controller had maintained control for 0.25 hours, the system was pushed with a 20 newton force. The force continued until the reflexive stabilizer was needed. Each force was applied in a randomly selected direction. This is shown in Figure 7.6.

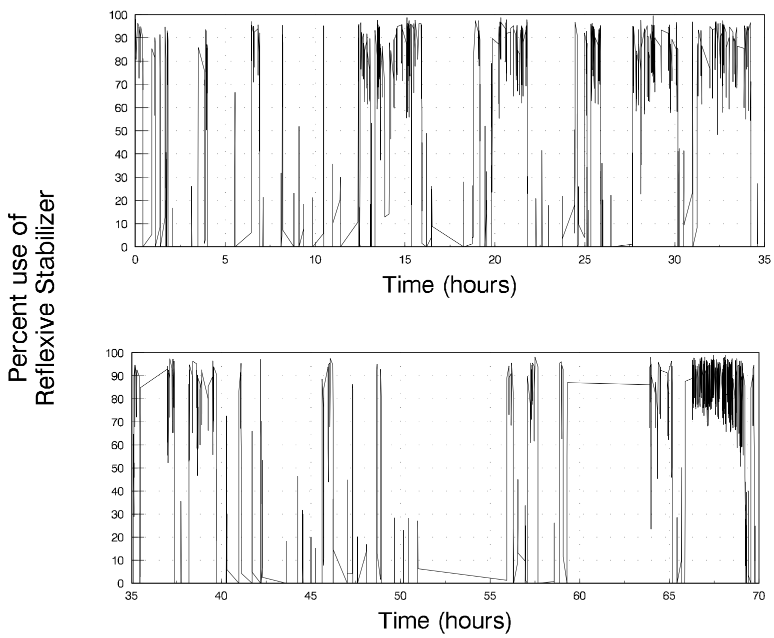

Figure 7.6: Extended simulation of reflexive stabilizer training. (Forced failure after 0.25 hours of continuous neural network control.) [** figure simplified for postscript printing **]

In Figure 7.6, the many spikes are the disturbances. The height of each spike indicates how much the reflexive stabilizer was needed to maintain the system. When the reflexive stabilizer is used for long bursts, the neural network controller is training. Figure 7.6 suggests that the network is performing poorly, but in spite of the forced failures the neural network maintains control for 56 percent of the time. Also, most of the forced failures are quickly recaptured by the neural network controller. Nevertheless, the neural network controller was not fully robust.

The network learns a subset of the entire control space. A heavy disturbance can move the system outside the neural network controller's subset of knowledge. This is not unexpected since the neural network was trained with examples from a small subset of the entire operating domain. The forced failures, in reality, test the ability of the controller to generalize.

Although the input space of the neural network controller is four dimensional, a reduced dimension scatter plot was produced to study the neural network controller's domain of control. Figure 7.7 plots the percent use of the reflexive stabilizer against the cart/pole state space position.

The solid patch on the lower plane is the neural network's region of control. The region is very small, but it contains more points than does the rest of the graph. This is expected since the neural network controller is to hold the cart/pole system near the center of the patch. The velocity effects

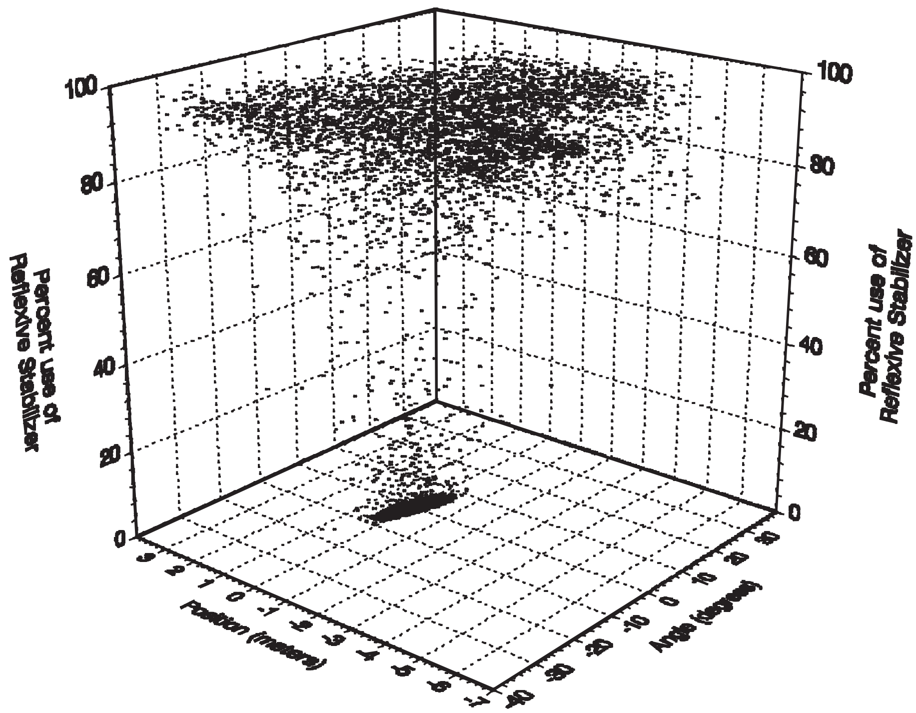

Figure 7.7: Controller operation from 33.3 hrs to 75 hrs.

of the cart/pole system are not directly plotted on Figure 7.7, so reading too deeply into the plot is not wise. In general, the reflexive stabilizer is used more as the conditions get farther from the patch. This is indicated by the inverted pyramid shape of the points in Figure 7.7.

The generalization capability is important for a controller if it is required to maintain stability in the face of large disturbances, but for this control architecture the reflexive stabilizer provides assurance of stability. Therefore, the neural network controller provides good control under normal conditions. The reflexive stabilizer assures the stability of the plant.

The second test evaluated the controller under more normal conditions. The test operated the neural network controller without the regular interruptions of neural network control. The 5 degree random noise was maintained, but the forced 900 second failures were disabled. Under these conditions, the neural network performed without failure for 30 hours of simulated time.

The main result of the extended tests was that the neural network controller can operate with knowledge of only a small subset of the state space. Due to limited knowledge in the neural network, the reflexive stabilizer is necessary to assure plant stability. The unusual neural network learning curve was a result of the network incompletely learning the system. Thus, the non-monotonic learning results from the neural network moving the hidden units around to characterize the system properly. The neural network, however, can control the system for extended intervals with only limited knowledge.

7.3.2 Control Performance

The global operation of the controller is important from the aspects of stability and robustness, but more important is the way in which the system is controlled. Figures 7.8 and 7.9 illustrate neural network control of the cart/pole system. The controller maintains good control of both the cart and

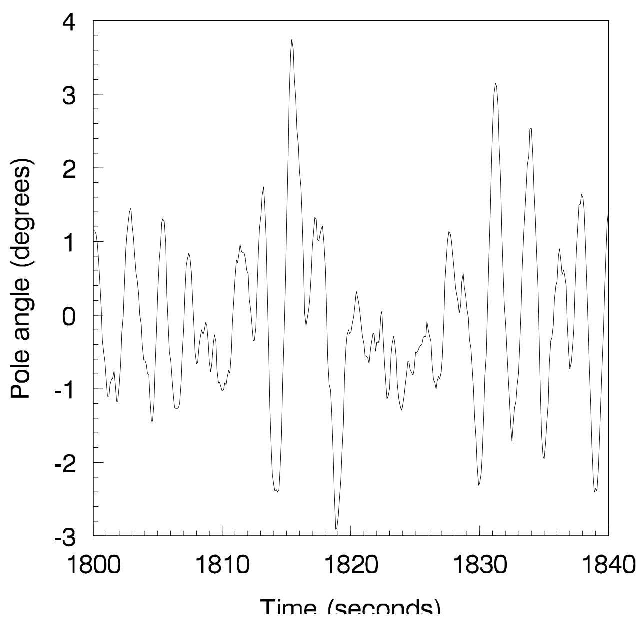

Figure 7.8: Neural network control of pole angle.

the pole in spite of the 5 degree random pole disturbances at each integration step. Figures 7.8 and 7.9 demonstrate the ability of the neural network controller to handle external disturbances, but do not indicate the steady behavior of the controller. To study the neural network's steady state

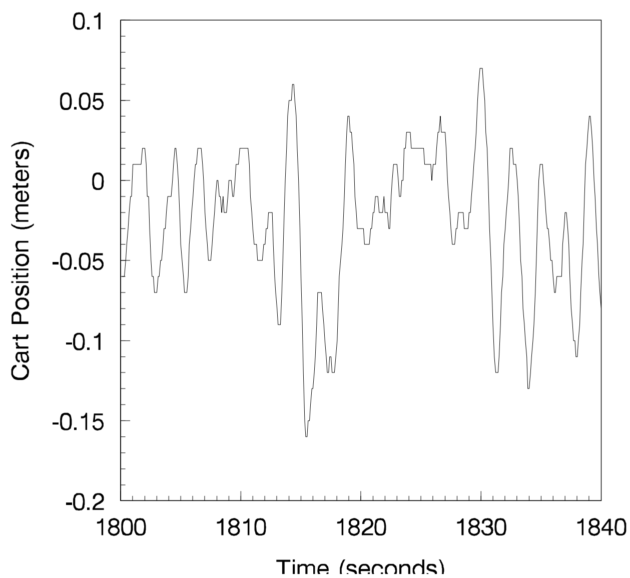

Figure 7.9: Neural network control of cart position.

behavior, the network was trained as previously, but 1820 seconds into the simulation the random disturbance was switched off. The results of this test are shown in Figures 7.10 and 7.11.

