Pole Balancing on a Real Rig using a Reinforcement Learning Controller

T.T.Jervis and F.Fallside

CUED/F-INFENG/TR 115

December 16, 1992

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

Email: ttj10@eng.cam.ac.uk

Pole Balancing on a Real Rig using a Reinforcement Learning Controller

Technical Report CUED/F-INFENG/TR 115

T.T.Jervis
ttj10@eng.cam.ac.uk

F.Fallside

December 16, 1992

Abstract

In 1983, Barto, Sutton and Anderson [3] published details of an adaptive controller which learnt to balance a simulated inverted pendulum. This *reinforcement learning* controller balanced the pendulum as a by-product of avoiding a cost signal delivered to the controller when the pendulum fell over. This paper describes their controller learning to balance a real inverted pendulum. As far as the authors are aware, this is the first example of a reinforcement learning controller being applied to a real inverted pendulum learning in real time.

The results show that the controller was able to improve its performance as it learnt, and that the task is computationally tractable. However, the implementation was not straightforward. Although some of the controller's parameters were tuned automatically by learning, some were not and had to be carefully set for successful control. This limits the usefulness of this kind of learning controller to small problems which are likely to be better controlled by other means. Before a learning controller can tackle more difficult problems, a more powerful learning scheme has to be found.

1 Introduction

Machines that perform difficult, mundane or dangerous tasks for us add to our quality of life. One way to improve our lives further is to make machines more capable. A machine that learns is potentially more capable. It might be possible to make a learning machine that solves problems that are too difficult for us to solve explicitly. It might also be possible for a learning machine to improve on solutions we already have, either by performing tasks in less time, or by using fewer resources, or both.

Generally a machine comprises a controller and some plant, which exist together in an environment. Making a learning machine is about making a learning controller. The controller should learn to improve the commands it issues to the rest of the machine to produce more satisfactory results in the environment.

Some learning controllers are already in use commercially; Åström [2] gives a number of examples. Self-tuning controllers adjust to fixed plant parameters that are not completely known at design time. Adaptive controllers control plants whose parameter values vary over time. When uncertainty in the controlled plant is not severe, self-tuning and adaptive controllers can even be proved stable. For example, Model Reference Adaptive Systems (MRAS) [2] use a kind of adaptive control that can sometimes be proved stable knowing only the sign, and not the value, of some plant parameters. Stability is an important aspect of a controller. It may be unsafe or financially risky to use a controller without a proof of its stability.

If less is known about the plant (in the example above, if not even the sign of the critical plant parameters is known) or if the plant is difficult to analyse (for example, a walking robot navigating through uneven terrain), a more powerful learning controller might be useful. Such a learning controller may not have the benefit of the rigour of conventional control theory, and might not be provably stable, but there may be little alternative.

A variety of test problems have been tackled with learning controllers that are more flexible than the standard self-tuning and adaptive kinds. Nguyen and Widrow [14] used two neural networks in a controller that learnt to reverse an articulated lorry into a loading bay (in simulation). Unfortunately, their solution requires a careful training schedule, and many training examples. Moore has approached a number of different control tasks, including playing billiards [13] and manipulating a real robot using two video cameras for sensory input [6]. His techniques make use of memory-based methods that have the advantage of one-shot learning. This approach can lead to reasonable performance with a minimum of experience. Miller has used a Cerebellar Model Arithmetic Computer (CMAC) structure in a real robot task using video input [9]. Tham [16] has used CMACs to learn to position a two-joint robot arm in an environment with obstacles (in simulation).

There are other examples of learning controllers being applied to problems of varying complexity, but there are problems with the controllers. Some require careful design for the task at hand, defeating the purpose of learning control. Some require many training examples that would wear out any physical apparatus before convergence, and although this problem may be alleviated by training using an off-line simulation, this also requires careful design. Thus there is still progress to be made, and applying learning controllers to problems which can already be solved by other techniques is still instructive. Learning controllers might later be applied to more complex tasks.

A standard problem for learning controllers is that of balancing an inverted pendulum pivoted on a trolley, a problem similar to that of balancing an upside-down broom on one's hand [8]. It is attractive because it is simple to simulate, requiring small amounts of memory and little computational effort, and yet its solution is not trivial. Michie and Chambers [12] presented a solution using their "boxes" paradigm, dividing up the problem into a number of simple problems to be solved locally. Jordan and Jacobs [11] used two neural networks in their forward-modelling solution, one to control the pendulum, another to judge the expected performance of the control policy. This solution is more adaptive, but takes a large amount of experience and time to train. Anderson [1] has also taken a neural network approach to the inverted pendulum task, but again the neural network, trained by gradient descent, requires many training examples.

The inverted pendulum problem is tackled in other fields too. Two recent examples of fuzzy-logic control include Berenji and Khedkar's [4] use of a neural network in combination with a fuzzy-logic controller to balance a pendulum. They do not indicate the amount of training required for convergence. Jang [10] also uses an adaptive fuzzy-logic controller on a restricted form of the problem (he ignores any requirements on the position and velocity of the trolley).

This work follows up a solution offered by Sutton, Barto and Anderson [3] to the problem of learning to balance a simulated inverted pendulum. By penalising the controller when the pendulum collapsed, their controller learnt a control policy that ensured the pendulum stayed upright indefinitely. This work is about applying their controller to a real rig, very similar to the simulated rig of the earlier work.

Sutton *et al*'s controller, the ACE/ASE controller, is an example of a *reinforcement learning* controller. Reinforcement learning is one way of realising control by only specifying the control goals. The idea is that desirable outcomes are reinforced positively and undesirable ones negatively. The problem of how to achieve positive reinforcement is left to the controller. Training through reinforcement learning requires a less knowledgeable teacher than supervised learning requires, since the teacher does not need to give the correct control actions as examples for the controller to learn.

The ACE/ASE algorithm learns a control policy to balance the pendulum. This work was not particularly concerned with the policy-learning aspect of the work, however. Instead it emphasises the problems of applying any policy learning scheme to control problems, and shows that there is more that a learning controller needs to learn before it becomes useful. Thus, the problems with this implementation are expected to be repeated with other policy-learning algorithms such as Watkins' Q-Learning [17], for example.

The structure of this work is as follows. The ACE/ASE controller of Barto *et al.* [3], successful at balancing the inverted pendulum in simulation, is outlined in section 2. Section 3 gives the changes made to the ACE/ASE controller to make it work on the real apparatus available for this experiment. Section 4 details how the modified ACE/ASE controller was then tested on the apparatus, the results being given in section 5. Section 6 draws some conclusions. Work for the future is described in section 7. More information about the particulars of the apparatus used, details of the controller's parameters, and an outline of the relevant control theory are given as appendices.

2 The ACE/ASE pendulum balancer

Figure 1 shows the general structure of the ACE/ASE controller and its relationship with its plant, the inverted pendulum. The main difference between this work and that described in the original paper [3] is that the pendulum/carriage system in this work is real, not a simulation.

