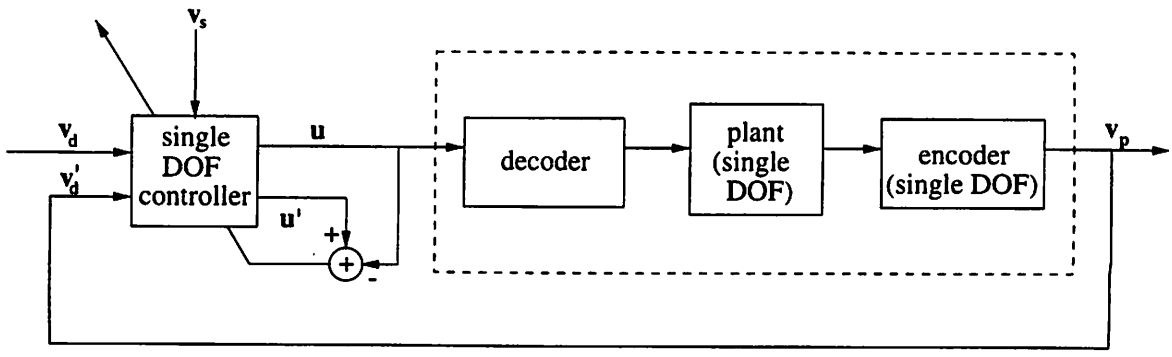

Figure 5: On-line training for the single-DOF controller.

6. The controller produces another output, $\mathbf{u}'$, in response to the new input.

7. The error between $\mathbf{u}'$ and $\mathbf{u}$ is used with a recursive least squares algorithm to adjust the weights in the radial basis function network.

The recursive least squares update we are using, for each control u_i, is:

$$\begin{aligned}
\epsilon(n+1) &= u_i(n) - u_i'(n) = u_i(n) - \Phi^T(n)\mathbf{w_i}(n) \\
\mathbf{k}(n+1) &= \frac{\mathbf{P}(n)\Phi(n)}{1 + \Phi^T(n)\mathbf{P}(n)\Phi(n)} \\
\mathbf{w_i}(n+1) &= \mathbf{w_i}(n) + \mathbf{k}(n+1)\epsilon(n+1) \\
\mathbf{P}(n+1) &= \mathbf{P}(n) - \mathbf{k}(n+1)\Phi^T(n)\mathbf{P}(n)
\end{aligned}$$

$\mathbf{P}(0)$ is set to the identity plus small random perturbations along the diagonal.

The recursive least squares algorithm acts to minimize $||\mathbf{u} - \mathbf{u}'||$, which implies, by definition, $||f(\mathbf{v_d}, \mathbf{v_s}) - f(\mathbf{v_d'}, \mathbf{v_s})||$ is also minimized. The structure of the radial basis function net certainly would permit $f(\mathbf{v_d'}, \mathbf{v_s}) \longrightarrow f(\mathbf{v_d}, \mathbf{v_s})$ without $\mathbf{v_d'} \longrightarrow \mathbf{v_d}$, but since the function we are trying to learn is injective, we can try to get the controller to converge to the desired fixed point by setting the initial weights using least squares on an initial dataset. We generate this dataset by simulating randomly generated controls (within the restricted control ranges) on the single-DOF plant, and then pruning to remove trials in which the outcomes were far outside the desired velocity range or had come up against the joint limits. The effective range of the controller is estimated by the spread of the $\mathbf{v_p}$s produced in this dataset. The larger the initial dataset, the more representative this estimate will be of the true range of outcomes achievable with the allowed controls. If the joint runs up against the joint limits during on-line learning, a virtual error estimating the overshoot prevented by the joint limit is added to the position error, to facilitate the learning.

