Adaptive Element Controller for an Inverted Pendulum

M. J. Colley

2 November 1990

Contents

List of Figures

List of Tables

1 Introduction

The project illustrates the ability of a neuronlike adaptive element to solve a control problem, the dynamics of which it has no prior knowledge. Using state normalisation, the element can be designed without being system specific ie. it can be used to control any system using a four dimensional state vector.

The task used as an example is to balance a pole that is attached to a movable cart by a hinge, by applying a force to the cart. The cart moves along a track of finite length.

The adaptive element (AE) uses reinforcement, or evaluation signals, to update an internal information base, which it uses to determine future control actions.

This report discusses the structure of the AE as simulated in the project and investigates the effect of the parameters of the AE on its ability to learn, and the speed at which it learns. This is done by outlining the implimentation of the AE, defining the parameters, and tabulating the results of some tests that were done to investigate parameter sensitivity, using the pole balancing task as an example of a controlled system. A discussion of these results concludes the report.

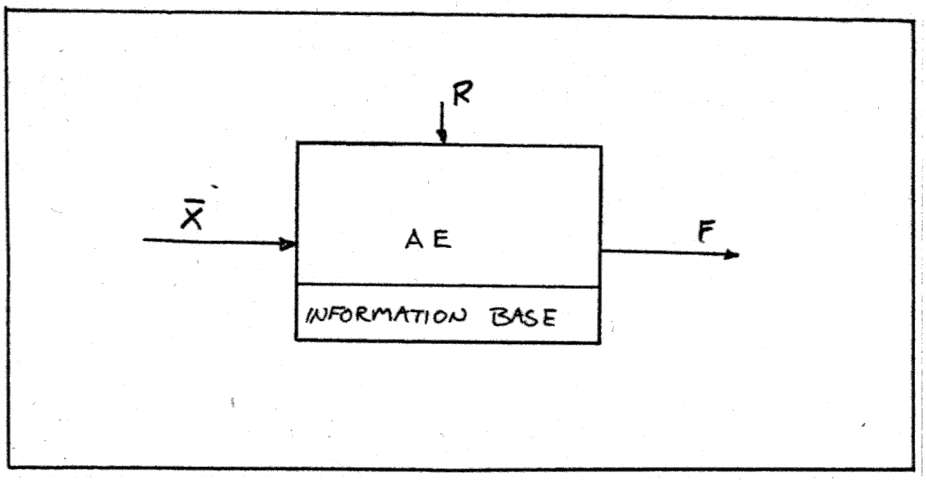

Figure 1: Structure of an Adaptive Element

2 The Adaptive Element

The AE (see fig. 1) has a multidimensional input X describing the state of the controlled system. It has an output F, which is the control variable, and it has an input R for reinforcement. It also has an internal database $W[j]$ which it uses to derive the control output. The reinforcement signal is used to update the database.

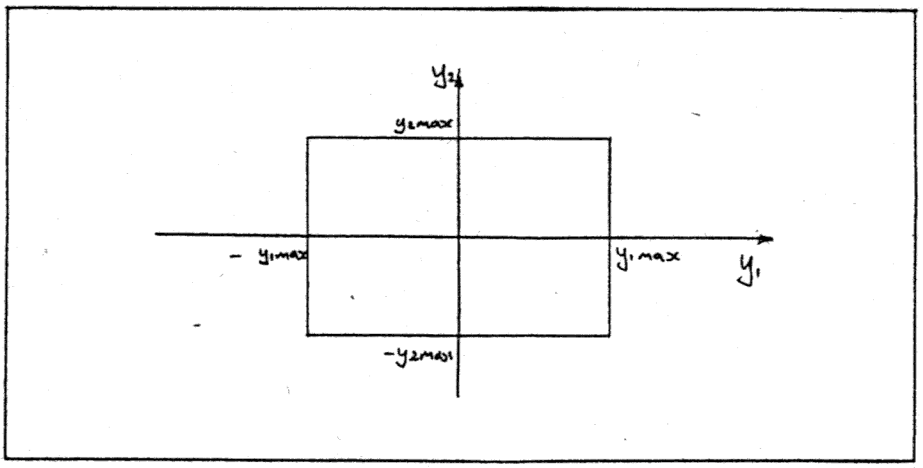

Figure 2: The two regions of the State Space

2.1 The Input Vector X, And Normalisation of the State Space

In the simulation, a four dimensional vector $X = [x_1, x_2, x_3, x_4]$ is used to describe the state of the controlled system ie. its position in the state space.