Once the noise is disabled, the cart position rapidly comes to rest with a small offset of several centimeters from center. The pole angle, which must vary to center the cart, requires a slightly longer time to damp to zero

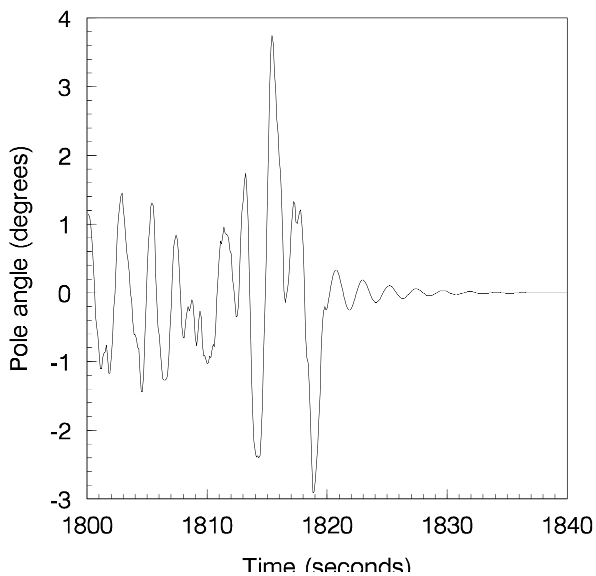

Figure 7.10: Neural network control of pole angle. (noise disabled at 1820 sec)

than does the cart. Both the pole and the cart come to a complete stop with no limit cycle motion.

The reflexive stabilizer and distal-error training algorithm incorporate two features necessary to independent long term on-line learning: learning regulation and stability assurance. The regulation of learning is important because the network spends most of the time near the setpoint. The

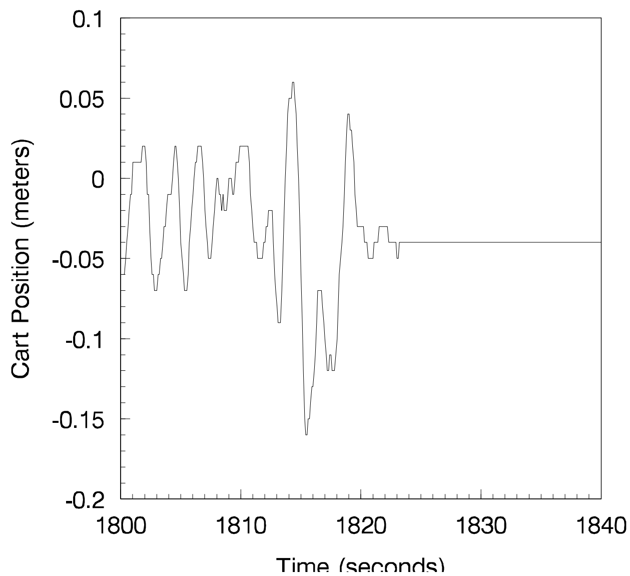

Figure 7.11: Neural network control of cart position. (noise disabled at 1820 sec)

network, however, learns only when a reflexive stabilizer boundary is violated. Thus, the network learning is shut off during long periods when the system is at steady state. Without this control over learning, the data learned during steady state operation would focus all the network resources on that microcosm of space and forget information from other regions. In other words, the training technique prevents over training of the neural network.

The second feature is assurance of stability, which is independent of the knowledge in the learning controller. Figures 7.7, 7.10, and 7.11 indicate that good control is possible with only highly local knowledge. Since the neural network has limited knowledge, a reflexive stabilizer is critical for stability. Without the stabilizer, the system could move beyond the neural network's knowledge and system failure would be the most likely result.

Although there were good results from training the neural network with a reflexive stabilizer, a setpoint input is missing. The setpoint is inferred as the point in the center of the reflexive stabilizer boundaries. That point may not be the desired setpoint and inference is a weak means for setting the setpoint. A way to pull the system to a specific position is needed. The work to accomplish this is the subject of the next section.

This section completes the third objective, which is training a neural network while a system is actively Lyapunov bound. The Lyapunov bound, formed by the reflexive stabilizer, was shown to be necessary to assure failure free operation. The distal-error training algorithm also was demonstrated.

7.4 Attractor Construction

To construct the fixed point attractor, all four components of the control architecture are used. The primary components responsible for the attractor construction are the path assessor and the search generator. To build the attractor, the system dynamics are augmented by the neural network and reflexive stabilizer dynamics. The reflexive stabilizer cancels the natural system attractor and the neural network controller learns to build an attractor at the setpoint.

To study attractor construction, the underpowered pendulum system was used. Two tests were conducted on the system. The first test was to build a fixed point attractor at the vertically up position and the second test was to build a fixed point attractor 30 degrees off vertical at 60 degrees.

The reflexive stabilizer was used as described in section 7.2 to swing the pendulum into the controllable range of the neural network controller. The search generator was set to generate gaussian noise with a variance of 3.5 newtons. The path assessor setpoint was at zero velocity and depending on the test, the angle was set at 60 or 90 degrees.

7.4.1 Building an Attractor at 90 degrees

The first tests built the pendulum attractor at an angle of 90 degrees. To balance the pendulum, the neural network used the control illustrated in Figure 7.12. The control appears simple, but it is finely tuned. The control action rapidly brings the pendulum to a stop, and then it uses the exploration

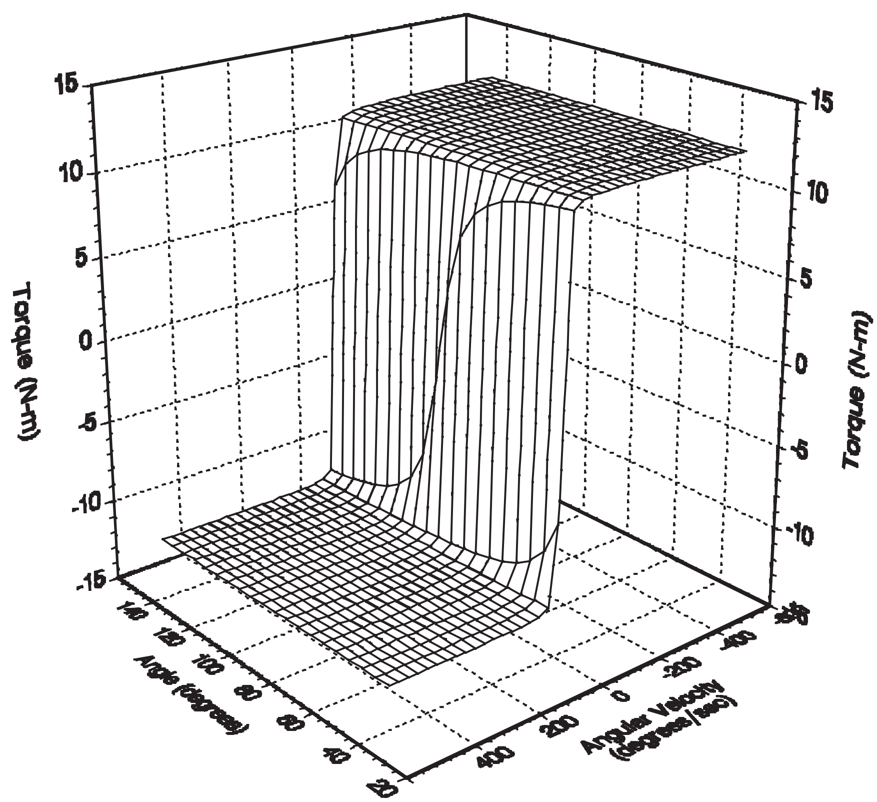

Figure 7.12: Neural network control force for vertical pendulum balancing.

noise and the learning algorithm to slowly move the pendulum to the vertical position. The most probable reason for this type of control is that the angular velocity is more closely controlled by the input torque than is the angular position. The angular velocity is one integration away from the control torque while the angular position is two integrations removed. Each integration blurs the information, making it difficult for the network to form correlations between torque and position. The information is blurred because the integrator acts like a moving average filter. Thus, the integrator behaves as a low pass filter.

The low dimensionality of the system contributes to the emphasis on minimum velocity. The result is that once the angular velocity stops, the position vector cannot be reduced because the pendulum is not moving. This tendency to move to zero angular velocity is illustrated in Figure 7.13. The angular velocity drops to zero and then the search generator pushes the system off zero velocity. Each little excursion off zero velocity is a training case for the network. Over time the network learns the good paths and ignores the bad paths. The result after training for 3-5 simulated hours is the control action shown in Figure 7.12. This control force generates the flow graph shown in Figure 7.14. The network catches the pendulum on the

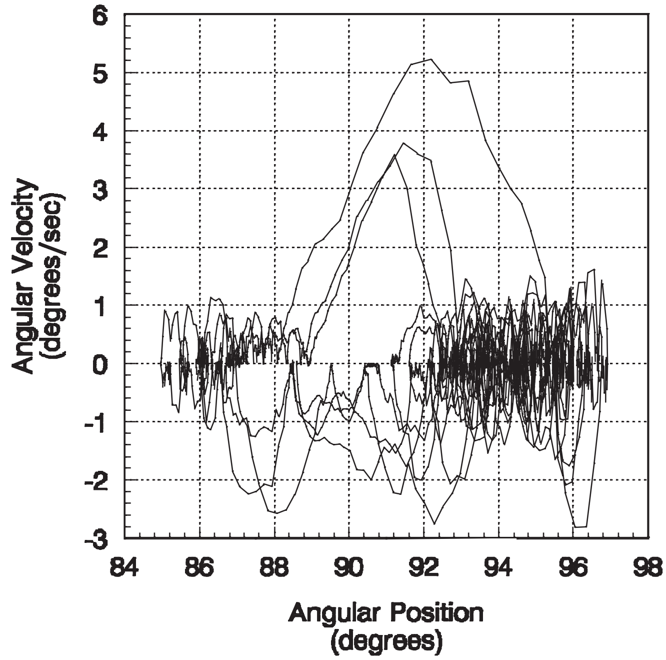

Figure 7.13: Phase plot during network training.

way toward the center and then quickly moves the system to zero angular velocity. Because the reflexive stabilizer controls the system outside the range of the neural network controller, the pendulum always enters the neural network control domain at approximately 150 degrees/second. As a result, paths entering the domain at lesser velocities are not encountered and the network does not learn to direct their motion.