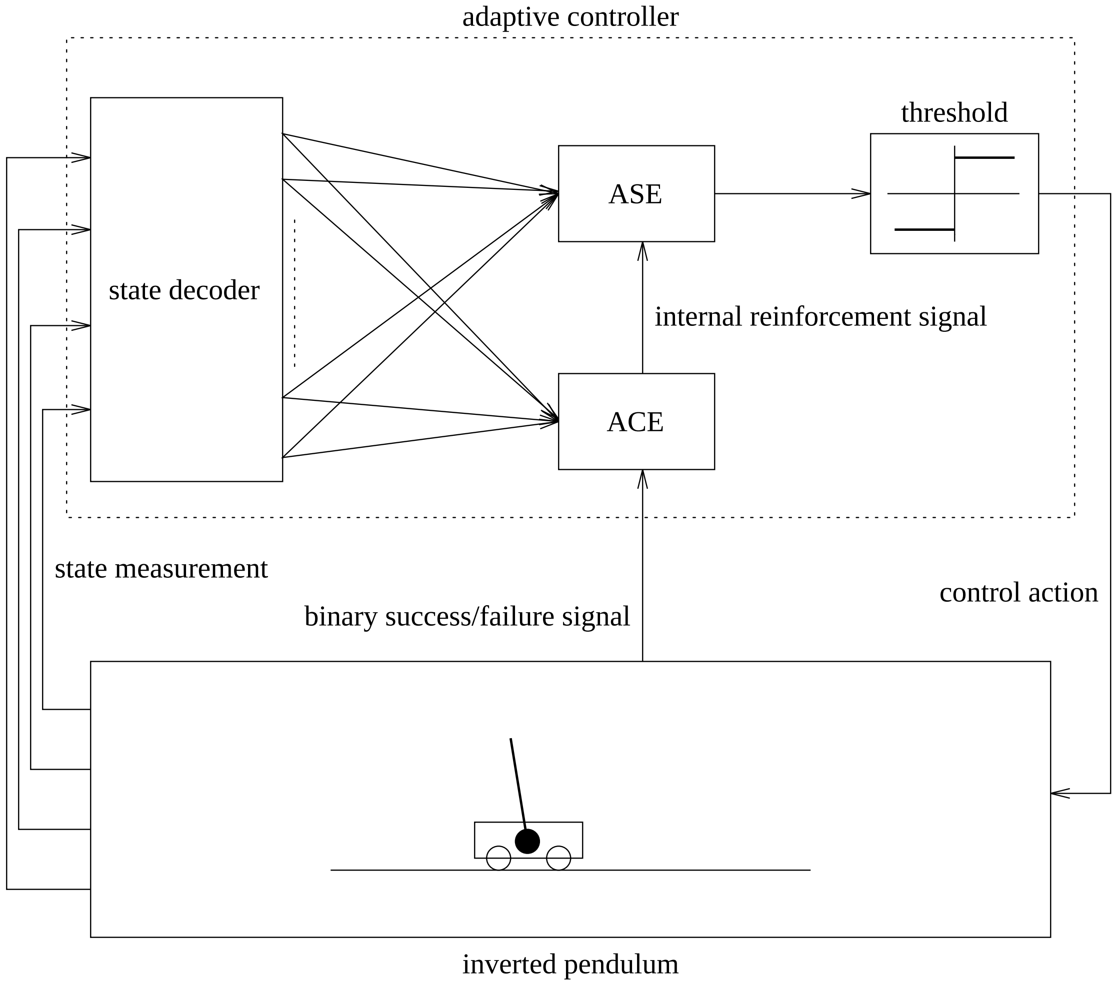


Figure 1: The ACE/ASE controller in place

Full details of the ACE/ASE scheme are given by Barto *et al.* [3]; what follows is only a brief outline of their earlier work. The modifications necessary to apply the scheme to the real apparatus are described later.

2.1 The simulated pendulum system

The simulated pendulum system of Barto *et al.* [3] was much like the real system shown in figure 11. It comprised a straight, horizontal track, like a railway track, with a carriage free to move along it. On the carriage was an axis, perpendicular to the track and pointing out to the side, about which a pendulum was free to turn. The controller's task was to keep the pendulum upright, by alternately pushing and pulling the carriage along the track.

The equations governing the simulation modelled the mass of the carriage and the inertia of the pendulum, under the influence of both gravity and the control force applied to the carriage. Rolling resistance between the carriage and the track was included, as well as friction at the pendulum's pivot. The carriage's position, its velocity, the angle of the pendulum from the vertical and the pendulum's angular velocity made up the state of the system. The simulation equations mapped the system's state and the control action into a linear acceleration of the carriage and an angular acceleration of the pendulum, which were then integrated using a simple Euler technique to generate the next state vector, and so on.

2.2 The ACE/ASE controller

The controller was made up from three elements: a decoder, an *Associative Search Element* (ASE), and an *Adaptive Critic Element* (ACE). The decoder passed the information in the state vector to the ASE, which held the control policy and generated the control action. The ACE judged the performance of the policy, allowing it to be improved. Each of these devices will be considered in turn.

2.2.1 The decoder

The decoder mapped the 4-dimensional state vector of the pendulum system into a 162 element vector, through a partition of the state space into 162 fixed regions, the so-called "boxes" of Michie and Chambers [12]. Each element of the decoder's output vector corresponded to a box within the state space. If a box was occupied by the current state of the system, its corresponding element in the output vector would be set to 1, otherwise it would be set to zero.

2.2.2 The ASE

Each element of the decoder's output vector was linked to the ASE. Each link had a weight. The output $y(t)$ of the ASE was defined as follows:

$$y(t) = f\left[\sum_{i=1}^{n} w_i(t)x_i(t) + \text{noise}(t)\right] \tag{1}$$

where t is a time index, n is the number of weights w_i equal to the number of boxes in the decoder, x_i is the ith element of the decoded state vector, noise(t) is a gaussian noise signal of small-variance and zero mean, used to explore the environment by randomising the policy, and f is a threshold function like an ideal relay. The threshold function is given by:

$$f(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \tag{2}$$

The output of the ASE was scaled by a fixed magnitude and fed to the pendulum system as a control action. The ACE/ASE controller had only two actions to offer, which made it easier to learn to balance the inverted pendulum. When one action was seen to be poor, the sole alternative was to select the other. However, the restricted action set also made the controller's task harder because no null action was available; taking an action also disturbed the pendulum from an upright position.

2.2.3 The ACE

The Adaptive Critic Element (ACE) solved the following *temporal credit assignment* problem. An obvious approach to reinforcement learning is to associate the most recent reinforcement with the most recent action. This means that the negative reinforcement from a collapse of the pendulum penalises the last action. However, more often than not a poor action taken earlier is to blame for the pendulum's collapse — recent actions could even have been optimal for the circumstances. The temporal credit assignment problem is the problem of associating the correct blame or credit with actions separated from their effects by time. Without a solution to this problem, a controller will fail to learn a successful policy.

The ACE solved the temporal credit assignment problem by maintaining what can be thought of as a 'danger level' for each box in the state space. Whenever the system moved towards a less dangerous box, the most recent action would be rewarded; whenever it moved to a more dangerous box, the most recent action would be penalised (thus making an alternative action more likely next time). Actions were rewarded or penalised by respectively increasing or decreasing the relevant weights w_i in the ASE.

The danger map was created using the external (or *primary*) reinforcement signal. Each entry in the map represented a prediction of the external reinforcement to be expected in the future. The control policy was not changed so long as the performance was expected, even if the performance was poor. Only unexpected performance would lead to a change in the ASE's policy.

The simplicity of the external reinforcement signal demonstrated the attractive goal-directed nature of the scheme. The signal was always zero, except if the pendulum was out of bounds (more than 12° from the upward vertical or 2.4 m from the centre of the track), when the reinforcement was -1.

The output of the ACE, the predicted performance $p(t)$, was given by:

$$p(t) = \sum_{i=1}^{n} v_i(t) x_i(t) \tag{3}$$

where x_i represents the same decoder outputs used in equation (1), and where v_i represents a set of link weights for the ACE, similar to those of the ASE.

2.3 Adapting the ASE and ACE

Each weight v_i of the ACE was updated as follows:

$$v_i(t+1) = v_i(t) + \beta \left[r(t) + \gamma p(t) - p(t-1) \right] \overline{x}_i(t) \tag{4}$$

where $r(t)$ is the crude external reinforcement signal, γ and β are positive constants, and $\overline{x}_i(t)$ is a trace of the decoder's output. The trace offered a way of generalising across boxes, and was defined as follows:

$$\overline{x}_i(t+1) = \lambda \overline{x}_i(t) + (1-\lambda) x_i(t) \tag{5}$$

where $0 \leq \lambda < 1$ determines the rate of trace decay. The trace worked by weighting boxes more heavily if they were occupied more recently.

The internal reinforcement signal used to update the ASE was defined by:

$$\hat{r}(t) = r(t) + \gamma p(t) - p(t-1) \tag{6}$$

The same term appears in brackets in the weight update equation for the ACE (equation (4)).

Normally, the external reinforcement $r(t)$ was zero, and the internal reinforcement $\hat{r}(t)$ was equal to $\gamma p(t) - p(t-1)$. When a collapse occurred, however, the external reinforcement was -1 and, because there was no box in the decoder that corresponded to a collapsed state, the prediction $p(t)$ was zero. Thus at collapse the internal reinforcement would be $-1 - p(t-1)$. This amounted to substituting the prediction $p(t)$ with the real reinforcement $r(t)$ at a collapse.

The internal reinforcement signal updated the ASE weights as follows:

ASE	
δ	0.9
α	1000.0
ACE	
λ	0.8
γ	0.95
β	0.5

Table 1: ACE/ASE constants

$$w_i(t+1) = w_i(t) + \alpha\hat{r}(t)e_i(t) \tag{7}$$

where α is a constant and e_i is a trace, similar to $\overline{x}_i(t)$ above and defined as follows:

$$e_i(t+1) = \delta e_i(t) + (1-\delta)y(t)x_i(t) \tag{8}$$

where $0 \leq \delta < 1$. The ASE traces were reset to zero after a collapse.