To ease illustration, the explanations are in terms of a two dimensional state space $Y = [y_1, y_2]$.

The state space of the system can be divided into two regions: the region of interest, and the sink cell (see fig. 2).

The system is initialised with $Y(0)$ inside the region of interest. This forms the input to the AE, and a control action is calculated, $F(0)$, which together with $Y(0)$ determines the next position in the state space, $Y(1)$, which becomes the AE's next input.

If, after a series of a control actions, the system enters the sink cell, then the system is reinitialised to a new $Y(0)$ inside the region of interest. This would correspond, in the cart-pole example, to the pole falling flat, or the cart reaching the end of the track, so that the region of interest is bounded in the one dimension by the length of the track, and in the another dimesion by some maximum angle of the pole.

In this project, the state space is normalised to the outer bounds of the region of interest (see fig. 3) ie. the state y_i is normalised so that $y_{imax} = 1$, and the sink cell is characterised by a state vector with one or more of its components having an absolute value greater than one.

The AE gets the normalised state vector as its input, ie any four dimensional system can be experimented with without changing the structure of the AE.

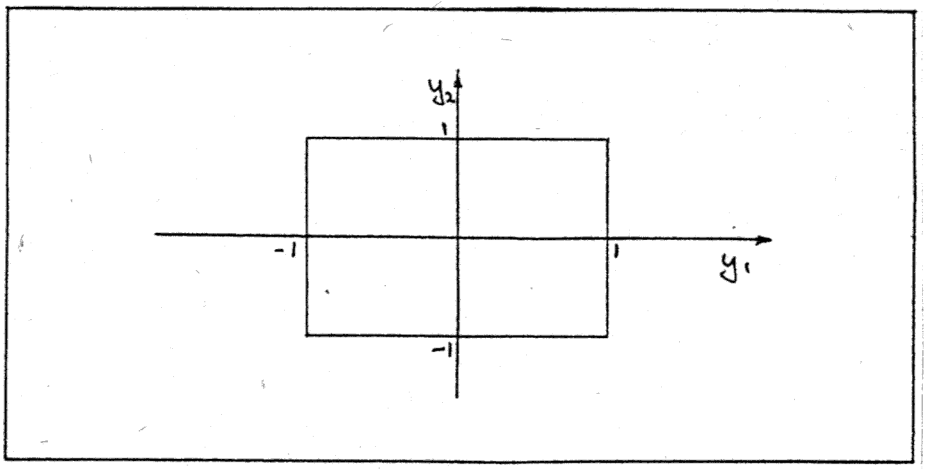

Figure 3: The Normalised State Space

2.1.1 The Cart Pole State Vector

In the cart pole example, the state vector comprises the position of the cart, z, the angle between the pole and vertical, θ, and the derivatives of these states, $\frac{dz}{dt}$ and $\frac{d\theta}{dt}$.

A scaling vector $S = [|z_{max}|, |dz_{max}|, |\theta_{max}|, |d\theta_{max}|]$ forms the boundry of the region of interest. The normalised state vector X is calculated by dividing each component of the state vector by its corresponding element in the scaling vector.