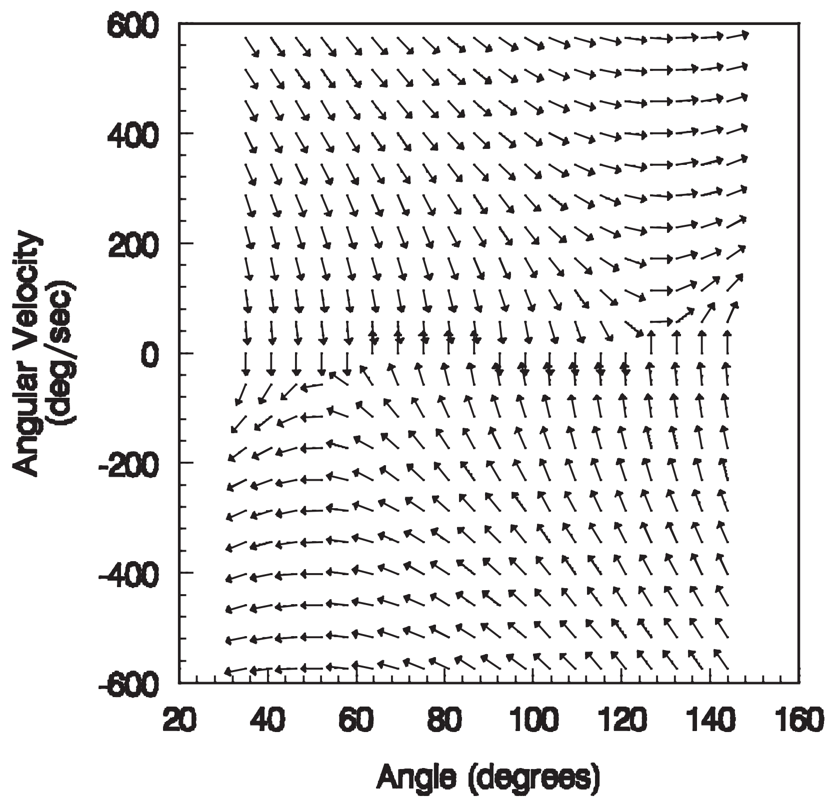

Figure 7.14: Vector flow graph of underpowered pendulum under control of neural network (vertical balancing).

7.4.2 Building an Attractor at 60 degrees

The second test, to build an attractor at 60 degrees, yields similar results to the 90 degree test. The force plot for 60 degrees is shown in Figure 7.15. As expected, the 60 degree flow graph, Figure 7.16, is similar

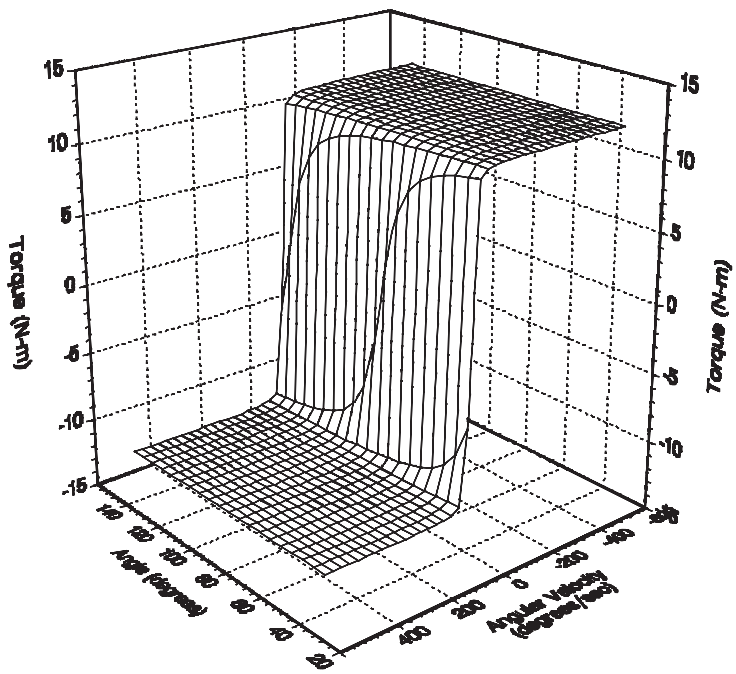

Figure 7.15: Neural network control force for 60 degree pendulum balancing.

to the vertical test graph, but the force field is shifted toward 60 degrees. Although there is a shift in the flow field, the neural network control still has the same strong tendency to zero the angular velocity. The motion to the setpoint is still the result of the training algorithm. Due to the coarseness of the flow vectors, Figure 7.16 does not show the fixed point at the position of 60 degrees. The control actions of both the vertical and 60 degree tests are similar because the main control action is to damp the pendulum to a stop.

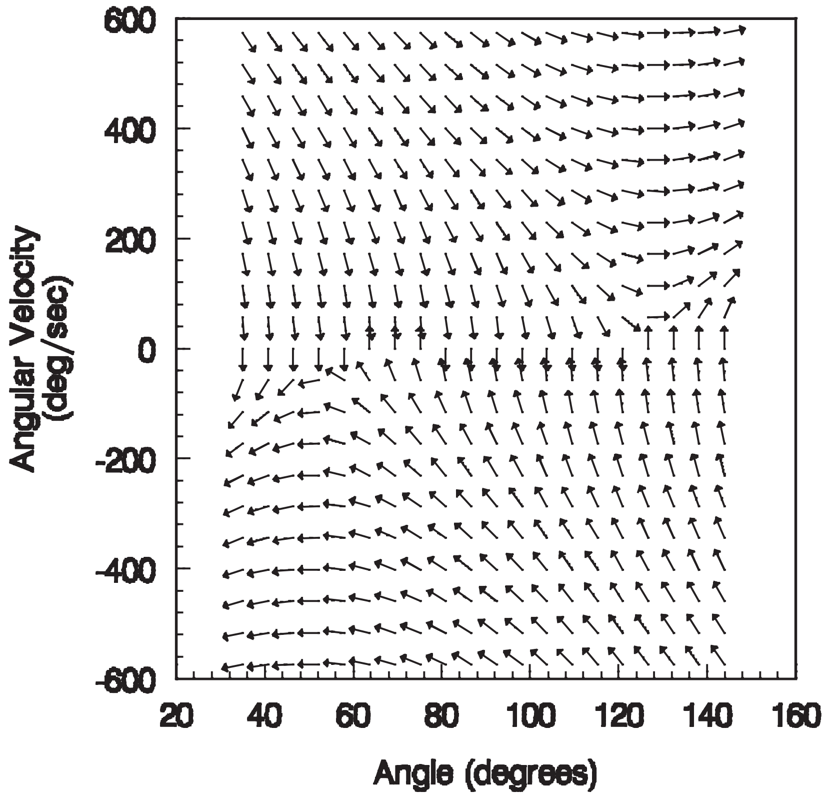

Figure 7.16: Vector flow graph of underpowered pendulum under control of neural network (60 degree balancing).

In both tests, the setpoint is not reached by learned control action; it is reached by the action of the training algorithm.

As in the previous section, the global ability of the controller is important, but the dynamics of the controlled system are the primary interest.

Figure 7.17 illustrates the underpowered pendulum under neural network control.

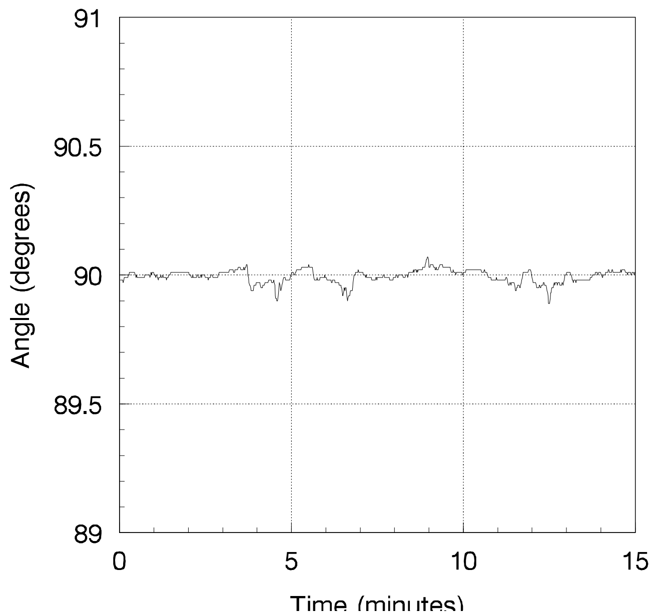

Figure 7.17: Neural network control of the underpowered pendulum balanced at 90 degrees.