The values for the constants used in the ACE/ASE equations are given in table 1.

3 Modifications to the original controller

Modifications to the original ACE/ASE controller of Barto *et al.* [3] were necessary because the real apparatus had different dimensions and masses to the simulation. Most notably, the working track length was shorter (0.6 m against 4.8 m). The differences meant the original ACE/ASE controller failed to keep the real pendulum upright for more than about a second.

The controller successfully balanced the simulation using the masses of the physical apparatus and a 4.8 m track. However, reducing the simulated track length to 0.6 m defeated the controller, producing poor performance similar to that seen on the real rig. Comparing the real rig with the simulation suggested that the simulation was a good approximation. However, this is not necessarily important, since to test a learning controller the simulation need only be a suitably difficult control problem.

3.1 The available hardware

The testbed for the learning algorithm was a control rig designed for an undergraduate control course at Cambridge University Engineering Department. Details are given in appendix A.

The rig was based around a track 1 m in length, along which a carriage was free to move. The carriage was attached by wires to a torque motor mounted on one end of the track. A short, light aluminium rod was free to rotate about a pivot on the carriage. On the end of the rod was a heavy brass weight.

The rig was controlled by a C++ program on an Inmos T800 Transputer, connected to the apparatus via a custom interface designed and built within Cambridge University Engineering Department. The interface board carried four analogue to digital converters (ADCs) for the sensory data and a digital to analogue converter (DAC) for the control action.

Sensory information from the rig comprised angular position and velocity of the pendulum and linear position and velocity of the carriage (i.e. full state information). These were measured using potentiometers and tachometers on both the axis of the torque motor and the pendulum's pivot. The carriage position potentiometer was geared, since the torque motor took several rotations to move the carriage from one end of the track to the other.

3.2 Assessing control feasibility

To determine the feasibility of controlling the pendulum with the available equipment, a number of steps were taken, helped by the existence of a linear state-feedback control expression (see appendix C).

3.2.1 Linear state-feedback

First it was verified that the T800 and interface were fast and accurate enough to balance the pendulum using simple state-feedback. Four control gains were applied to the elements of the system's state to yield a proportional control torque from the torque motor. The result was a robust solution to the pendulum-balancing problem. It was possible to place a coffee mug on top of the pendulum and still observe stability.

The form of the linear state-feedback expression is:

$$u = k_x x + k_{\dot{x}} \dot{x} + k_\theta \theta + k_{\dot{\theta}} \dot{\theta} \tag{9}$$

where u is the control action, x is the carriage position from the centre of the track, $\dot{x}$ is the carriage velocity, θ is the angle of the pendulum from the vertical and $\dot{\theta}$ is the pendulum angular velocity. The subscripted k's are the control gains.

3.2.2 Bang-bang control

To verify the viability of a bang-bang control scheme (the kind of control offered by the ACE/ASE controller), the linear state-feedback control action from above was thresholded to give an output of fixed magnitude but variable sign. Stable control was observed, although not as robust as in the previous proportional case. Both the magnitude of the bang-bang control action and the sampling frequency were altered to discover the ranges of these parameters that led to stability. A sample frequency of 50 Hz and a control action magnitude of 0.08 Nm at the torque motor worked well, and were used for all the subsequent controllers.

The success of the bang-bang scheme verified the existence of a working control policy for a high-resolution boxes scheme. That is, a boxes scheme that partitioned state space to the resolution of the ADCs should have worked. Unfortunately, memory and processor speed preclude so many boxes. Thus the next step was to find a partitioning of state space with fewer boxes that still offered reasonable performance.

3.3 Finding a partitioning scheme

Successful partitioning schemes were found by trial and error. The search was initially fruitless, but was later successful when helped by two heuristic devices. The first avoided the need to wait for the controller to attempt to learn a policy before a partitioning could be rejected or accepted. The second helped to choose sensible partitionings to test.

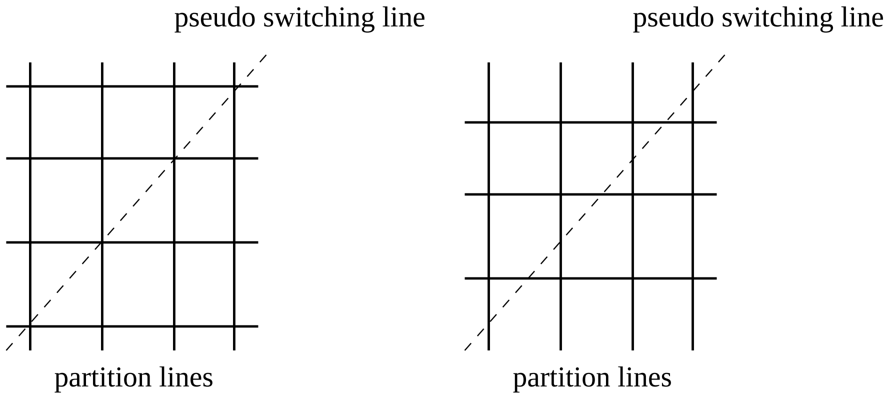
Testing a candidate partitioning of the state space was speeded up by pre-filling each box of the partitioning with the control action that the bang-bang scheme would take at a candidate point in each box. For boxes of finite size, their centres were chosen, and for semi-infinite boxes (those at the edges of the partitioning), points were selected a suitable distance away from their inner edge.

Since each dimension of the state space could be dissected in many ways, the number of reasonable partitionings was large, and it was soon realised that a lot of them failed to work. A heuristic based upon 'pseudo switching lines', detailed below, helped to choose candidate partitioning schemes, and seemed to work well.

3.3.1 Pseudo switching lines

A switching line is a line through two-dimensional state space (or a hyperplane though multi-dimensional state space) which divides regions for which the bang-bang control actions are different. By setting the terms of any two state variables to zero in the state-feedback equation (9), the relationship between the remaining state variables defines a rough approximation to a switching line, a 'pseudo switching line'.

It was reasoned that one way of choosing partitioning schemes would be relative to these pseudo switching lines. Two possibilities presented themselves: either the boxes themselves could be placed corner-to-corner to approximate the lines, like a string of diamonds, or the edges of the boxes could be set like a staircase through the lines. The two patterns are shown diagrammatically in figure 2.



(a) diamonds (b) staircase

Figure 2: Partitions relative to pseudo switching lines

Three partitioning schemes were studied in detail. The first, with 750 boxes, used the staircase pattern. The second had 1296 boxes in a diamond pattern. The last scheme used 864 boxes, based on the diamond pattern but with two fewer partitions on the carriage position axis, the remaining carriage position partitions spread away from the centre.

The pseudo switching line heuristic is weak. Nevertheless, with it a number of partitioning schemes were eventually found to work with a pre-filled control policy. Knowing that for each of these partition schemes, at least one stationary control policy existed (i.e. the policy defined by the pre-filling procedure), it was expected that the ACE/ASE policy learning algorithm would find a control policy to balance the pendulum.

3.4 Processing speed

Real apparatus requires real time control. A feature of this work is that both control and learning were in real time. The sample frequency, the rate at which the controller was updated and the rate at which new control actions were issued were all 50 Hz. The earlier ACE/ASE experiments [3] simulated the same rates.

Because the ACE/ASE algorithm required only coarse resolution of the system's state, real-time implementation was straightforward. The boxes scheme only required a small number of comparison operations between elements of the state vector and partitions of the state space axes. A control algorithm that needed to know distances between the plant's state and some datum points in state space would be more computationally demanding.

3.4.1 Too many boxes

The transputer was only able to process around 350 boxes within a sampling interval. Although the sampling could be reduced a little, runs with the pre-filled boxes showed that 50 Hz was better than, say, 45 Hz. The computational demands had to be slackened.

Most of the processing was needed to update the traces on the links between the ACE, ASE and decoder. Since the trace variables quickly become small, it was decided to operate the weight update scheme on only the last 200 boxes visited by using a circular queue. If a box was visited more than once in the last 200 samples, it would be entered in the queue more than once. A single pass through the queue would therefore update the parameters of that box more than once. Although the trace would decay faster, the reinforcement would be applied more often. These effects seemed to balance; a more involved rule was not deemed necessary.

An alternative solution would have been to parallelise the algorithm, an opportunity that could still be realised using more than one transputer. However, one transputer and the adjustments in the algorithm described here were seen to give suitable results. Since the purpose of the work was to verify that the pole could be balanced, the parallelisation project was seen as work for the future.