$$x_1 = \frac{z}{z_{max}} \quad (1)$$

$$x_2 = \frac{dz}{dz_{max}} \quad (2)$$

$$x_3 = \frac{\theta}{\theta_{max}} \quad (3)$$

$$x_4 = \frac{d\theta}{d\theta_{max}} \quad (4)$$

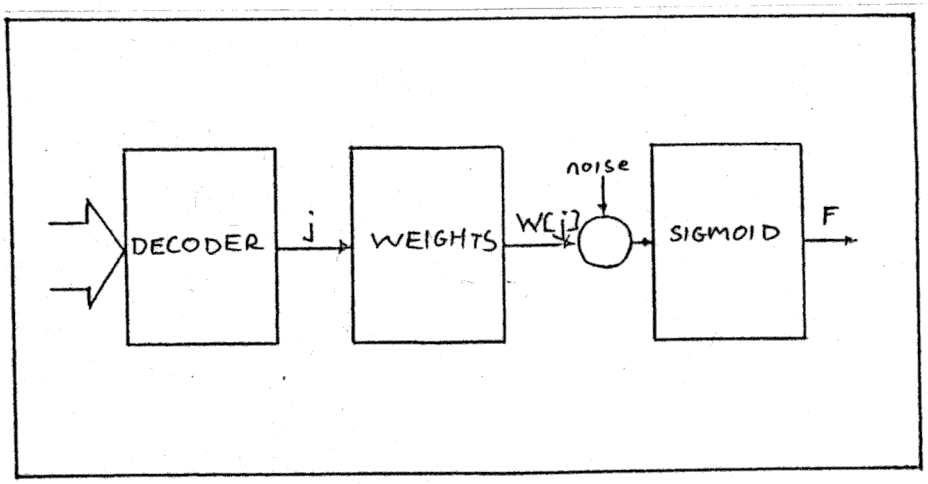

Figure 4: The Internal Structure of the AE

3 The Internal Structure of the AE

The structure of the AE is shown in diagramatic form in fig. 4. The normalised state vector is discretised by the state decoder into 1 of 1296 discrete regions. For each of the discrete regions there is a corresponding weight.

The control output is a sigmoidal function of the weight corresponding to the discrete state plus noise. This makes the control output a random function biased by the weight corresponding to the state the system is in at the time of discretisation.

3.1 State Decoder

Michie and Chambers [2,3] developed a system they called the BOXES system to interpret the position of the controlled system in the state space for the AE. Although this function could be carried out by a set of AEs on a patern recognition basis, most of the research since then into AE controllers uses some form of the BOXES system. The same has been done here, although it is developed as a four dimensional disctetisation process.

Discretisation of the controller state vector is achieved by dividing the normalised region of interest for each state into six intervals. For each state x_i, two thresholds are defined $t_1[i]$ and $t_2[i]$, as illustrated in fig. 5.

This results in dividing the four dimensional region of interest into $6*6*6*6 = 1296$ discrete regions.

The choice of method and numbering system for the different regions of each state was made with decoder speed in mind. For each state the decoder needs a maximum of three tests, and a minimum of two tests, giving a total maximum of twelve tests if the position in the state space is close to the boundry of the region of interest.

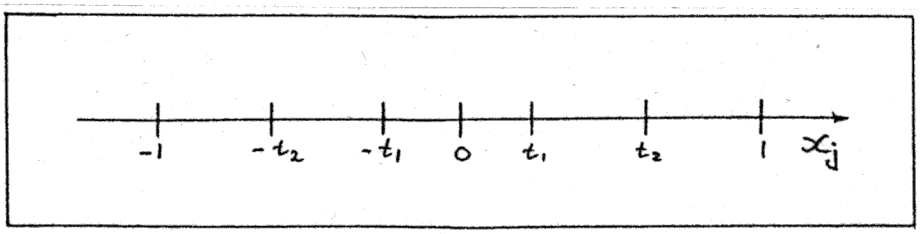

Figure 5: State x_i showing Thresholds t_1, and t_2.

If the position in the state space is close to the position $X = [0, 0, 0, 0]$ then the decoder will need only eight tests for the four dimensional disctetisation process.

The first test in each state is a sign test, the second test compares the state's magnitude to the first threshold, and if it is greater, then the third test is done comparing magnitude to the second threshold.

From the results of these tests, a number from 0 to 5 is found for each state, resulting in a hex number four digits long. This hex number is then converted to decimal to find the discrete state, which is the decoder output, j.

3.2 Weight Table, Noise, and the Output F

The data base is an array $W[0..1296]$. Corresponding to disctete state j, is a floating point number $W[j]$. The output F is calculated using the following sigmoidal function of the weight $W[j]$ plus $noise()$,

$$F = \frac{2}{1 + exp(-Alpha * W[j] + noise())} - 1 \tag{5}$$

giving an output $-1 < F < 1$.

Alpha is a parameter of the AE, and the function $noise()$ makes use of the random function to give a random number between $Noisemax$ and $-Noisemax$, where $Noisemax$ is a parameter of the AE.

4 The Learning Process

A novice AE is initialised with all the weights $W[j] = 0$. The Weight table is updated during the learning process using the reinforcement input R, which can have the values 0, -1, and 1, corresponding to no reinforcement, negative reinforcement (-Ve_R), and positive reinforcement (+Ve_R).