For the neural network to settle in to the control shown in Figure 7.17, it is necessary to reduce the perturbation from the search generator. Initially, the search generator was generating gaussian noise with a variance of 3.5 N-m. A smaller variance may be selected, but the learning is greatly slowed. Therefore, the search generator was modified so that noise was reduced as the setpoint was reached. This was done by multiplying the 3.5 N-m variation by the distance in state space between the present position and the setpoint. To prevent too large a variance far from the setpoint, the correction was limited to operate within a radius of one from the setpoint. This modification provided a smooth shutoff of the search generator.

The modification, however, caused a small problem with the 60 degree test. With the noise reduced within a phase space radius of one, the learning of the control was very slow and resulted in a final offset of several degrees from the setpoint of 60 degrees. By reducing the squelching radius to a radius of 1/5, the network trained more rapidly and the offset was reduced to approximately 0.6 degrees. Figure 7.18 illustrates balancing the pendulum at 60 degrees. The 60 degree balancing case was more difficult for the controller to learn and required more learning time. Figure 7.18 suggests that the control is slowly moving toward the setpoint, but it is only a slow drift. The pendulum will drift back and forth as much as 0.1 of a degree.

The tests of directly constructing an attractor at a setpoint were successful. This method does not require a system model, and it uses the on-line data to provide control without manual adjustment. The section demonstrated the first objective, which was to demonstrate control by attractor construction. Fixed point attractors were constructed to balance the pole vertically and 30 degrees off vertical.

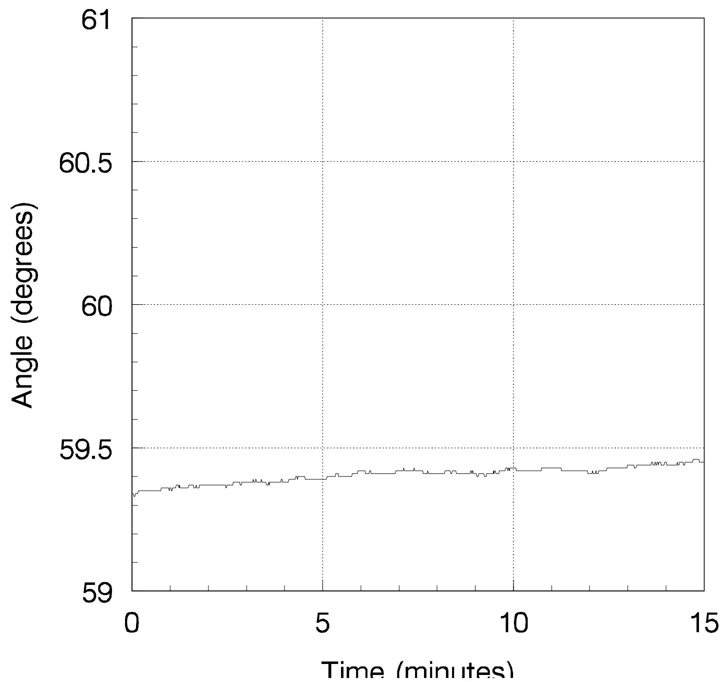

Figure 7.18: Balancing the underpowered pendulum at an angle of 60 degrees.

This chapter presented the results of the simulation tests. The reflexive stabilizer was demonstrated both as an assistant controller and as a system stabilizer. Neural network training with the distal-error learning algorithm was investigated. Attractor construction tests were also presented.

CHAPTER VIII

CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions

The primary thrust of this research was to demonstrate practical techniques for nonlinear adaptive control. The main issues of the study were learning methods, plant stability, and model free nonlinear control via direct attractor construction. The primary results are summarized below:

1). Adaptive nonlinear control by directly constructing the desired attractor was demonstrated. The technique is general in nature and does not require simplifying assumptions or linearization of the system. This is a method for controlling any nonlinear system without drawing knowledge from a system model. The direct attractor construction was demonstrated using the underpowered pendulum.

2). A new reinforcement training algorithm was developed and demonstrated. It is a constant memory algorithm and does not require increasing memory for increasing sequence length. New mathematical operators were developed and rules of operation were proven. The new mathematics was demonstrated by developing the reinforcement training algorithm. The training algorithm was tested by learning to control the cart/pole system.

3). Learning to control an unstable system while it is under active Lyapunov stabilization was demonstrated. It was shown to be a key element for practical application of the neural network controller, because generally the knowledge in a neural network controller is unknown and non-stationary. The active Lyapunov stabilization was demonstrated by constructing a reflexive stabilizer from the plant design model. Reflexive stabilizers were constructed for both the underpowered pendulum and the cart/pole system.

Other significant results include: a demonstration of a reflexive stabilizer to assure stability without restrictive assumptions, training a neural network with a reflexive stabilizer, methods for controlling on-line learning, and studies of state space exploration. Demonstrated were techniques for practical control of unstable nonlinear systems.

These results demonstrate that a reflexive stabilizer and an artificial neural network can ensure system stability while providing adaptive control of difficult nonlinear systems without fitting a system model. The control architecture incorporates significant knowledge from the plant design models without requiring the models to be highly accurate. Thus, the architecture builds on the knowledge from the design process while using the learning components to solve the difficult fine tuning of the controller. The artificial neural network control techniques are methods for directly approaching the problems of nonlinearity and stability in control applications.

8.2 Recommendations

Overall, the work is a scratch on the surface of the possibilities for the demonstrated techniques. Below are some suggestions for further research.

This study focused on the use of an artificial neural network for the adaptive controller. This, however, is only one method for learning to control a system. Other nonlinear learning controllers could be coupled with the other components of the control architecture.

The new mathematics demonstrated should be applied to recurrent neural networks. The memory requirements will be lower than the feedforward network case and it offers a solution to the inter-sequence updating problem.

Direct attractor construction and use of a reflexive stabilizer for multiple input/output systems should be explored. Application to time delayed systems and actual systems also should be studied.

Methods for shaping the control attractor are needed. For example, techniques for producing a minimum time or minimum energy attractor would be highly useful. Also needed are methods for characterizing and measuring the volume of influence for the constructed n-dimensional attractors.

All the systems considered in this dissertation had the desired control setpoint within the reflexive stabilizer boundaries. Learning on the boundaries of constraints should be considered, since many optimal operating conditions lie against constraints.

The state space exploration is one area where more study is needed. The methods demonstrated here are sufficient but are not ideal. Schmidhuber (1991) and Thrun *et al.,* (1991), have investigated the use of plant models to direct the exploration in state space. This and other methods of state space exploration greatly warrant further study.

APPENDIX A

FUNDAMENTAL PROOFS

A.1 Preliminaries

The purpose of this appendix is to introduce the new mathematical operators and prove necessary theorems and rules of operation. Each new operator is illustrated and their algebraic rules of operation are proven. Also included is a proof of Theorem 1 and Theorem 2 which are required to develop the distal-error training algorithm. Section 5.5 contains a similar presentation of the notation and operators, but they are presented again here to aid the reader while reading the proofs and theorems.

The new operators are designed to manipulate multi-dimensional arrays. To indicate the dimensions of the multi-dimensional arrays, the notation is to subscript each variable with indices to each dimension. For example, the row vector $\mathbf{P}$ in this notation is $\mathbf{P}_{1q}$. To avoid confusion, initial subscripts on variables are promoted to regular size and are appended to the variable name. Thus, $\mathbf{W}_2$ becomes $\mathbf{W2}$.

A.2 Multi-Dimensional Operators

The analysis also requires three new operators to operate on the multi-dimensional arrays. The first operator is an element multiply and is indicated by:

$$C_{nmq} = A_{nmq} \odot B_{nmq}, \tag{A.1}$$

where in scalar notation:

$$c_{ijk} = a_{ijk} b_{ijk} \qquad \forall \quad \begin{aligned} i &= 1 \dots m \\ j &= 1 \dots n \\ k &= 1 \dots q. \end{aligned} \tag{A.2}$$

The element multiply operator was defined in a three dimensional space, but it also applies to larger and smaller spaces.

The next two operators are actually the result of splitting a regular matrix multiply into two separate operations. Thus, a matrix multiply is indicated by:

$$D_{mq} = A_{mn} B_{nq} = \langle A \otimes B \rangle_{mnq}^{n}, \tag{A.3}$$

where the partial multiply operator:

$$C_{mnq} = A_{mn} \otimes B_{nq}, \tag{A.4}$$

is defined in scalar notation as:

$$c_{ijk} = a_{ij} b_{jk} \qquad \forall \quad \begin{aligned} i &= 1 \ldots m \\ j &= 1 \ldots n \\ k &= 1 \ldots q, \end{aligned} \tag{A.5}$$

and the compaction operator,

$$D_{mq} = \langle C_{mnq} \rangle^{n} = \langle C \rangle^{n}_{mnq}, \tag{A.6}$$

is defined in scalar notation as

$$d_{ik} = \sum_{j=1}^{n} c_{ijk} \qquad \forall \quad \begin{aligned} i &= 1 \ldots m \\ k &= 1 \ldots q. \end{aligned} \tag{A.7}$$

The subscripts of the compaction operator indicate the number and size of each dimension of the multi-dimensional structure. The superscript indicates the dimensions over which the compaction sum is to be performed.