3.4.2 Control action delay

Delay between measuring the state of the pendulum system and delivering a control action might have had an effect on the ability of the controller to balance the pendulum. In order to gauge how significant the effect might be, a delay was built into a pre-filled controller. A delay of one sample period between measuring the state and applying the control action was seen to have little effect on the stability of the control. The delay in the learning controller could not be greater than a sample period, and so it was expected that the effect of any delay would not be serious. Significantly poorer performance was noticed as the delay was increased beyond one sample period.

The performance of the pre-filled controller with single sample delay was expected to be bettered both because the delay would be less than a full sample period, and because the learning controller could discover a control policy for the best performance (within the constraints of the state space partitioning). Once released from the fixed policy created for the pre-filled test run, the controller would be expected to find, if one existed, a modified policy that performed better, reducing the effect of delay.

4 Training procedure

Data were collected from a series of experimental runs. A run consisted of a long series of trial balances, and was either terminated after a fixed number of samples, or if performance was poor, after 500 trials. Before a run, the ASE was initialised with random weights, the ACE was initialised with zero weights, and the sensors were calibrated to compensate for drift in their values. (The sensors were not of high quality. Position, for example, was measured using geared potentiometers and not optical tracks.)

4.1 A control trial

A trial was started with the pendulum stationary, hanging down in the middle of the track. The pendulum was then flicked upright by a strong kick from the torque motor, which lifted the pendulum high enough for a linear state-feedback controller to catch it upright and bring it smoothly back to the centre of the track. When the pendulum was upright, stationary and close enough to the centre of the track, the ACE/ASE controller was applied, and learning began.

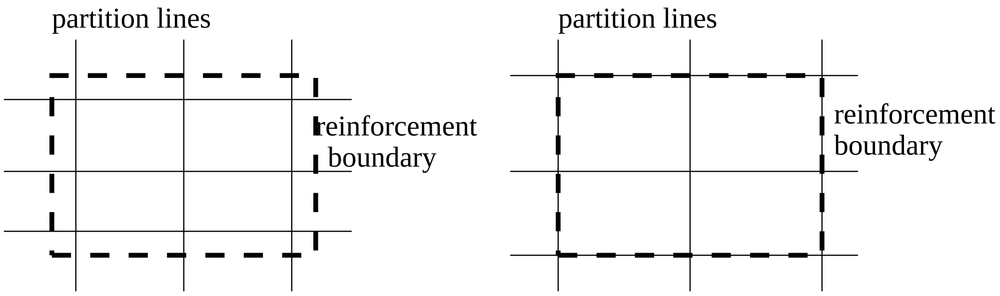
During the ACE/ASE control period, the state of the system was sampled and fed through the decoder to the ASE, which returned an action that was immediately applied to the rig. In the remainder of the sample period, a reinforcement of zero was fed to the ACE, which consequently supplied an internal reinforcement (normally non-zero) to the ASE. All the learning thus took place within a sample period. Finally the processor waited until the next state sample was ready from the ADCs.

ACE/ASE control continued until the system had fallen out of its pre-set bounds (the carriage more than 0.3 m from the centre of the track, or the pendulum more than 12° from the vertical). At this point a park program took control of the pendulum, which moved it to the centre of the rig like a critically damped crane. With the pendulum at rest a reinforcement signal of -1 was sent to the ACE, and its subsequent output then passed on to the ASE. Although this stage could have been performed within a sampling period, due to hardware constraints the next stage could not: the output of trial data to the host system for logging. Once the data was transferred, the pendulum was flicked upright once again, and another trial started.

4.2 Reinforcement and trials

An important difference from Barto *et al.*'s work [3] occurred when the pendulum collapsed. Barto *et al.* had no box that corresponded to the state of a collapsed pendulum, so the external reinforcement was always the same within any box, namely zero. In this implementation, however, the learning scheme was changed to treat the reinforcement signal more generally. Not all tasks can be assumed to break down into trials, and reinforcement is not always discrete. State space partitions can be mis-aligned with reinforcement boundaries, so a box in state space might experience different values of external reinforcement over time.

The partitioning schemes presented here covered all of state space, and did not align with the out-of-bounds boundary in state space. The success of the algorithm under this distortion shows the algorithm's robust nature. Much faster learning is expected with suitable alignment. Also, alignment is reasonable to expect since the reinforcement function is known before learning as the goal-directed specification of the control requirements. The idea of aligning partitions with the reinforcement boundary is shown in figure 3



(a) mis-aligned (b) aligned

Figure 3: Reinforcement boundary and partition lines

5 Results

The performance of the controller was seen to improve as the time between collapses (the trial length) increased during learning. The best performance, a trial of 2100 seconds (35 minutes), was observed using the 864 box decoder after 255 collapses and 5043 seconds (84 minutes) of training time. However, the next longest trial was only half that length. These unusual runs are not seen on the plots below, since the smoothing procedure removes them to show the general trend. In comparison, a random policy would lead to the pole falling out of bounds after around 10 samples (0.2 seconds).

5.1 Learning rate

It was often noted that in simulation the controller would improve its performance dramatically over a short period of time. The same kind of behaviour was seen on the real rig. This is reflected in the roughly linear appearance of the performance plots on log scales.

Runs which started with longer periods of poor performance would lead to better results in the medium term, whereas runs which showed quicker convergence to reasonable performance took much longer to achieve better results. This observation might be explained by a kind of persistent excitation argument: poor initial controllers learn about their environment more quickly than controllers that make mistakes less often.

5.2 Variations in trial lengths

The results show a large variation in trial lengths, both within and between runs using the same controller. This variation is partly a function of the bang-bang control used, as can be seen with reference to figure 4, which shows runs for which learning was turned off after 200 collapses. In comparison to the other results where learning was not stopped, the figure also shows the positive effect of continued learning.

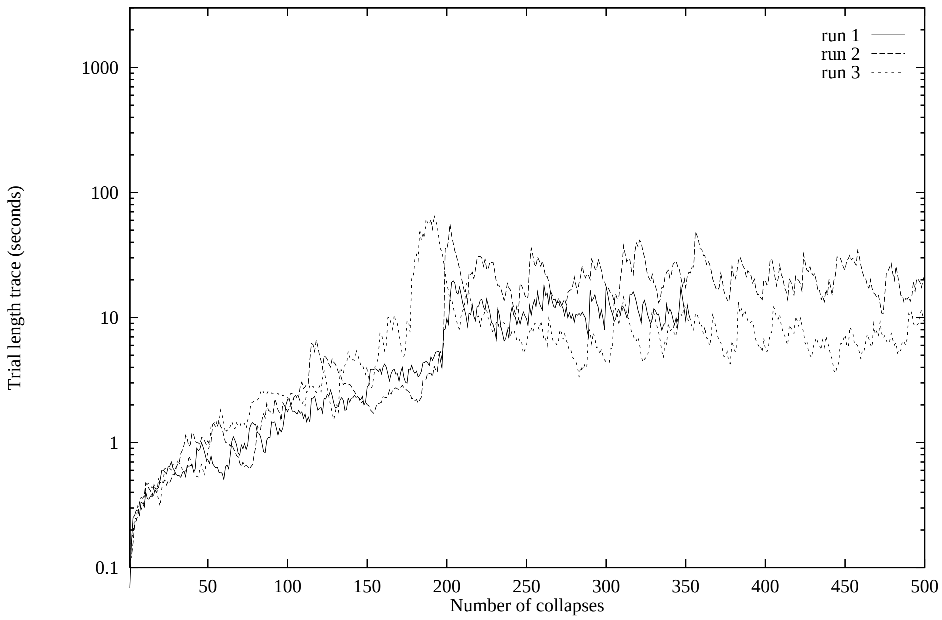


Figure 4: Three runs using an 864 box controller, with learning stopped after 200 collapses

Two of the runs in figure 4 show a poorer performance after 200 collapses (when learning was turned off) than that just prior to 200 collapses. This is because the ASE changes policy as it learns. Such changes may not always be for the best, and thus the performance can fall. With learning stopped, the policy selected just before learning was switched off becomes are permanent feature, along with its possibly poorer performance.

The other plots (where learning was not stopped) indicate a slightly larger variation in trial length as performance generally improves. An explanation of this might be that the number of control policies is large, but only a few perform well. Early in the learning phase, the controller changes between several similar policies, but since each is of poor performance, the variation in trial length is small. After learning has progressed, although some of the policies the controller invokes will be of much higher quality, poor policies may still linger. The general performance is better, but the variation is larger.

Randomisation of the ASE's initial weights affected the initial conditions of a learning run, which helps to account for differences between runs. However, similar differences were also seen between runs starting with identical weights. Sensor drift and noise are therefore assumed to account for most of the variation between runs.