Michie and Chambers as mentioned in [1] used only negative reinforcement in their solution of the cart pole controller. Barto et al.[1] used a seperate AE, called the "Adaptive Critic Element (ACE)", whose output (positive or negative) formed the R input of the controller AE, which reduces the uncertainty under which the controller AE must learn.

The reinforcement methods used in this solution is discussed in the following sections.

4.1 Negative Reinforcement (-Ve_R)

A -Ve_R vector is defined as a parameter of the AE. This vector describes the maximum amplitude that each state in the normalised vector can take on before -Ve_R takes place. The vector $-Ve\_Rvec$ effectively describes a region within the normalised region of interest. If $-Ve\_Rvec = [1, 1, 1, 1]$, then -Ve_R takes place when the system leaves the region of interest. -Ve_R has the effect on the weights that results in the AE being less likely to make the same control decision that caused the system to leave the -Ve_R region in future.

4.2 Positive Reinforcement (+Ve_R)

Similarly, a +Ve_R vector, describing a region in the normalised state space is defined as an AE parameter. If the state vector is outside this region, and enters the +Ve_R region, then the weights are adapted so as to strengthen the control decisions made to cause this to happen, making it more likely to happen again.

4.3 Eligibility

A trace of the dicrete states, together with the control actions at each iteration is stored during each trial, for reinforcement purposes. When reinforcement takes place ie the system leaves the area described by $-Ve\_Rvec$, or enters the area described by $+Ve\_Rvec$, the most recently used weights are more responsible for the result than those corresponding to discrete states entered earlier. For this reason, these

discrete states are more eligible to have their weights reinforced, and an eligibility trace is calculated to correspond to the trace of discrete states entered.

When a new trial starts, an iteration counter "trial_loop", is set to zero, and incremented each time a control decision is made. This counter forms the index for the state trace, and the eligibility trace. At the time of reinforcement a counter i is decremented from the value of trial_loop down to zero.

The eligibility trace is calculated using the recursion formula

$$e(i) = e(i+1) - de * e(i+1) \tag{6}$$

where $e(i)$ is the eligibility of the discrete state entered at iteration i to have its weights changed, and de is an AE parameter, "eligibility decay rate".

Reinforcement is implimented at each value of i using the formula

$$W[j] = W[j] + R * dw * e(i) * F(i) \tag{7}$$

where $W[j]$ is the weight corresponding to discrete state j, entered at iteration i; R takes on the value +1 for +Ve_R, and -1 for -Ve_R; dw is an AE parameter "rate of change of weights"; and $F(i)$ is the control decision made at iteration i.

5 System Parameters

The AE parameters used during the simulation have an effect on the learning curve of the AE, and can be changed by the user to investigate their effects.

5.1 *dw* : Rate of Change of Weights

This parameter effects how much a weight is altered during reinforcement process. If *dw* is set to zero, reinforcement has no effect.

5.2 *de* : Eligibility Decay Rate

During a reinforcement process, the states (discrete) entered that led to the reinforcement action have their corresponding weights altered by an amount proportional to their responsibility for the result. A weight used at iteration i has its weights altered by a factor *de* more than a weight used at iteration $i-1$, as mentioned in the section titled "Learning Process".

5.3 *Alpha* : The Sigmoid Parameter

Alpha effects the "flatness" of the sigmoidal function used to calculate F, the AE output. As *Alpha* is increased, the sigmoid becomes steeper - nearing a relay function, and a smaller value gives a gradual increase - aproaching a linear function.

5.4 T : The Cycle Period

This parameter is used to calculate the state resulting from a control action as described in the section titled "Cart Pole System".

5.5 Reinforcement

Two parameters *r_positive*, and *r_negative* are boolean variables indicating the use of +Ve_R and -Ve_R respectively.

5.6 *Noisemax*

This parameter determines the upper and lower bounds of the *noise*() function used in calculating the AE output.

5.7 State Normalisation Vector

This vector defines the outer bounds of the region of interest, and is used to scale the state vector to a maximum value of $X = [1, 1, 1, 1]$, so that the dynamics of the AE are not system specific ie any four dimensional system can be experimented on without changing the structure of the program.