Each partial multiply increases the dimension of the resulting structure by one. Thus, three matrix multiplications yield a four dimensional structure:

$$A_{mn} B_{nq} C_{qr} = \langle A \otimes B \otimes C \rangle^{nq}_{mnqr} = \langle D \rangle^{nq}_{mnqr} = E_{mr}. \tag{A.8}$$

A.3 Element Multiply Algebra

For the element multiply operator to be used in calculations, algebraic laws are needed. The first of these is associativity of scalar multiplication:

$$c(A \odot B) = (cA) \odot B = A \odot (cB)\,. \qquad \textbf{(A.9)}$$

Equation A.9 can be shown to be true by writing it in scalar notation:

$$c(a_{ij}b_{ij}) = (ca_{ij})\,b_{ij} = a_{ij}(cb_{ij}) \quad \forall\ i = 1\ \dots\ n,\ j = 1\ \dots\ m\,. \qquad \textbf{(A.10)}$$

The operator also satisfies the associative law:

$$C \odot (A \odot B) = (C \odot A) \odot B\,, \qquad \textbf{(A.11)}$$

$$c_{ij}(a_{ij}b_{ij}) = (c_{ij}a_{ij})\,b_{ij} \quad \forall\ i = 1\ \dots\ n \ \ \text{and} \ \ j = 1\ \dots\ m\,. \qquad \textbf{(A.12)}$$

The element multiply operator satisfies the commutative law:

$$A \odot B = B \odot A\,, \qquad \textbf{(A.13)}$$

$$a_{ij}b_{ij} = b_{ij}a_{ij} \qquad \forall\ i=1\ \dots\ n \quad and \quad j=1\ \dots\ m. \tag{A.14}$$

The final property, the distributive law, also is satisfied:

$$C\odot(A+B) = (C\odot A)+(C\odot B), \tag{A.15}$$

$$c_{ij}(a_{ij}+b_{ij}) = (c_{ij}a_{ij})+(c_{ij}b_{ij}) \qquad \forall\ i=1\ \dots\ n, \quad j=1\ \dots\ m. \tag{A.16}$$

All the basic algebraic laws were demonstrated and proven in a two dimensional space for clarity. The algebraic rules, however, also apply to higher dimensional spaces. The proofs simply require more indices.

A.4 Partial Multiply and Compaction Algebra

The partial multiply and compaction operator as an extension of the common matrix multiply are bound by the same algebraic laws as the common matrix multiply. These rules are associativity of scalar multiplication, distributive law for postmultiplication and premultiplication, and associative law of matrix multiplication. The reason this is true is the partial multiply and compaction operators simply move the summation of the elements to the beginning of the expression, which does not affect the result. In fact, Wylie and Barrett (1982) prove all the above algebraic rules for standard matrix multiplication with the summation moved to the beginning of the expression.

To develop the distal-error training algorithm, however, an operational rule for combining the element multiply with the partial multiply must be demonstrated. The rule concerns application of the partial multiply operator to equations which contain an element multiply. For example, consider the equation:

$$E_{mp} = A_{mp} \odot \left[B_{mn} C_{np} \right]. \qquad \textbf{(A.17)}$$

The goal is to encapsulate the entire right hand side of the equation inside a compaction operator, but first consider postponement of the matrix multiply only:

$$E_{mp} = A_{mp} \odot \left\langle B_{mn} \otimes C_{np} \right\rangle. \qquad \textbf{(A.18)}$$

By applying the partial multiply equation A.18 can be written as:

$$E_{mp} = A_{mp} \odot \left\langle D_{mnp} \right\rangle. \qquad \textbf{(A.19)}$$

where,

$$D_{mnp} = B_{mn} \otimes C_{np} . \qquad \textbf{(A.20)}$$

But, to move the compaction operator across the element multiply, the matrix, **A**, must be expanded to more dimensions. This done by simply replicating the matrix to extra dimensions. In scalar notation this is:

$$\hat{a}_{ijk} = a_{ik} 1_j \qquad \forall \quad \begin{matrix} i = 1 \dots m \\ j = 1 \dots n \\ k = 1 \dots p . \end{matrix} \qquad \textbf{(A.21)}$$

This expansion is indicated by a placing a caret over the original vector to illustrate that it is expanded. Thus, by applying this expansion the compaction operator can encapsulate the entire equation:

$$E_{mp} = \left\langle \hat{A}_{mnp} \odot D_{mnp} \right\rangle = \left\langle \hat{A} \odot D \right\rangle_{mnp}^{n} . \qquad \textbf{(A.22)}$$

A.5 Theorem 1

Theorem 1 proves the distributive law for the partial and element multiply.

THEOREM 1: Let **A**, **B**, and **C** be two dimensional matrices, then

$$\langle A \otimes (B \odot C) \rangle = \langle (A \otimes B) \odot \langle \bar{1} \otimes C \rangle \rangle . \quad \textbf{(A.23)}$$

PROOF: First, let

$$E_{mp} = \langle A_{mn} \otimes (B_{np} \odot C_{np}) \rangle , \quad \textbf{(A.24)}$$

then write equation A.24 in scalar notation:

$$e_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk} c_{jk} \qquad \forall \quad \begin{array}{l} i = 1 \dots m \\ k = 1 \dots p . \end{array} \quad \textbf{(A.25)}$$

Multiply equation A.25 by a matrix where all elements are one and group the factors:

$$e_{ik} = \sum_{j=1}^{n} (a_{ij} b_{jk}) (1_{ij} c_{jk}) \qquad \forall \quad \begin{array}{l} i = 1 \dots m \\ k = 1 \dots p . \end{array} \quad \textbf{(A.26)}$$

Finally, transform equation, A.26, back to multi-dimensional notation:

$$E_{mp} = \langle (A \otimes B) \odot (\overline{1} \otimes C) \rangle . \qquad \textbf{(A.27)}$$

Therefore,

$$\langle A \otimes (B \odot C) \rangle = \langle (A \otimes B) \odot (\overline{1} \otimes C) \rangle . \quad \blacksquare \qquad \textbf{(A.23)}$$

An important corollary to this result is that the compaction operator does not satisfy the distributive law.

COROLLARY: Let **A**, **B**, and **C** be two dimensional matrices, then

$$\langle A \otimes (B \odot C) \rangle \neq \langle A \otimes B \rangle \odot \langle \overline{1} \otimes C \rangle . \qquad \textbf{(A.28)}$$

PROOF: First, let

$$E_{mp} = \langle A_{mn} \otimes (B_{np} \odot C_{np}) \rangle , \qquad \textbf{(A.24)}$$

as shown in the previous proof, equation A.24 can be written as:

$$e_{ik} = \sum_{j=1}^{n} (a_{ij} b_{jk}) (1_{ij} c_{jk}) \qquad \forall \quad \begin{matrix} i = 1 \ldots m \\ k = 1 \ldots p. \end{matrix} \tag{A.26}$$

The summation, however, cannot be distributed:

$$e_{ik} = \sum_{j=1}^{n} (a_{ij} b_{jk}) (1_{ij} c_{jk}) \neq \sum_{j=1}^{n} (a_{ij} b_{jk}) \sum_{j=1}^{n} (1_{ij} c_{jk}) \qquad \forall \quad \begin{matrix} i = 1 \ldots m, \\ k = 1 \ldots p. \end{matrix} \tag{A.29}$$

Therefore,

$$\langle A \otimes (B \odot C) \rangle \neq \langle A \otimes B \rangle \odot \langle \bar{1} \otimes C \rangle . \qquad \blacksquare \tag{A.28}$$

The corollary highlights the importance of performing all interior operations prior to contraction. This corollary was the initial motivation for the higher dimensional mathematics since the normal matrix multiply results in the same problem as the corollary.

A.6 Theorem 2

The restriction of the above corollary, however, does not apply if the inputs are vectors because the summation (equation A.29) over j is trivial and can be eliminated.

THEOREM 2: Let **A** be a m x 1 column vector and let **B** and **C** be 1 x p row vectors, then

$$\vec{A}\,(\,\vec{B} \odot \vec{C}\,) = (\,\vec{A}\,\vec{B}\,) \odot (\,\vec{1}\,\vec{C}\,)\,. \tag{A.30}$$

PROOF: Let

$$E = \vec{A}\,(\,\vec{B} \odot \vec{C}\,)\,. \tag{A.31}$$

In scalar notation, equation A.31 can be written as:

$$e_{ik} = \sum_{j=1}^{1} a_{ij}\,b_{jk}\,c_{jk} \qquad \forall \quad \begin{array}{l} i = 1 \ldots m \\ k = 1 \ldots p\,. \end{array} \tag{A.32}$$

Noting that the summation can be eliminated because it sums from one to one and multiplying by a vector that contains one as all elements, equation A.32 can be written as follows:

$$e_{ik} = a_{ij}\,b_{jk}\,c_{jk} = (a_{ij}\,b_{jk})\,(1_{ij}\,c_{jk}) \qquad \forall \quad \begin{array}{l} i = 1 \ldots m, \\ j = 1\,, \\ k = 1 \ldots p, \end{array} \tag{A.33}$$

which in vector notation is:

$$\vec{E} = (\vec{A}\vec{B}) \odot (\vec{1}\,\vec{C}). \tag{A.34}$$

Therefore,

$$\vec{A}(\vec{B} \odot \vec{C}) = (\vec{A}\vec{B}) \odot (\vec{1}\,\vec{C}). \quad \blacksquare \tag{A.30}$$

GLOSSARY

As in any area of specification, specific terms are used to communicate concisely and precisely ideas among researchers in the field. This section presents the control and dynamic systems terms that are important to this dissertation. The dynamics systems terms are presented first, followed by the control terms.