5.3 Smoothing the graphs

Graphs of the raw results show a lot of variation between the lengths of trials, even after learning for some time (figure 5 is typical). Thus in order to better discern a trend in the data, and to make a composite plot of several runs comprehensible, a filtered version of the trial lengths was plotted. The graphs show the value of a trace variable of trial length against the number of collapses. The trace variable t is defined as follows:

$$t(n) = (1 - \tau)t(n - 1) + \tau d(n) \tag{10}$$

where $d(n)$ is the length of the nth trial. τ was set to 0.2 for all the plots. This exponential weighting of the results avoids the discontinuity of a binning procedure. The difference the trace makes may be seen by comparing the raw data of figure 5 with a traced version of it in figure 6.

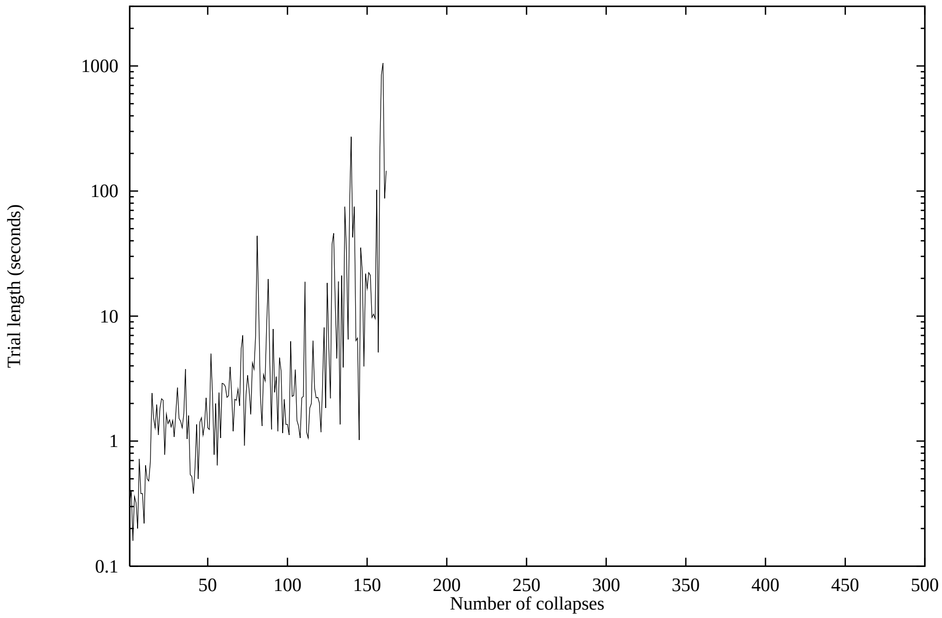


Figure 5: 200,000 sample run using 864 boxes (full data)

This paper lacks a serious statistical analysis of the results. Such an analysis would be of value to judge the comments made earlier on the controller's performance, and variations in that performance. The opportunity to carry out such analysis is left to the future. The results given here are expected to convey (1) that reasonable performance was achieved with the controller, and (2) that the configuration of the decoder (the state space partitioning) had an impact on that performance.

5.4 Changing the decoder specifications

Figure 7 shows the results of applying almost the same partitioning as the original simulation work. Because of the different track length, the carriage position partitions were moved to keep the same ratio with carriage position bounds (4.8 m in simulation, 0.6 m for the real apparatus). Reasonable performance was not observed with this partitioning, nor for any of several alternative partitioning schemes tried using 162 boxes.

Figure 8 is the result of a boxes scheme that attempted to re-create the rough switching lines outlined in section 3.3.1 using box edges in a staircase pattern.

Figure 9 shows the results of three runs with a 1296 box controller. This box scheme covered the pseudo switching lines in a diamond pattern.

Figure 10 shows three 400,000 sample runs using a decoder with 864 boxes. This decoder is similar to the 1296 decoder, but uses fewer boxes over the carriage position dimension (giving less

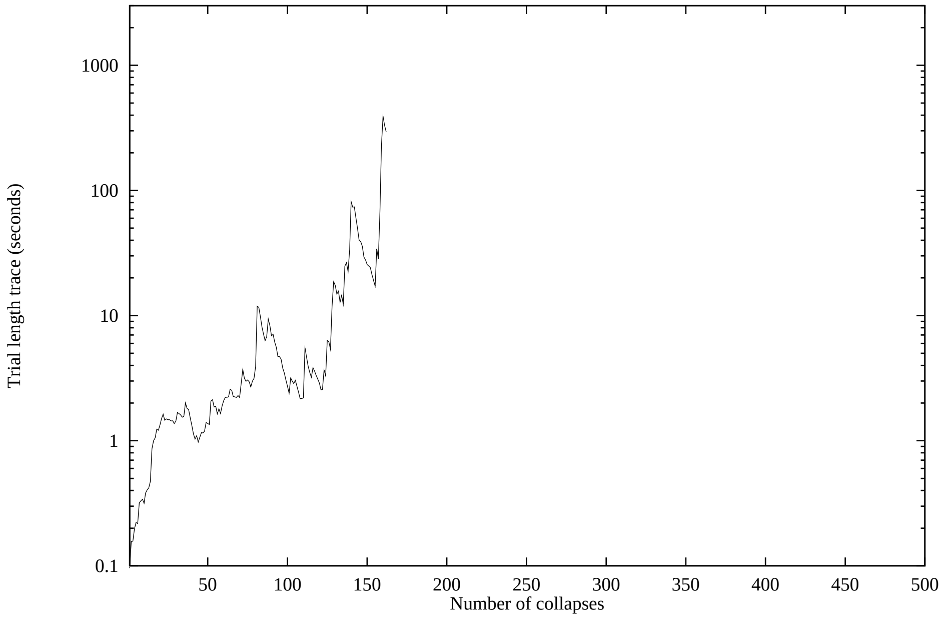


Figure 6: 200,000 sample run using 864 boxes (trace)

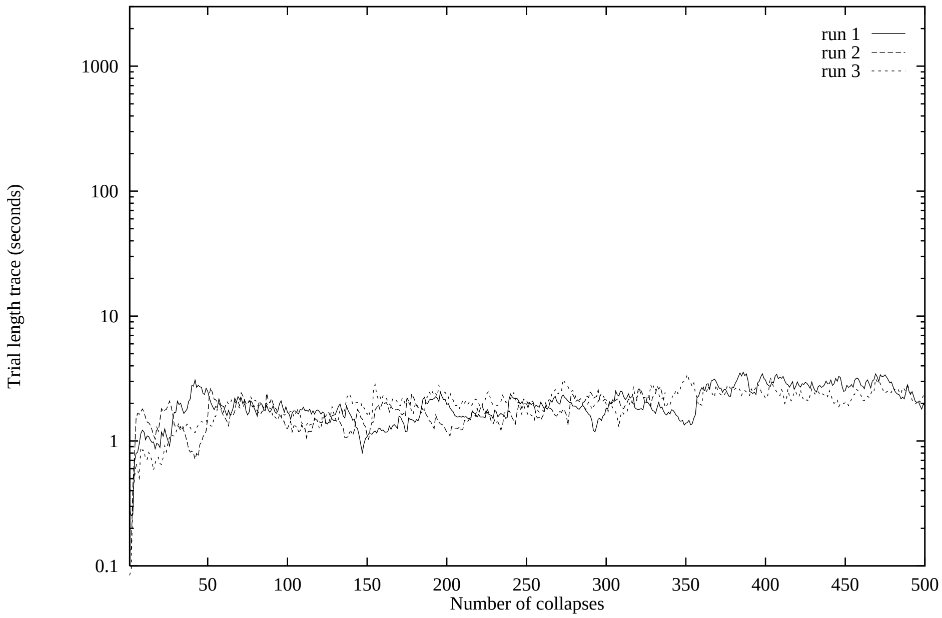


Figure 7: Runs using 162 boxes (trace)