5.8 $t_1[]$ and $t_2[]$ The Threshold Vectors

These two vectors define the thresholds used in the state decoder,subdividing the region of interest for each state. By setting $t2[j] = 1$, the number of subdivisions of the state j will be reduced to 4, and similarly if $t1[j] = 1$, there will only be two discrete regions of the state j.

5.9 $+Ve\_Rvec$: +Ve Reinforcement test vector

This vector describes the region of the normalised state space that the system must enter for +Ve_R to take place. To illustrate this, consider the vector $+Ve\_Rvec = [1, 1, 0.2, 1]$ in the example of the cart pole system. The third element, corresponding to normalised θ indicates that if $|\theta|$ returns to less than $(0.2 * \theta_{max})$ then +Ve_R takes place, θ_{max} being the boundry of the region of interest of the state θ. The values of one for the remaining states in the vector indicate that these states do not effect the test for +Ve_R.

5.10 $-Ve\_Rvec$: -Ve Reinforcement

This vector describes the region in the normalised state space that the system must leave for -Ve_R to take place. If the magnitude of one of the states in the normalised state vector exeeds the corresponding element in the $-Ve\_Rvec$, -Ve_R takes place. If the vector has the value $-Ve\_Rvec = [1, 1, 1, 1]$, then this only happens if the system leaves the region of interest.

6 Choice of parameters

The choice of parameters effects the learning speed of the AE - as the environment might effect the learning curve of a system of real neurons. In fact as mentioned by Barto et al.[1], when Mitchie and Chambers first developed the BOXES system, they pointed out that it is easy to choose partitions, (corresponding to the threshold vectors), that would make the learning task impossible.

The ideal situation would be to update the parameters dynamicaly, as the learning process is taking place; for example, the rate of change of weights could be decreased as the AE develops. The use of extra AEs as Barto et al. [1] used for the reinforcement, could be extended to control all the parameters involved in the AE's learning, as well as for the decoder.

In this project, as the program was developed, a set of default parameters was developed by trial and error - based on the mathematics of the system.

For example, the threshold vectors were tailored so as to give the derivative states in the cart-pole system less divisions in the decoder, which results in a faster conversion in those weights which are used - even if some of the weights remain unused. The AEs developed by Barto et al. [1] had three divisions of each state except θ, which had six divisions - corresponding to $3 * 3 * 3 * 6 = 162$ discrete states.

With the default thresholds $t_1 = [0.2, 0.001, 0.2, 0.001]$ and $t_2 = [0.6, 1, 0.6, 1]$, there are 6 divisions in the angle and displacement states, and effectively 4 states in each of their corresponding derivative states, using $4 * 4 * 6 * 6 = 576$ of the maximum 1296 weights being made use of.

7 The Programs *NNC.EXE* and *WANAL.EXE*

The simulation program *NNC.EXE* was written in *C v*2.0.

The structure is such that any four dimensional system can be implimented simply by changing one procedure, $Update(*state, reaction)$, which calculates the new normalised state vector, taking as parameters the old state vector and the actuation output of the AE - also normalised.

The graphics functions - which shows the cart and pole - is also isolated from the workings of the AE, so that this can also be adapted easily to illustrate a new four dimensional system.

The AE's detailes - the weights and a summary of their history - are stored in one file, and the parameters chosen for the neuron are stored in another. In this way a number of program runs can be done with the same set of weights - with the user being able to alter parameters between runs.

The program *WANAL.EXE* can be used to compare results of program runs. It shows statistics and parameter values for specific tests.

8 The Cart Pole System

The system dynamics of the cart-pole control problem (see fig. 6) can be modelled as follows [4] :