G.1 Physical System Terms

attractor: If undisturbed trajectories of a dynamic system approach a common subset of state space, the subset of state space is called an attractor. See Section 2.2.

chaos: Chaos refers to a particular phenomena, found only in a nonlinear system, where a deterministic system's dynamics appear random. The term is also used to describe the science of studying chaotic behavior in systems.

phase space: Phase space is a general term for a space, which is the same dimension as the state space. Phase space is commonly used in the context of a phase space plot, which is a plot of dynamic system through state space. Other variables like time delayed or derivatives of state variables also may be used in a phase space plot. Thus, phase space is a space constructed of states, delayed states, state derivatives or other transformed state variables.

state: The system state or state of the system describes the conditions of a system at instant in time. States are the conditions of a system which change with time. For example, the states of a pendulum are its position and its velocity. The mass of the pendulum and the length of the pole are not considered states, because they do not vary with time.

state variables: A state variable is a mathematical variable, which represents the condition of a system state. Generally, a system is mathematical represented by a set of state variables and a set of differential equations that describe the interaction between them.

state vector: A state vector is the set of all the state variables, which define the entire state of a system. The variables are commonly arranged in vector form to allow the system state to be expressed as a single mathematical symbol.

state space: State space is the n-dimensional vector space defined over the field containing the state variables. Just as a position in physical space is defined by the vector of position variables, x, y, and z, state space locations are specified by the state vector. One dimension is defined for each state variable in the system. Thus, a system with four state variable is embedded in a four dimensional state space.

G.2 Control Terms

disturbance: A disturbance or external disturbance describes an external unmodelled influences on the system. Wind gusts on an airplane or rain on a chemical plant distillation column are considered disturbances. In control studies, disturbances are usually modelled by adding random noise to the system.

plant: Plant is another name for the physical system. The term originates from the wide use of automatic controls in manufacturing plants.

system dimension: The dimension of a system is the number of state variables required to define the entire system. With mechanical systems, the system dimension is often twice the number of degrees of freedom.

time domain analysis: Time domain analysis is the analysis of systems where time is one of the variables. In contrast, a system may be studied in the frequency domain by transforming the system equations with a Laplace transform. Time in the frequency domain is part of the transform rather than an independent variable. State space methods are considered time domain analysis.

BIBLIOGRAPHY

Anderson, C. W. (1987) "Strategy Learning with Multilayer Connectionist Representations.", *Proceedings of the Fourth International Workshop on Machine Learning*", Irvine, Calif, Morgan Kaufmann, pp 103-114.

Anderson, C. W., (1989a) "Learning to Control an Inverted Pendulum Using Neural Networks", *IEEE Control Systems Magazine*, 9(3):31-37.

Anderson, C. W., (1989b) "Tower of Hanoi with Connectionist Networks: Learning New Features", *Proc 1989 Machine Learning Workshop*, Cornell Univ., Ithaca, NY, June 1989.

Anderson, J. A. and E. Rosenfeld (Eds) (1988), Neurocomputing : Foundations of Research, MIT Press, Cambridge, Mass.

Andreiev, N. (1989), "A New Dimension: A Self-tuning Controller that Continually Optimizes PID constants," *IEEE Control Systems Magazine*, **9**, 100-101.

Astrom, K. J. (1983), "Theory and Applications of Adaptive Control--A Survey", *Automatica*, Vol 19, No 5, pp471-486.

Astrom, K. J. (1986), "Expert Control," *Automatica*, **13**, pp 277-286

Astrom, K. J. (1988), "Toward Intelligent Control", *IEEE Control Systems Magazine*, April 1988.

Bak, P., K. Chen (1991), "Self-Organized Criticality", *Scientific American*, January 1991 pp 46-53.

Bernard, J. A., D. D. Lanning and A. Ray, (1984) "Digital Control of Power Transients in a Nuclear Reactor", *IEEE Transactions on Nuclear Science*, Vol NS-31, No. 1, February 1984.

Bernard, J. A., A. F. Henry and D. D. Lanning, (1988a) "Application of the 'Reactivity Constraint Approach to Automatic Reactor Control", *Nuclear Science and Engineering*, Vol 98, No 2, pp 87-95.

Bernard, J. A. (1988b), "Use of a Rule-Based System for Process Control", *IEEE Control Systems Magazine*, October 1988.

Barto, A. G., C. W. Anderson, and R. S. Sutton, (1982) "Synthesis of Nonlinear Control Surfaces by a Layered Associative Search Network", *Biological Cybernetics*, **43**, pp 175-185.

Barto, A. G. R. S. Sutton, and C. W. Anderson (1983), "Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems", *IEEE Transactions on Systems, Man and Cybernetics* SMC-13: 834-846.

Barto, A. G. and P. Anandan (1985), "Pattern Recognizing Stochastic Learning Automata", *IEEE Transactions on Systems, Man and Cybernetics*, 15, pp 360-375.

Barto, A. G., and M. I. Jordan, (1987) "Gradient Following without Back-propagation in Layered Networks", *In Proceedings of the IEEE First Annual Conference on Neural Networks*, Vol II, San Diego, CA, pp 629-636.

Barto, A. G., (1989) "Connectionist Learning for Control: An Overview", *University of Massachusetts COINS Technical Report 89-89*, Amherst MA, Sept 1989.

Baum, E. B., J. Moody, and F. Wilczek (1988), "Internal Representations for Associative Memory", *Biological Cybernetics*, 59, 217-228.

Becker S. and Y. le Cun(1989), "Improving the Convergence of Back-Propagation Learning with Second Order Methods", *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, San Mateo, CA.

Bell, T. (1988), "Sequential Processing using Attractor Transitions", *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, San Mateo, CA.

Bhat, N., and T. McAvoy (1989), "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process Systems", *1989 American Automatic Control Conference*, Pittsburgh, PA, June 1989.

Brogan, W. L. (1985), Modern Control Theory, (2ed), Prentice-Hall.

Caudill, M., "Neural Networks Primer" Part I, *AI Expert*, Dec 1987.

Caudill, M., "Neural Networks Primer" Part II, *AI Expert*, Feb 1988.

Caudill, M., "Neural Networks Primer" Part III, *AI Expert*, June 1988.

Caudill, M., "Neural Networks Primer" Part VI, *AI Expert*, Feb 1989.

Caudill, M., "Neural Networks Primer" Part VII, *AI Expert*, May 1989.

Caudill, M., "Neural Networks Primer" Part VIII, *AI Expert*, Aug 1989.

Caudill, M. (1990a), "Using Neural Networks: Hybrid Expert Networks", *AI Expert*, November 1990.

Caudill, M. (1991a), "Evolutionary Neural Networks", *AI Expert*, March 1991, pp 56-61

Caudill, M. (1991b), "Neural Networks Training Tips and Techniques", *AI Expert*, January 1991, pp 56-61.

Cowan, J. D. (1990), "Neural Networks: The Early Days", Advances in Neural Information Processing Systems 2, D. S, Touretzky ed, Morgan Kaufmann Publishers, San Mateo, California. pp 828-842.

Cramer, J. E. (1990), "Adaptive Control using Neural Networks", Masters Thesis, The University of Texas at Austin, Austin, Texas.

le Cun, Y. (1989), "A Theoretical Framework for Back-Propagation", *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, San Mateo, CA.

Elman, J. L. (1988), "Finding Structure in Time", *CRL Technical Report 8801, Center for Research in Language*, University of California, San Diego, April 1988.

Erickson, J. D. (1990), "Intelligent Systems in Nasa Space Missions", *Proceedings of International Conference on Supercomputing in Nuclear*

Applications, Japanese Atomic Energy Research Institute, Mito City, Ibaraki, Japan, March 1990

Farmer, J. D., and J.J. Sodorowich (1988), "Exploiting Chaos to Predict the Future and Reduce Noise", *Technical Report LA-UR-88-901*, Los Alamos National Laboratory, March 1988.

Franklin, J. A. (1987), "Learning Control in a Robotics System", *In Proceedings of the 1987 International conference on Systems, Man and Cybernetics*, Oct 1987, pp 466-470.

Franklin, J. A. (1988), "Refinement of Robot Motor Skills using Reinforcement Learning.", *In Proceedings of the 27th IEEE Conference on Decision and Control*, Dec 1988, pp 1096-1101.

Fu, K. (1970), "Learning Control Systems-Review and Outlook", *IEEE Transactions on Automatic Control*, Vol AC-15, No. 1, pp210-221.

Gleick, J. (1987), Chaos, Viking Penguin Inc, NY, NY.

Goodwin, G. C., D. J. Hill, and M. Palaniswami (1984), "A Perspective on Convergence of Adaptive Control Algorithms", *Automatica*, Vol 20, No. 5, 1984, pp 519-531.

Grossberg, S. (1988), "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures", *Neural Networks*, Vol 1, No 1, Pergamon Press, New York, 1988.

Handelman, D. A., S. H. Lane, and J.J. Gelfand (1989), "Integration on Knowledge-Based System and Neural Network Techniques for Autonomous Learning Machines", *Proceedings of the International Joint Conference on Neural Networks*, Washington D.C.

Hansen, L. K. and P. Salamon (1990), "Neural Network Assemblies", *IEEE Transactions of Pattern Analysis and Machine Intelligence*, Vol 12, No. 10, October 1990, pp 993-1001.

Hartman, E., J. Keeler, and J. Kowalski (1989), "Layered Neural Networks with Gaussian Hidden Units as Universal Approximators", *MCC Technical Report*, ACT-ST-272-89.

Hartman, E. J. (1990), "A High Storage Capacity Neural Network Content Addressable Memory", *MCC Technical Report*, ACT-NN-173-90.