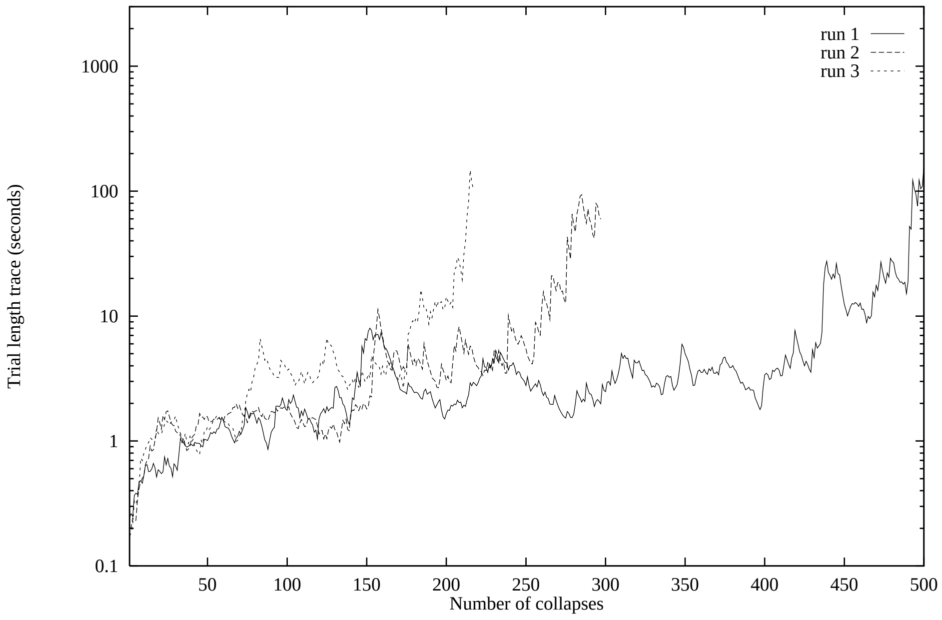


Figure 8: Runs using 750 boxes (trace)

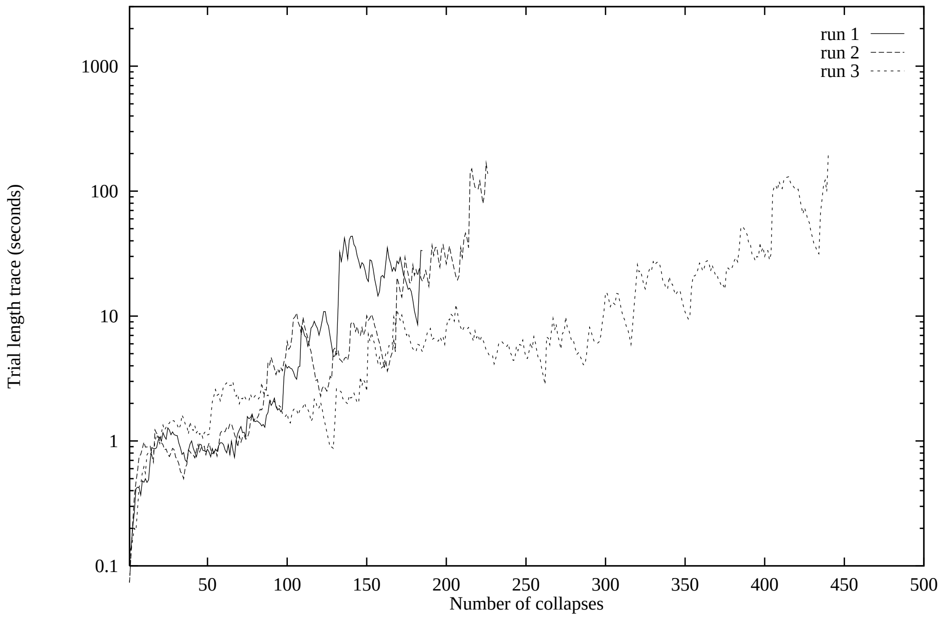


Figure 9: Runs using 1296 boxes (trace)

resolution but more generalisation). The lines in the plots are of different lengths because the 400,000 samples were distributed over the trials differently for each run.

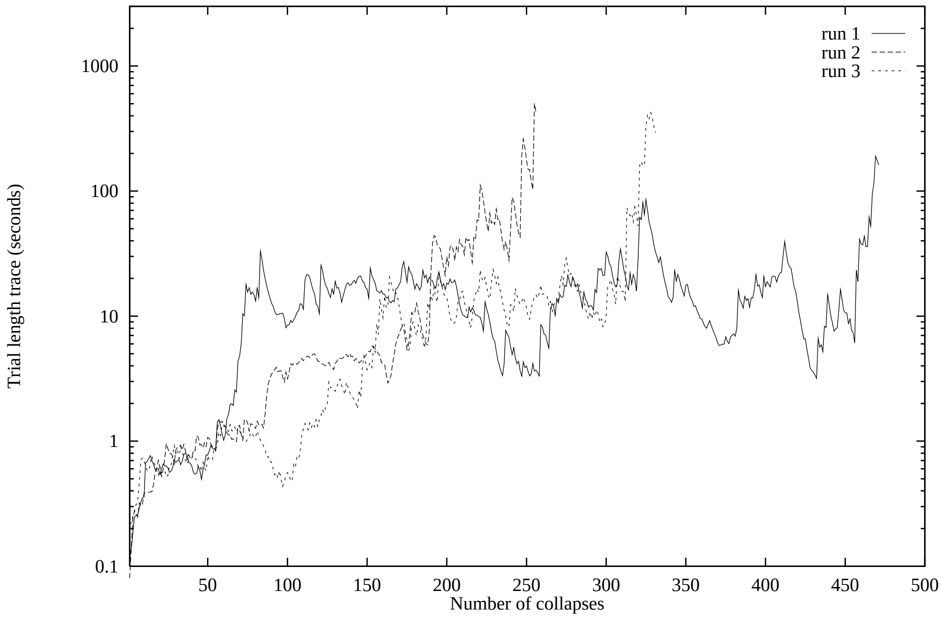


Figure 10: 400,000 sample runs using 864 boxes (trace)

5.5 Comparison with the original ACE/ASE work

It is not possible to directly compare these results with the earlier results of Barto *et al.* [3] because of the difference in the plant parameters. Other differences include the reduced number of link weight updates to decrease the computational load, and more significantly the divergence in the treatment of the reinforcement signal through the mis-alignment of the state partitions and the reinforcement boundary.

6 Conclusions

This work verified the applicability of a reinforcement learning controller to a real hardware, real-time task. The task is too simple and contrived to be of any practical use in itself, but represents a step towards the solution of more complex problems.

6.1 The difficulty of pendulum balancing

The existence of a linear state-feedback controller that is stable and robust suggests that balancing the pendulum is not difficult. A more difficult problem, with a correspondingly more interesting solution, arises when incomplete state information is used. Hecht-Nielsen [8] cites Tolat and Widrow attempting this more difficult problem. They used a video image of the inverted pendulum system as the input to their inverted pendulum controller, trained off-line using a simulation. Even in its more straightforward form, however, the inverted pendulum problem still has a use in assessing learning controllers, since a learning algorithm of satisfactory flexibility and speed has yet to be applied to the problem.

6.2 State space partitioning

The partitioning scheme had to reconcile two conflicting demands. So the controller could use its experience in novel situations, the boxes in the decoder needed to be large and few. However, to resolve the state space sufficiently well for a working control policy, the boxes needed to be small and numerous.

The search for a suitable partitioning scheme would have been hard without the linearised state-feedback control expression to choose, and especially to verify, various candidates. Unfortunately, the need for such prior knowledge largely defeats the purpose of learning. It is a consolation that, if it means a policy can be learnt which works better than the pre-filled policy, learning might still be worthwhile. A better policy might be learnt that took account of delays in the control loop, for example. The effect of delay was explored, however, in section 3.4.2, and was not found to be significant for this particular task.

Although aligning the state partitions with the reinforcement boundary might have improved the learning rate, it would have made no difference to the pre-filled test runs using various partition schemes. These tests suggest that successful partitioning schemes with less than around 1200 boxes are rare.

6.3 Exploration

The ACE/ASE controller must explore to find a good control policy. The original scheme of Barto *et al.* [3] explored by including a noise term in its determination of control actions. However, the noise was small and did not have a significant effect — the controller worked just as well without it. This suggests that the policy learning task is not hard. The ACE/ASE scheme applied to the real rig also worked with no noise added to the control decision.

Although the difficulty of partitioning the state space was overcome for this particular implementation, the problem still stands in the way of a more general utilisation of policy learning schemes. Its solution requires more work, which is easier in simulation.

There are more powerful learning schemes available, such as the neural network schemes used by Jordan and Jacobs [11] and Anderson [1], that avoid the need for an explicit state space partitioning. Unfortunately, these schemes require a large number of training examples. Anderson's results [1] show that around 6000 failures are needed before performance improves, a magnitude greater than the number of failures necessary for the ACE/ASE scheme to give suitable performance. Although this is to be expected of a control scheme that has more to learn, the thousands of examples necessary for the neural network solutions impose severe demands on any physical plant, motivating a search for a more efficient learning scheme.