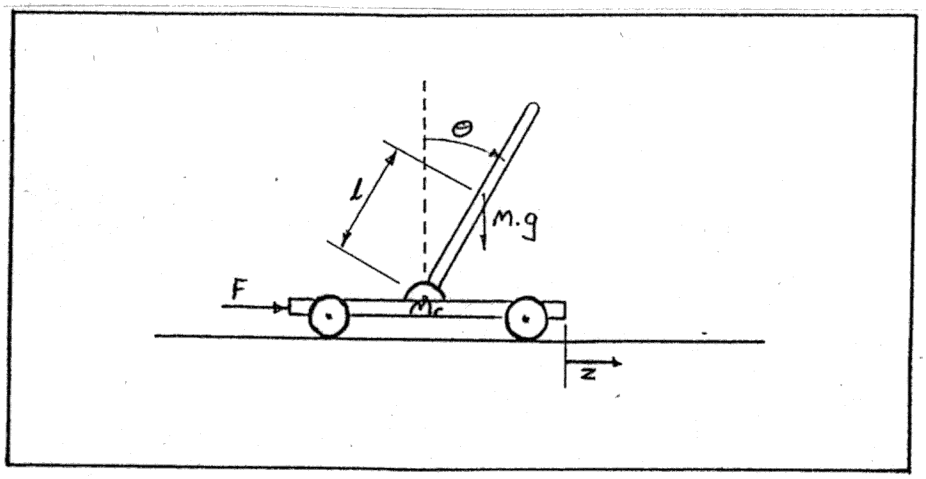

Figure 6: The Cart Pole System

Where $F(t)$ is the actuation force; g the force of gravity; m the mass of pole; z is displacement along the track; l the 1/2 length of the pole; and mc is the mass of the cart.

Assumptions made : $Sin\theta = \theta$; $Cos\theta = 1$; (ie θ is small); No friction.

Horizontal forces :

$$F(t) = (m + mc) * \frac{d^2z}{dt^2} + m * l * \frac{d^2\theta}{dt^2} * Cos\theta \tag{8}$$

Rotational forces :

$$m * g * Sin\theta = m * l * \frac{d^2z}{dt^2} * Cos\theta + m * l^2 * \frac{d^2\theta}{dt^2} \tag{9}$$

With the assumptions made, this gives the state equations :

$$\frac{d^2z}{dt^2} = \frac{F(t) - m * g * \theta}{mc} \tag{10}$$

$$\frac{d^2\theta}{dt^2} = \frac{g * \theta - \frac{d^2z}{dt^2}}{l} \tag{11}$$

$$\frac{dz}{dt} = \frac{dz}{dt} + T * \frac{d^2z}{dt^2} \tag{12}$$

$$z = z + T * \frac{dz}{dt} \tag{13}$$

$$\frac{d\theta}{dt} = \frac{d\theta}{dt} + T * \frac{d^2\theta}{dt^2} \tag{14}$$

$$\theta = \theta + T * \frac{d\theta}{dt} \tag{15}$$

The magnitude of the controlling output from the AE has a limit of 1, which is multiplied by $10N$ to get a maximum actuation force of $10N$.

System constants used in the simulation are as follows :

$$g = 9.8m/s^2 \tag{16}$$

$$mc = 1kg \tag{17}$$

$$m = 0.1kg \tag{18}$$

$$l = 0.5m \tag{19}$$

+Ve_R?	1	Positive Reinforcement
-Ve_R?	1	Negative Reinforcement
dw	1	Rate of Change of Weights
de	0.1	Eligibility Decay Rate
Noisemax	1.0	Maximum Noise Amplitude
Alpha	2.2	Sigmoid parameter
T	0.02	Cycle period

Table 1: Table of Default Parameters

Scaling vec	$sc =$	[1,	500,	0.2094	500]
Threshold_1	$t_1 =$	[0.2,	0.001,	0.2,	0.001]
Threshold_2	$t_2 =$	[0.6,	1,	0.6,	1]
+Ve_R vec	$t_3 =$	[1,	1,	0.2,	1]
-Ve_R vec	$t_4 =$	[1,	1,	1,	1]

Table 2: Table of Default Vectors

9 Readings : Choice of Parameter Values

A number of tests were done to try and illuminate the effects of the different parameters on the learning curve of the neuron, and the capabilities of the finished product.

These readings were done by comparing the test trials (1000 iterations) of AE's with altered parameters to those of an AE's with default parameters, after a learning run (10000 iterations) had been completed.

The default parameters chosen are as follows are shown in table 1 and 2.

The two variables compared after the learning and test runs are the average trial length $\overline{TL}$ ie the average number of iterations before state reset, and $NonZ$, the number of nonzero weights in the array $W[j]$.