Our simulations use the SD/FAST software package [19] with a three-dimensional diver model generously shared with us by Jessica Hodgins (see [32]). Preliminary simulations on the single-DOF controllers are promising. Figure 6 shows the squared errors in the controls $\mathbf{u}$ and the plant output $\mathbf{v_p}$ for online training of a PHM controller for shoulder

abduction/adduction with $\mathbf{v_s} = (0, 0)$. The network has 739 basis functions with $\sigma = .06$. The basis functions are arranged in a grid-centered sphere packing so that the closest distance between basis function centers is $\sqrt{2}\sigma$, or 2σ along the grid axes. The network was initialized with a dataset containing 1439 elements after pruning. Figure 7 shows a typical movement produced by this controller.

As can be seen in Figure 6, the errors converge, but a rather large error in the final velocity remains. This error is smaller with larger networks on finer grids, but this brings us up against the curse of dimensionality common with locally-acting approximators: to double the number of basis functions along each dimension, we must increase the total number of basis functions in the controller by a factor of eight. We can also expect convergence to be slower with larger networks. The situation is still worse because of the two dimensions added by requiring a different set of weights for each $\mathbf{v_s}$. The computation and storage required for this scheme can quickly become huge. Global approximation methods such as neural networks do not suffer from the curse of dimensionality to such an extent as this, but they are generally trained with local gradient methods, which cannot guarantee a global solution. Techniques such as covariance reset, which revitalize the recursive least squares training algorithm, may improve the convergence, however, and possibly allow us to use smaller networks. It is clear that much remains to be done in terms of exploring systems such as these and investigating new controller designs.

4.2 Coordinating Controller

For the coordinating controller, a similar learning scheme may also be possible. However, a more fruitful approach may be that of reinforcement learning. Good surveys of reinforcement learning can be found in [26] and [22]. In reinforcement learning, the output error $||\mathbf{y_p} - \mathbf{y_d}||$ is minimized directly, and the learning can be distributed over sequences of actions. One variant which does not require a system model is Q-learning. A function Q is defined for each state $\mathbf{y} = (\mathbf{y_d}, \mathbf{y_s})$ and each control action $\mathbf{v}$ as

$$Q(\mathbf{y}, \mathbf{v}) = R(\mathbf{y}) + \max_{\mathbf{v}'} Q(\mathbf{y}', \mathbf{v}')$$

where $\mathbf{y}'$ is the successor state to $\mathbf{y}$ under $\mathbf{v}$ and R is the reward accrued in each state. For the diver problem, no reward is accrued until the end of the dive, when the diver reaches the water. Then the reward is based on the error between $\mathbf{y_p}$ and $\mathbf{y_d}$. The radial basis function network is required to learn an approximation to the Q function for the system, with the error, or temporal difference,

$$d = R(\mathbf{y}) + \max_{\mathbf{v}'} Q(\mathbf{y}', \mathbf{v}') - Q(\mathbf{y}, \mathbf{v})$$

being used at each transition for recursive least squares update of the network weights. Thus the values of the final states are propagated backward to earlier states. Such an approach, using dynamic programming ideas with a network approximation of the Q or value function, has been called neuro-dynamic programming [3]. A separate action selector would select the