A learning controller that is successful with more complex problems needs to be more powerful. Learning more control parameters efficiently requires marrying exploration of the plant with a utilisation of the plant's resources, a non-trivial problem known in the field of control as *dual control* [7] and in artificial intelligence as the *two-armed bandit problem* [5].

7 Future work

The demands of real hardware require a controller to make as much as possible of its experience. The problem of efficient exploration is difficult, so progress towards a better learning controller will be slow. Even the simply-stated two-armed bandit problem [5] is yet to be fully solved [15].

The use of probability to express uncertainty offers interesting possibilities, however. A controller with multiple models of the environment, weighted in probability, might be used to tackle the inverted pendulum problem. Multiple candidate models would avoid the need for detailed prior knowledge of the system, whilst giving the opportunity to use any prior knowledge available. Updating the model probabilities using Bayes' Theorem would make the most of the controller's experience. Probability, expressing uncertainty, gives the opportunity for rationalised exploration. The possibility of employing a Bayesian multiple-model learning controller to the inverted pendulum problem might be studied.

Symbol	Description	Value
L	Working track length	0.80 m
l	Pivot to pendulum centre of mass distance	0.125 m
a	Pulley radius	0.016 m
m	Pendulum mass	0.230 kg
M	Trolley mass	0.700 kg
I	Motor shaft's moment of inertia	80×10^{-6} kg m^2
k_m	Torque motor constant	0.080 Nm A^{-1}
k_a	Amplifier constant	-0.50 A V^{-1}

Table 2: Physical constants of the apparatus

This work suggests that policy learning controllers need to be more adaptive before they can be more widely applied. Neural network approaches can offer adaptability but work needs to be done to improve their learning speeds.

Acknowledgements

This work has been enhanced through the contributions of Georges Wong, Chen Tham, Andrew Senior, Steve Hodges, David Mackay, Tony Robinson and Patrick Gosling. It would have been more difficult without the technical support of Patrick Gosling, Andy Piper, Richard Prager, Dave Gautrey, Alan Thorne and many other members of Cambridge University Engineering Department.

A Hardware

Figure 11 shows the configuration of the experimental apparatus, which was designed and built within Cambridge University Engineering Department. Table 2 details certain constants for the carriage/pendulum hardware. An important feature of the apparatus was its robustness, a great advantage during the long experiment runs.

The force required to move the carriage along the track at constant low velocity ranged between 1 and 3 N, a loose indication of the size and non-uniformity of the friction along its length.

To protect the equipment, a hardware cut-out on the rig stopped the carriage smashing into the end-stops of the track. Unfortunately, once the cut-out was activated, the rig was beyond the control of the transputer, so a safety margin was introduced into the software to ensure the plant stayed within safe operating bounds. This further reduced the available length of track, and explains why only 0.6 m of the 1 m long track was used for training.

Figure 12 shows a picture of the apparatus. The analogue power amplifer and sensor amplifiers are in the box behind the track on the left, the transputer and interfaces are in the box on the right. The scale is indicated by the 300 mm ruler under the pendulum in the centre of the track. The pendulum is being held upright in this picture by a linear state feedback controller.

B Decoder partitions

Several state space partitioning schemes were used in the experimental runs. Table 3 details the partitions used in the first attempt to balance the inverted pendulum. This partitioning differs from the original work of Barto *et al.* [3] by only a linear scaling of the carriage position partitions (the simulated system had carriage bounds at +/- 2.4 m, the real apparatus had carriage bounds at +/- 0.3 m).

Tables 4, 5 and 6 give the partition positions for the 1296, 864 and 750 decoders respectively.

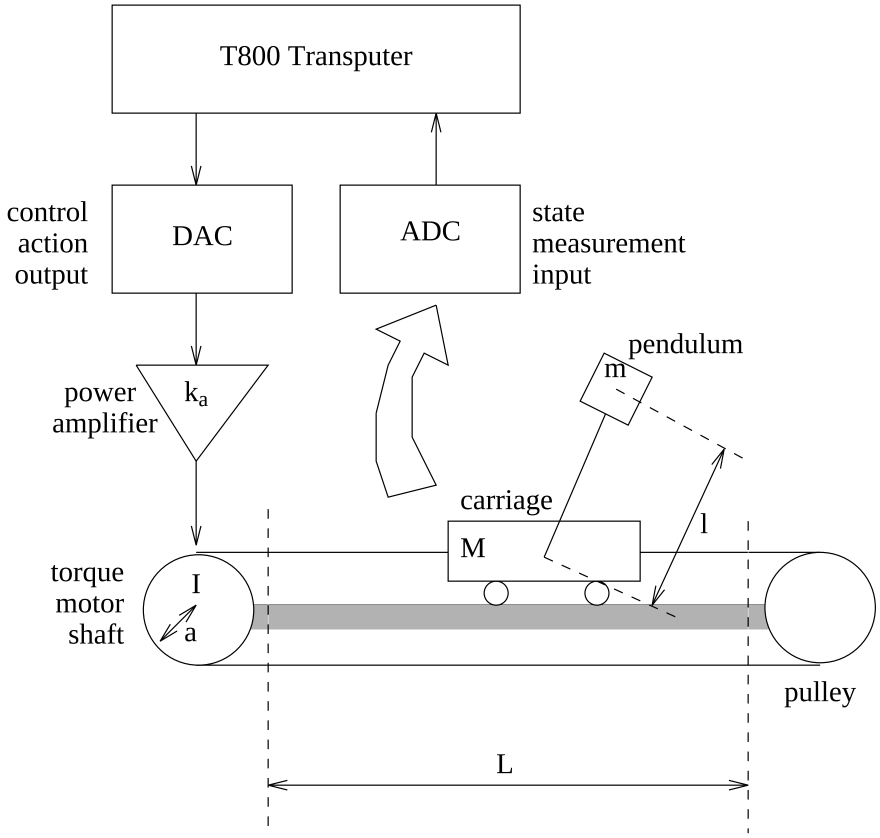


Figure 11: Hardware configuration

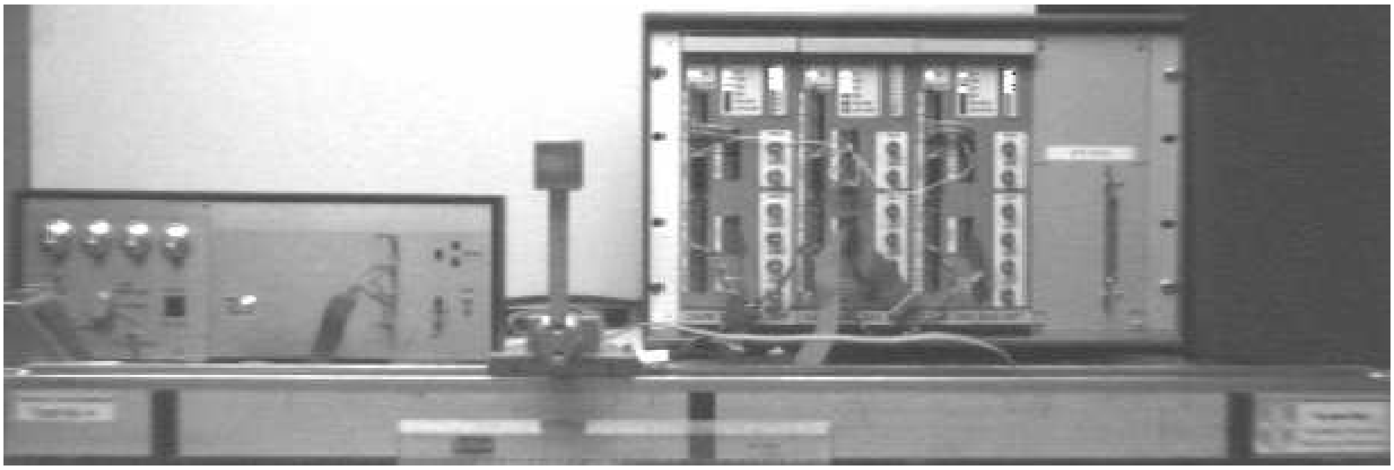


Figure 12: The apparatus

Number of boxes	162
Pole position partitions	
1	$-6°$
2	$-1°$
3	$0°$
4	$1°$
5	$6°$
Pole velocity partitions	
1	$-50°$ s^{-1}
2	$50°$ s^{-1}
Carriage position partitions	
1	-0.1 m
2	0.1 m
Carriage velocity partitions	
1	-0.5 m s^{-1}
2	0.5 m s^{-1}