The vectors changed in the tests are:

$t_{1A} = [0.1, 0.0005, 0.1, 0.0005]$
$t_{1B} = [0.4, 0.001, 0.4, 0.001]$
$t_{2A} = [0.8, 1, 0.8, 1]$
$t_{3A} = [0.2, 1, 0.2, 1]$
$t_{3C} = [0.6, 1, 1, 1]$

Simulated time		3.33 mins		20 sec	
AE Name	Parameter Chaged	$\overline{TL}$	$NonZ$	$\overline{TL}$	$NonZ$
TA1	Default Parameters	79.36	254	200	0
TA2	Default Parameters	79.36	254	125	1
TA3	Default Parameters	79.36	254	100	2
TA4	Default Parameters	86.96	258	100	0
TA	Default Average	81.26	255	131.25	0.75
TB	No +Ve_R	36.76	295	28.57	5
TC	No -Ve_R	43.10	150	52.63	2
TD	$dw = 0.7$	62.5	245	76.9	1
TE	$de = 0.5$	46.73	275	62.5	8
TF	$Alpha = 1$	42.19	204	52.6	6
TG	$Alpha = 4$	68.49	261	83.33	0
TH	$t_1 = t_{1A}$	113.64	281	200	0
TI	$t_1 = t_{1B}$	120.48	189	250	0
TJ	$t_2 = t_{2A}$	125.0	188	250	0
TK	$t_3 = t_{3A}$	89.29	206	111.11	2
TL	$t_3 = t_{3C}$	27	299	30.3	7
TM	$sc = sc_A$	151.5	186	200	7

Table 3: Table of Results

10 Discussion on Results

The results of the test trials are shown in table 3.

The two factors $\overline{TL}$ and $NonZ$ play off against each other in an evaluation of the parameter choise made, ie with a low number of nonzero weights goes the posibility that in future trials the unaltered weights will cause quick failure. A high average trial length, together with a low number of nonzero weights is difficult to compare to a low average trial length together with good average convergence of most of the weights.

In some cases a conclusion cannot be reached in a comparison, for instance, in test TM, (see table) an $\overline{TL} = 200$ compared to th average of $131,75$ for the default case is due to the fact that the cart has a longer track to move on, and therefore takes more iterations to reach the barrier from $z = 0$.

The Main conclusion that can be reached is that positive and negative reinforcement applied during training is more efficient than only one or the other. It is also obvious from tests TH through TL that the choice of threshold is very important.

Specificaly, smaller discrete regions around the zero points of the states results in a better ability to keep the pole stable, but less likely to get it into that state in the first place. There are also a larger number of states altered due to the fact that the

extreme regions of the state space begin closer to the origin than in the default case, making it more likely that the system will enter these regions.

Aplying +Ve_R on displacement along the track as well as on angle of the pole also resulted in a longer $\overline{TL}$, but at the same time, learning speed was decreased - as indicated by the fact that two new states (discrete) were entered during the test period; wheras in the default case, where +Ve_R was implimented with the aim of encouraging the AE to keep the pole upright, fewer new states were entered during the test period.

The default value for the parameter *Alpha* - the sigmoid parameter - was chosen by comparing a series of values. Tests TF and TG show the effects of larger and smaller choices for this parameter respectively. A lower value decreases learning speed because more learning steps (reinforcement) are necessary in order to get the weight values up to the magnitude necessary for a control action close to the maximum magnitude of 1.

Test TD shows that a smaller value of the parameter *dw*, "rate of change of weights", decreases learning speed - as the name of the parameter might indicate. The nearer the weight values come to their convergence values, the lower this parameter would need to be, however, otherwise convergence would be slowed down due to oscilations about the convergence value.

11 Conclusion

This type of adaptive element shows promise in the area of system controllers although a theory relating the parameter values and training methods is lacking.

While optimum values for the parameters during early learning might be found, these values would need to be altered throughout the learning process so as to give a stable value for the all the weights that can be used. This function might be best given to another AE, or net of AE's, but there will always be the problem of choosing parameters and training methods for the overall system.

12 References

1. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems." *IEEE Trans. Syst., Man., Cybern.,* vol. SMC-13, pp834-846, 1983.

2. D. Michie and R. A. Chambers, "BOXES: An Experiment in Adaptive Control," in *Machine Intelegence 2,* E. Dale and D. Michie, Eds. Edinburgh: Oliver and Boyde, 1968.

3. D. Michie and R. A. Chambers, "'Boxes' as a Model for Pattern-Formation," in *Towards a Theoretical Biology,* vol. 1, *Prolegomena,* C. H. Waddington, Ed. Edinburgh: Edinburgh Univ. Press, 1968.

4. Classnotes from *Beheerstelsels 444,* 1989.