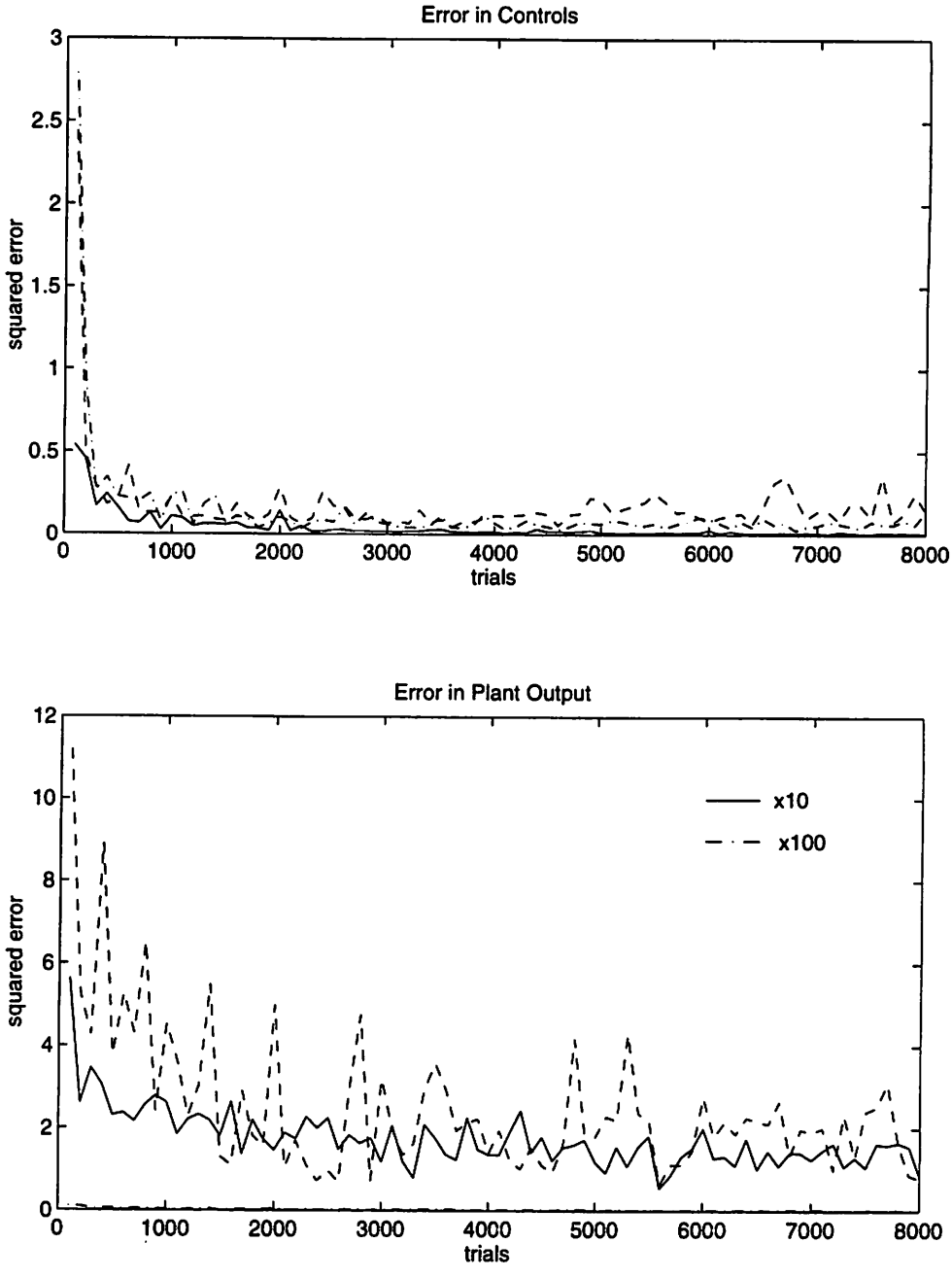

Figure 6: Squared errors averaged over groups of 100 trials for training a PHM controller for shoulder abduction/adduction. The top plot shows the squared errors in $\mathbf{u}$ before squashing (with $\beta = .8$). After squashing, each control is a scaled value in $(0, 1)$. The solid line represents pw, the dashed line ph_1, and the dash-dot line ph_2. The bottom plot shows the squared errors in $\mathbf{v_p}$; the solid line represents $\Delta\theta$ (squared error multiplied by a factor of ten for visibility), the dashed line $\Delta\dot{\theta}$, and the dash-dot line T (multiplied by 100). θ is measured in radians, $\dot{\theta}$ in radians per second, and T in seconds.