Table 3: 162 box decoder parameters

Number of boxes	1296
Pole position partitions	
1	$-6°$
2	$-2°$
3	$0°$
4	$2°$
5	$6°$
Pole velocity partitions	
1	$-68°$ s^{-1}
2	$-23°$ s^{-1}
3	$0°$ s^{-1}
4	$23°$ s^{-1}
5	$68°$ s^{-1}
Carriage position partitions	
1	-0.22 m
2	-0.07 m
3	0 m
4	0.07 m
5	0.22 m
Carriage velocity partitions	
1	-0.27 m s^{-1}
2	-0.09 m s^{-1}
3	0 m s^{-1}
4	0.09 m s^{-1}
5	0.27 m s^{-1}

Table 4: 1296 box decoder parameters

Number of boxes	864
Pole position partitions	
1	$-6°$
2	$-2°$
3	$0°$
4	$2°$
5	$6°$
Pole velocity partitions	
1	$-68°\ s^{-1}$
2	$-23°\ s^{-1}$
3	$0°\ s^{-1}$
4	$23°\ s^{-1}$
5	$68°\ s^{-1}$
Carriage position partitions	
1	-0.15 m
2	0 m
3	0.15 m
Carriage velocity partitions	
1	-0.27 m s^{-1}
2	-0.09 m s^{-1}
3	0 m s^{-1}
4	0.09 m s^{-1}
5	0.27 m s^{-1}

Table 5: 864 box decoder parameters

Number of boxes	750
Pole position partitions	
1	$-6°$
2	$-2°$
3	$0°$
4	$2°$
5	$6°$
Pole velocity partitions	
1	$-45°\ s^{-1}$
2	$-11°\ s^{-1}$
3	$11°\ s^{-1}$
4	$45°\ s^{-1}$
Carriage position partitions	
1	-0.15 m
2	-0.04 m
3	0.04 m
4	0.15 m
Carriage velocity partitions	
1	-0.18 m s^{-1}
2	-0.05 m s^{-1}
3	0.05 m s^{-1}
4	0.18 m s^{-1}

Table 6: 750 box decoder parameters

C Linearised state-feedback

Analysis of the pole system shown in figure 13 gives the following equations, ignoring friction:

$$
\begin{aligned}
l\ddot{\theta} &= g\sin\theta + \ddot{x}\cos\theta && (11)\\
\left\{\frac{M}{m} - \frac{I}{ma^2} + 1\right\}\ddot{x} &= \frac{T}{ma} - l\ddot{\theta}\cos\theta + l\dot{\theta}^2\sin\theta && (12)
\end{aligned}
$$

where l is the length of the pendulum, θ is the angle of the pendulum from the vertical, g is the acceleration due to gravity, x is the position of the carriage along the track, M is the mass of the carriage, m is the mass of the pendulum, a is the radius of the torque motor shaft, I is the moment of inertia of the torque motor about its axis, and T is the torque supplied by the motor. Signs for the quantities are in the sense depicted in figure 13.

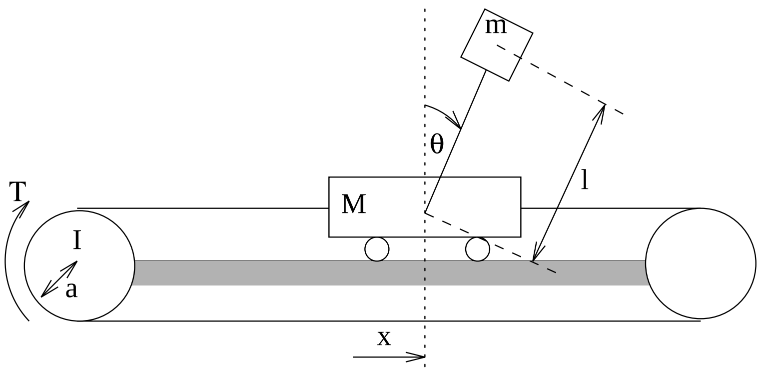


Figure 13: Pole dynamics

Linearising equations (11) and (12) for small values of θ gives

$$
\begin{aligned}
l\ddot{\theta} &= g\theta + \ddot{x} && (13)\\
\left\{\frac{M}{m} - \frac{I}{ma^2} + 1\right\}\ddot{x} &= \frac{T}{ma} - l\ddot{\theta} && (14)
\end{aligned}
$$

which may be re-arranged and expressed in matrix form:

$$
\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{g}{J+1} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{gJ}{J+1} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ma(J+1)} \\ 0 \\ \frac{1}{ma(J+1)} \end{bmatrix} T \qquad (15)
$$

where

$$
J = \left\{\frac{M}{m} - \frac{I}{ma^2} + 1\right\} \qquad (16)
$$

Linear state-feedback fixes the torque in equation (15) to be a linear combination of the elements of the state vector $\mathbf{x}$, where

$$
\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \qquad (17)
$$

k_x	1.15 N
$k_{\dot{x}}$	0.93 N s
k_θ	2.4 N m
$k_{\dot{\theta}}$	0.21 N m s

Table 7: Linearised state-feedback gains

Thus T is set to

$$T = \begin{bmatrix} k_x \\ k_{\dot{x}} \\ k_\theta \\ k_{\dot{\theta}} \end{bmatrix}^T \mathbf{x} \tag{18}$$

and equation (15) can be expressed in the form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \tag{19}$$

The positions of the eigenvalues of $\mathbf{A}$ in the complex plane characterise the stability of the linearised system. So long as the eigenvalues are in the left half plane the linear system is globally asymptotically stable. A range of values for the elements of $\mathbf{k}$ results in suitable eigenvalue placement. The values of $\mathbf{k}$ chosen to set the decoder partitions are given in table 7.

References

[1] Charles W. Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, pages 31–37, April 1989.

[2] K.J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley series in electrical engineering: control engineering. Addison-Wesley, Reading, Massachusettes, 1989.

[3] A.G. Barto, R.S. Sutton, and C.W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13:834 – 846, Sept/Oct 1983.

[4] Hamid R. Berenji and Pratap Khedkar. Learning and tuning fuzzy logic controllers through reinforcement. *IEEE Transactions on Neural Networks*, 3(5):724–740, September 1992.

[5] Donald A. Berry and Bert Fristedt. *Bandit Problems: sequential allocation of experiments*. Monographs on statistics and applied probability. Chapman and Hall, London, 1985.

[6] W.F. Clocksin and A.W. Moore. Experiments in adaptive state-space robotics. In A.G. Cohn, editor, *Proceedings of the 7th Conference of the Society for Artificial Intelligence and Simulation of Behaviour*, pages 115–125. Morgan-Kaufmann, 1989.

[7] A.A. Feldbaum. Dual-control theory I. *Automation and Remote Control*, 21(9):874–880, April 1961. Translation of "Avtomatika i telemekhanika", Academy of Sciences, Moscow, USSR, September 1960.

[8] Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley Publishing Company, Inc., Reading, Massachusettes, 1990.

[9] W. Thomas Miller III. Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-19:825 – 831, July/August 1989.

[10] J.-S. Roger Jang. Self-learning fuzzy controller based on temporal back-propagation. *IEEE Transactions on Neural Networks*, 3(5):714–723, September 1992.

[11] Michael I. Jordan and Robert A. Jacobs. Learning to control an unstable system with forward modelling. Technical report, M.I.T., Brain and Cognitive Sciences, M.I.T., Cambridge, MA 02139, U.S.A., 1990.

[12] D. Michie and R.A. Chambers. Boxes: An experiment in adaptive control. In E. Dale and D. Michie, editors, *Machine Intelligence*, volume 2, pages 137–152. Oliver and Boyd, 1968.

[13] Andrew W. Moore. Fast, robust adaptive control by learning only forward models. In J.E.Moody, S.J.Hanson, and R.P.Lippmann, editors, *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, Dec. 1991.

[14] D. Nguyen and B. Widrow. The truck backer-upper: An example of self-learning in neural networks. In *Proceedings of the International Joint Conference on Neural Networks, Volume 2*, pages 357–363, Washington D.C., June 18-22 1989.

[15] E.L. Presman and I.N. Sonin. *Sequential Control with Incomplete Information.* Economic Theory, Econometrics and Mathematical Economics. Academic Press Limited, London, 1990.

[16] Chen K. Tham and Richard W. Prager. Reinforcement learning for multi-linked manipulator control. Technical report, Cambridge University Engineering Department, Cambridge University Engineering Department, Trumpington Street, Cambridge, England, July 1992.

[17] Christopher J.C.H. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, May 1989.