Hartman, E. and J. D. Keeler (1991a), "Predicting the Future: Advantages of Semi-Local Units", *MCC Technical Report*, ACT-NN-002-91, Jan 1991.

Hartman, E. and P. D. Heermann (1991b), "Fuzzy Control Systems", *MCC Technical Report*, ACT-NN-402-91.

Heermann P. D. and N. Khazenie (1992), "Classification of Multi-Spectral Remote Sensing Data with a Back-Propagation Neural Network", to appear in January issue of *IEEE Transactions of IGARSS*.

Higdon, D. T. (1963), "Automatic Control of Inherently Unstable Systems with Bounded Control Inputs", Thesis, Stanford University.

Hinton, G. E. (1987), "Connectionist Learning Procedures", *Tech Report CMU-CS-87-115* v II, Carnegie-Mellon University, Pittsburgh, PA, 1987.

Hirsch, M. W., and S. Smale (1974), Differential Equations, Dynamical Systems, and Linear Algebra, Academic Press, Inc.

Holland, G. H., and B. F. Skinner, The Analysis of Behavior, McGraw-Hill, New York, 1961.

Hopfield, J. J. (1982), "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of the National Academy of Sciences 79:2554-2558*. (reprinted in Neurocomputing: Foundations of Research (see above))

Hoptroff, R. G., T. J. Hall, R. E. Burge (1990), "Experiments with a Neural Controller", *Proceedings of International Joint Conference on Neural Networks-San Diego*, IEEE Publishing Services, Vol 2, pp 735-740.

Hornik, K.M., M. Stinchcombe, and H. White, "Multi-Layer Feedforward Networks are Universal Approximators," UCSD Department of Economics Discussion Paper, June 1988.

Howell, J (1990). "Inside a Neural Network", November 1990, *AI Expert*, pp 29-33.

Hudson, J. L., M. Kube, R. A. Adomaitis, I. G. Kevrekidis, A. S. Lapedes, R. M. Farber (1990), "Nonlinear Signal Processing and System Identification: Applications to Time Series from Electrochemical Reactions", *Los Alamos National Laboratory Technical Report LA-UR-90-1800*.

Ivakhnenko, A. G. (1971), "Polynomial Theory of Complex Systems",*IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-1, No 4, October 1971.

Jacobs R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton (1991), "Adaptive Mixtures of Local Experts", *Neural Computation*, **3, 79-87.**

Jameson, J. (1990), "A Neurocontroller Based on Model Feedback and the Adaptive Heuristic Critic", *Proceedings of IJCNN San Diego*, IEEE, New York.

Jameson, J. (1991), "Delayed Reinforcement Learning with Multiple Time Scale Hierarchical Backpropagated Adaptive Critics", to appear in *Handbook for Intelligent Control*.

Jordan, M. I.(1988), "Supervised Learning and Systems with Excess Degrees of Freedom", *COINS Technical Report 88-27*, University of Massachusetts at Amherst, May 1988.

Jordan, M. I. (1989), "Generic Constraints on Underspecified Target Trajectories", *IJCNN Proceedings*, IEEE, New York, June 1989.

Jordan, M. I. and R. A. Jacobs (1990a), "Learning to Control an Unstable System with Forward Modeling", Advances in Neural Information Processing Systems 2, D. S, Touretzky ed, Morgan Kaufmann Publishers, San Mateo, California.

Jordan, M. I. and D. E. Rumelhart (1990b) "Forward Models: Supervised Learning with a Distal Teacher". *MIT Center for Cognitive Science Occasional Paper #40*.

Judd, S. (1987a), "Learning in Networks is Hard", *In Proceedings of the IEEE First Annual Conference on Neural Networks*, Vol II, San Diego, CA, pp 685-692.

Judd, J. S. (1987b), "Complexity of Connectionist Learning with Various Node Functions", *COINS Technical Report 87-60*, University of Massachusetts at Amherst, July 1987.

Judd, J. S. (1988), "On the Complexity of Loading Shallow Neural Networks", *Journal of Complexity*, Sept 1988.

Kangas, J. (1990), Time-Delayed Self-Organizing Maps, *Proceedings of International Joint Conference on Neural Networks-San Diego*, IEEE Publishing Services, Vol 2, pp 331-336.

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

Kohonen, T. (1988a), "An Introduction to Neural Computing", *Neural Networks*, Vol 1, No 1, Pergamon Press, New York.

Kohonen, T. (1988b), Self-Organization and Associative Memory (2ed), Springer-Verlag, New York.

Kou, B. C. (1982), Automatic Control Systems 4ed, Prentice-Hall, Englewood Cliffs, NJ.

Kreinovich, V. Y. (1991), "Arbitrary Nonlinearity Is Sufficient to Represent All Functions by Neural Networks: A Theorem", *Neural Networks*, Vol 4, pp 381-383.

Kumar, S. S., and A. Guez (1991), "ART Based Adaptive Pole Placement for Neurocontrollers", *Neural Networks*, Vol 4, pp 319-335.

Lapedes, A. and R. Farber (1987), "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modelling", *Los Alamos National Laboratories Technical Report: LA-UR-87-2662*.

Lapedes, A. and R. Farber (1988), "How Neural Nets Works", *Los Alamos National Laboratories Technical Report: preprint LA-UR-88-418*.

Luenberger, D. G. (1984), Linear and Nonlinear Programming (2ed), Addison-Wesley, Menlo Park, Calif.

Lukes, G., B. Thompson and P. Werbos (1990), "Expectation Driven Learning with an Associative Memory", *Proceedings of the International Joint Conference on Neural Networks* (Vol 1), Lawrence Erlbaum Associates, Hillsdale, New Jersey, January 1900, pp 521-534.

Mead, C. (1990a), :Invited Talk: Time the Essential Dimension", *International Joint Conference on Neural Networks*, Wash D.C., January 1990.

Mead, C (1990b), "Neuromorphic Electronic Systems", *Proceedings of the IEEE*, Vol 78, No. 10, October 1990.

McAvoy, T., N. Wang, S. Naidu, and N, Bhat (1989), "Use of Neural Nets for Interpreting Biosensor Data", *Proceedings of Joint Conference on Neural Networks*, Wash DC. pp 1227-1233, June 1989.

Michie, D. and R. A. Chambers (1968), "BOXES: An Experiment in Adaptive Control", *Machine Intelligence 2*, Oliver and Boyd, pp 137-152.

Minsky, M. L. and S. A. Papert, Perceptrons (1ed), MIT Press, Cambridge, Mass, 1969.

Minsky, M. L. and S. A. Papert, Perceptrons (2ed), MIT Press, Cambridge, Mass, 1988.

Miyamoto, H., M. Kawato, T. Setoyama, and R. Suzuki (1988), "Feedback-Error-Learning Neural Network for Trajectory Control of a Robotics Manipulator", *Neural Networks*, Vol 1, 1988, pp 251-265.

Moed, M. C. and G. N. Saridis (1990), "A Boltzmann Machine for the Organization of Intelligent Machines", *IEEE Transactions on Systems, Man and Cybernetics*, Vol 20, No 5., September/October 1990.

Munro, P. (1987), "A Dual Back-Propagation Scheme for Scalar Reward Learning", *In Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle WA, pp 165-176.

Narendra, K. S. and M. A. L. Thathachar (1974), "Learning Automata - A Survey", *IEEE Transactions on Systems, Man and Cybernetics*, 4:323-334.

Narendra, K. S. and K. Pathasarthy (1990). "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Transactions on Neural Networks*, Vol 1, No 1, March 1990, p 4-27.

Narendra, K. S. and K. Pathasarthy (1991). "Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks", *IEEE Transactions on Neural Networks*, Vol 2, No 2, March 1991, p252-262.

Nilsson, N. L. (1965), <u>Learning Machines</u>, McGraw Hill Inc.

Nguyen, D. and B. Widrow (1989), "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks", *In Proceedings of the International Joint Conference on Neural Networks*, Vol 1, pp 357-364.

Obermayer, K. H. Ritter, and K. Schulten, "Large-scale Simulations of Self-Organizing Neural Networks on Parallel Computers: Application to Biological Modelling", *Parallel Computing* 14, North-Holland, 1990, pp 381-404.

Packard, N. M., J. P. Crutchfield, J. D. Farmer, and R. S. Shaw (1980), "Geometry from a Time-Series", *Physical Review Letters*, Vol 45,No 9, Sept 1, pp 712-716.

Pearlmutter, B. A. (1988), "Learning State Space Trajectories in Recurrent Neural Networks", *Carnegie Mellon Technical Report: CMU-CS-88-191*.

Pearlmutter, B. A. (1990), "Dynamic Recurrent Neural Networks", *Carnegie Mellon Technical Report: CMU-CS-90-196*

Peek, M. D. and P. J. Antsaklis (1990), "Parameter Learning for Performance Adaptation", *IEEE Control Systems Magazine*, Dec 1990, pp 3-11.

Pedrycz, W. (1989), Fuzzy Control and Fuzzy Systems, John Wiley & Sons Inc, pp 61-80.

Pellionisz, A. J. (1990), "About the Geometry Intrinsic to Neural Nets", *Proceedings of the International Joint Conference on Neural Networks* (Vol 1), Lawrence Erlbaum Associates, Hillsdale, New Jersey, January 1990, pp 15-18.

Percival, I. and D. Richards (1982), Introduction to Dynamics, New York, Cambridge Universal Press.

Pineda, F. J. (1987a), "Generalization of Back-Propagation to Recurrent and Higher Order Neural Networks", *Proceedings of IEEE Conference on Neural Information Processing Systems*, Denver Colorado, November 1987, pp 2229-2232.