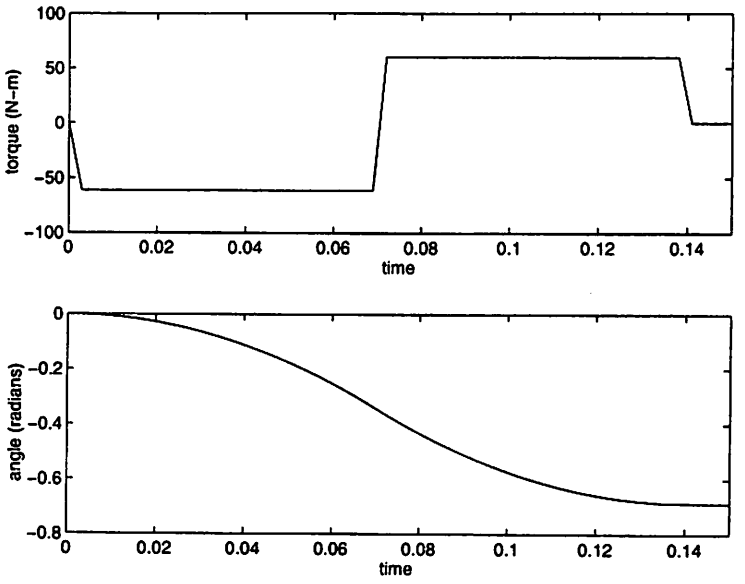

Figure 7: A movement produced by the trained PHM single-DOF controller. The input is $\Delta\theta_d = -\frac{\pi}{4}, \Delta\dot{\theta}_d = 0, T_d = .15$. $\dot{\theta}_0 = 0$, $\tau = 0$. The controls are $\text{ph}_1 = -61.365$, $\text{ph}_2 = 60.211$, $\text{pw} = .0697$, and the plant output is $\Delta\theta = -.687$, $\Delta\dot{\theta} = -.132$, $T = .15$.

next action taken by the controller based on a balance between maximizing the Q function over all the possible control actions (an optimal policy) and exploring the action space. Since the desired movement parametrization appears in the input to the controller, the same network can be trained to produce several different dives. We are currently implementing this reinforcement learning algorithm for training the coordinating controller.

For systems in which the state space is discrete, and the Q values can be stored in a table, if the states contain sufficient information that the system is Markov, then the Q-learning algorithm converges (see [1], [20], [29]). For a system like the diver, however, where we require a function approximator, convergence is more problematic. Boyan and Moore [4] give simple examples in which substituting a function approximator for a lookup table results in loss of convergence. These examples can be made to converge by changing slightly the methods used, however; see [22] for a summary. Certain types of function approximators have been shown to guarantee convergence when combined with dynamic programming techniques; in particular, neural network approximators may not converge, but certain linear interpolation approimators will [12], as will some feature-based methods (including radial basis function networks) satisfying certain properties, under a modified dynamic programming algorithm [30]. It is our hope that these results can be extended for our radial basis function networks with reinforcement learning algorithms similar to the Q-learning algorithm described above.

5 Conclusions

The hybrid, hierarchical, learning control structure biological motor control systems use to deal with system complexity and unknown models can provide inspiration for tackling difficult control problems. In designing a hybrid control structure, often the most critical pieces are the decoder and encoder. Biological systems suggest pattern generators as models for the decoder, and suggest using desired features of the movement (rather than a specific desired trajectory) to create the encoder. We have designed a learning control structure using these ideas and are testing it on the diving problem. The single-DOF controllers play the role of pattern generators in the controller, restricting the allowed torque profiles to a family of two-pulse controls. The coordinating controller provides the tuning inputs to the single-DOF controllers based on the type of dive that is desired. For the lower-level controllers, a modified form of supervised learning can be applied, but for the coordinating controller, reinforcement learning is more appropriate. We believe that new approaches such as the learning controller presented here will be essential to making headway on difficult behavioral control problems; much work remains to be done in this area.

6 Acknowledgments

We would like to thank Jessica Hodgins for providing us with the physical human model used in this work, and Stuart Russell and Ron Parr for their interest and helpful comments.

References

[1] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[2] C. Batterman. *The Techniques of Springboard Diving.* MIT Press, Cambridge, MA, 1968.

[3] D. P. Bertsekas and J. N. Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of the 34th Conference on Decision and Control*, pages 560–564, 1995.

[4] J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: safely approximating the value function. In *Advances in Neural Information Processing 7*, pages 369–376, 1995.

[5] R. Brockett. On the computer control of movement. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1988.

[6] R. W. Brockett. Systems theory on group manifolds and coset spaces. *SIAM Journal of Control*, 10(2):265–284, 1972.

[7] R. W. Brockett. Analog and digital computing. In *Future Tendencies in Computer Science, Control and Applied Mathematics. International Conference on the Occasion of the 25th Anniversary of INRIA, Proceedings*, pages 279–289, 1992.

[8] L. S. Crawford and S. S. Sastry. Biological motor approaches for a planar diver. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 3881–3886, December 1995.

[9] C. Frohlich. Do springboard divers violate angular momentum conservation? *American Journal of Physics*, 47(7):583–592, July 1979.

[10] P. Di Giamberardino, S. Monaco, and D. Normand-Cyrot. Digital control through finite feedback discretizability. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3141–3146, 1996.

[11] J-M. Godhavn, A. Balluchi, L. S. Crawford, and S. S. Sastry. Path planning for nonholonomic systems with drift. Submitted to the 1997 American Control Conference., 1996.

[12] G. J. Gordon. Stable function approximation in dynamic programming. CMU-CS-95-103, Carnegie Mellon University, 1995.

[13] D. Gorinevsky, A. Kapitanovsky, and A. Goldenberg. Radial basis function network architecture for nonholonomic motion planning and control of free-flying manipulators. *IEEE Transactions on Robotics and Automation*, 12(3), June 1996.

[14] G. L. Gottlieb, D. M. Corcos, and G. C. Agarwal. Strategies for the control of voluntary movements with one mechanical degree of freedom. *Behavioral and Brain Sciences*, 12:189–210, 1989.

[15] S. Grillner. Locomotion in vertebrates: central mechanisms and reflex interaction. *Physiological Reviews*, 55(2):247–304, April 1975.

[16] S. S. Haykin. *Neural Networks: A Comprehensive Foundation.* MacMillan, New York, 1994.

[17] J. Hertz, Anders Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley Publishing Company, Redwood City, California, 1991.

[18] J. K. Hodgins and M. H. Raibert. Biped gymnastics. *International Journal of Robotics Research*, 9(2):115–132, April 1990.

[19] M. G. Hollars, D. E. Rosenthal, and M. A. Sherman. *SD/FAST user's manual.* Mountain View, CA, 1991.

[20] T. Jaakkola, M. I. Jordan, and S. P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201, 1994.

[21] M. Jeannerod. *The Neural and Behavioral Organization of Goal-Directed Movements.* Clarendon Press, Oxford, 1988.

[22] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[23] I. Kolmanovsky and N. H. McClamroch. Developments in nonholonomic control problems. *IEEE Control Systems*, 15(6):20–36, December 1995.

[24] I. V. Kolmanovsky, N. H. McClamroch, and V. T. Coppola. Controllability of a class of nonlinear systems with drift. In *Proceedings of the 33rd Conference on Decision and Control*, pages 1254–1255, 1994.

[25] A. C. Pil and H. Asada. Recursive experimental structure re-design of a robot arm using rapid prototyping. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 2, pages 1094–1099, 1994.

[26] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Englewood Cliffs, NJ, 1995.

[27] R. A. Schmidt. A schema theory of discrete motor skill learning. *Psychological Review*, 82(4):225–260, 1975.

[28] Schöner and Kelso. A synergetic theory of environmentally-specified and learned patterns of movement coordination: I. relative phase dynamics. *Biological Cybernetics*, 58:71–80, 1988.

[29] J. N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202, 1994.

[30] J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94, 1996.

[31] B. Vereijken, H. T. A. Whiting, and Beek. A dynamical systems approach to skill acquisition. *Quarterly Journal of Experimental Psychology Section A - Human Experimental Psychology*, 45(2):323–344, August 1992.

[32] W. L. Wooten and J. K. Hodgins. Animation of human diving. *Computer Graphics Forum*, 15(1):3–13, March 1996.