Pineda, F. J. (1987b), "Generalization of Back-Propagation to Recurrent Neural Networks", *Physical Review Letters*, November 1987, pp 2229-2232.

Pineda, F. J. (1988), "Dynamics and Architecture in Neural Computation", *Journal of Complexity*, special issue on Neural Networks, Sept 1988.

Pineda, F. J. (1990), "Time Dependent Adaptive Neural Networks", Advances in Neural Information Processing Systems 2, D. S, Touretzky ed, Morgan Kaufmann Publishers, San Mateo, California.

Robinson, A. J., and F. Fallside (1987), "The Utility Driven Dynamic Error Propagation Network", *Technical Report CUED/F-INFENG/TR.1*, Cambridge University Department of Engineering.

Rosenberg, R. C., and D. C. Karnopp (1983), Introduction to Physical Systems Dynamics, McGraw-Hill Inc.

Rumelhart, D. E. and J. L. McCelland (Eds) (1986)., Parallel Distributed Processing (Vol 1), MIT Press, Cambridge, Mass.

Rumelhart, D. E. and J. L. McCelland (Eds) (1988)., Parallel Distributed Processing (Vol 3), MIT Press, Cambridge, Mass.

Saha, A. and J. D. Keeler (1990), "Algorithms for Better Representation and Faster Learning in Radial Basis Function Networks", *Neural Information Processing Systems*, Vol 2, Ed. David Touretzky, Morgan Kaufmann Publishing, San Mateo, CA.

Schmidhuber, J. (1990a), "Learning to Generate Focus Trajectories for Attentive Vision", *Technical Report FKI-128-90*, Institut fur Informatik, Technische Universitat Munchen, Germany, April 1990.

Schmidhuber, J. (1990b), "Towards Compositional Learning with Dynamic Neural Networks",, *Technical Report FKI-129-90*, Institut fur Informatik, Technische Universitat Munchen, Germany, April 1990.

Schmidhuber, J. (1990c), "Making the World Differentiable: On Using Self-supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments" , *Technical Report FKI-126-90(revised)*, Institut fur Informatik, Technische Universitat Munchen, Germany, Nov 1990.

Schmidhuber, J. (1990d) "Networks Adjusting Networks", *Technical Report FKI-125-90(revised)*, Institut fur Informatik, Technische Universitat Munchen, Germany, Nov 1990.

Schmidhuber, J. (1991) "Adaptive Confidence and Adaptive Curiosity", *Technical Report FKI-149-91*, Institut fur Informatik, Technische Universitat Munchen, Germany, April 1991.

Servan-Schreiber, D., A. Cleeremans and J. L. McClelland (1988), "Encoding Sequential Structure in Simple Recurrent Networks", *Technical Report CMU-CS-88-183*, Carnegie Mellon University, November 1988.

Sifer, J. F., S. J. Prouty, and P. H. Mak (1991), "Advanced Concepts for Launch Vehicle Control", *IEEE Aerospace & Electronic Systems Magazine*, Feb 1991.

Smith, S. W. (1966), "Design of Quasi-Optimal Minimum-Time Controllers", *IEEE Transactions on Automatic Control*, Vol AC-11, No. 1, Jan 1966.

Specht, D. F. (1990), "Probabilistic Neural Networks", *Neural Networks*, Vol 3, pp 109-118.

Stinchcombe M. and H. White, (1989), "Universal Approximation Using Feedforward Networks with Non-Sigmoid Hidden Layer Activation Functions", *Proceedings of the International Joint Conference On Neural Networks*, IEEE, New York.

Sutton, R. S., and B. Pinette (1985), "The Learning of World Models by Connectionist Networks.", *Proceedings of the Seventh Annual Conf. of the Cognitive Science Society*, p54-64.

Sutton, R. S. and A. G. Barto (1987), "A Temporal-Difference Model of Classical Conditioning. *In Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA.

Sutton, R. S. (1988), *"Learning to Predict by the Methods of Temporal Differences"*, Machine Learning, 3:9-44.

Takens, F. (1981), "Detecting Strange Attractors in Fluid Turbulence", In Dynamical Systems and Turbulence, D. Rand and L. S. Young, ed., Springer-Verlag, Berlin.

Tao, K. M, and M. Jurik (1988), "Improved Back-Propagation Algorithms -- A Systems Control and Identification Approach", *23 Asilomar Conference on Signals, Systems and Computers*.

Thrun, S. B., K. Moller and A. Linden (1991), "Planning with an Adaptive World Model", Advances in Neural Information Processing Systems 3, D. S. Touretzky and R. Lippmann (eds), San Mateo, CA: Morgan Kaufmann

Tveter, D.(1990), "Better Speed through Integers", *AI Expert*, November 1990.

Von Neumann, J. (1958), The Computer and the Brain, Yale University Press, New Haven.

Waltz, M. D. and K. S. Fu (1965), "A Heuristic Approach to Reinforcement Learning Control Systems", *IEEE Transactions on Automatic Control*, 10:1965, pp 390-398.

Wasserman, P. D. (1989), Neural Computing Theory and Practice, Van Nostrand Reinhold, New York.

Werbos, P. J. (1987), "Building and Understanding Adaptive Systems: A Statistical/Numerical Approach to Factory Automation and Brain Research", *IEEE Transactions on Systems, Man and Cybernetics*, 1987.

Werbos, P. J. (1989), "Maximizing Long-Term Gas Industry Profits in Two Minutes in Lotus Using Neural Network Methods", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol 19, No. 2

Werbos, P. J. (1989), "Neural Networks for Control and System Identification", *IEEE CDC Proceedings*, IEEE, New York.

Werbos, P. J. (1990a), "Consistency of HDP Applied to a Simple Reinforcement Learning Problem", *Neural Networks*, Vol 3, pp 179-189.

Werbos, P. J. (1990b), "Neurocontrol and Related Techniques", In Handbook of Neural Comp. Appl, A. Maren (ed), Academic Press.

Werbos, P. J. (1990c), "Backpropagation through Time: What it does and How to do it", *Proceedings of the IEEE*, October 1990.

Werbos, P. J. (1991), "Neurocontrol and Fuzzy Logic: Connections and Designs", Preprint.

Widrow, B., N. K. Gupta, S. Maitra (1973), "Punish/Reward: Learning with a Critic in Adaptive Threshold Systems", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-3, No 5, Sept 1973.

Widrow, B. J., J. McCool, and B. Medoff (1978), "Adaptive control by Inverse Modeling", *Twelfth Asilomar Conference on Circuits, Systems, and Computers*.

Widrow, B. (1987), "The Original Adaptive Neural Net Broom-Balancer", *Proceedings of IEEE International Symposium on Circuits and Systems*.

Wieland, A. P. (1990), "Evolving Controls for Unstable Systems", Connectionist Models: *Proceedings of the 1990 Summer School*, Morgan Kaufmann Publishers.

Williams, R. J., (1988) "On the use of Backpropagation in Associative Reinforcement Learning", *In Proceedings of the IEEE International Conference on Neural Networks*, San Diego.

Williams, R. J. (1988), "A Learning Algorithm for Continually Running Fully Recurrent Neural Network", *ICS Report 8805*, Oct 1988.

Williams, R. J., and D. Zipser (1989), "Experimental Analysis of the Real-time Recurrent Learning Algorithm", *Connection Science*, Vol 1, No 1.

Witten, I. H. (1977), "An Adaptive Optimal Controller for Discrete-Time Markov Environments", *Information and Control*, Academic Press, pp 286-295.

Wittenmark, B. and K. J. Astrom, (1984) "Practical Issues in the Implementation of Self-tuning Control", *Automatica*, Vol 20, pp595-605.

Wylie, C. R., and L. C. Barrett (1982), <u>Advanced Engineering Mathematics</u>, McGraw-Hill Book Company.

Yager, R. R., (1990), "A Model of Participatory Learning", *IEEE Transactions of Systems, Man and Cybernetics*, Vol 20, No 5 Sept/Oct , pp1229-1234.

Yuh, J. (1990), "Modelling and Control of Underwater Robotics Vehicles", *IEEE Transactions on Systems, Man and Cybernetics*, Vol 20, No 6, Nov/Dec 1990.

Zak, M. (1989), "Terminal Attractors for Addressable Memory in Neural Networks", *Physics Letters*, A, 133, p18-22.

Zak, M. (1989), "Terminal Attractors in Neural Networks", *Neural Networks* (Vol 2, No 4), Pergamon Press, New York.

VITA

Philip Dale Heermann was born in Fort Collins, Colorado, on December 18, 1962, the son of Betty Marie Heermann and Dale Frank Heermann. After completing his work at Rocky Mountain High School, in 1981, he entered Colorado State University. While attending Colorado State University he was employed as a Research Assistant at the REPEAT Passive Solar Test Facility. He received a Bachelor of Science in Mechanical Engineering from Colorado State University in May 1985. In September, 1985, he entered the Graduate School of the University of Texas at Austin, where he completed a Master of Science in Mechanical Engineering in December of 1987. From December 1985 to the present he has been employed by ALFA Engineering Inc. He was a member of the Research Staff at Micro-electronics and Computer Technology Corporation from August 1990 through May 1991. He has also provided consulting services to USDA-ARS, Colorado Energy Associates, Fort Collins Sports Medicine Clinic, and The Analytical Science Corporation (TASC).

Permanent address: 4780 Hogan Drive
Fort Collins, Colorado 80525

This dissertation was typed by the author.