Controlling a Steel Mill with BOXES

Michael McGarity, Claude Sammut and David Clements

The University of New South Wales

Abstract

We describe an application of the BOXES learning algorithm of Michie and Chambers (1968) to a large-scale, real-world problem, namely, learning to control a steel mill. By applying BOXES to a model of a skinpass mill (a type of steel mill), we find that the BOXES algorithm can be made to produce a robust controller relatively quickly. Various aspects of the BOXES algorithm are adapted for the to higher dimensionality and noise present in the skinpass mill. These changes are critically examined to find those which give a better controller.

1 Introduction

Boxes began as an exploration into the possibility that many small tasks may be easier for a computer to learn than one large one. That is, it was thought that by breaking up a complex problem, difficult to solve as it stood, into many smaller and more tractable problems, the original problem could be solved with greater speed or ease. Although some information is always lost by splitting the problem into sub-problems, it was hoped that the advantages gained with the improvements in complexity would offset this. The heart of the BOXES algorithm is that a simple, decision-array control strategy is altered by an incremental process based on the success or failure of the controller on the last trial.

The BOXES algorithm has traditionally been applied to unstable, linearisable, low noise, single input plants such as the pole and cart (Michie and Chambers 1968). We might therefore ask what changes we might have to make to adapt the algorithm to be suitable for a larger class of tasks. Some of the issues to be considered are as follows.

- A much larger action space due to multiple inputs will make it harder to learn to choose a good action within a reasonable time.
- Most viable controllers for physical systems need to minimise the number of switches sent to the actuators, as this behaviour carries with it high running and maintenance costs.
- Typical industrial plants are designed to be stable and therefore, the plant will not present examples of marginal failure to the controller

during learning. The large amount of noise usually present may offset this effect.

To explore these problems, we apply BOXES to a model of a working steel mill. The skinpass mill is a plant designed to flatten a strip of steel. It does this by passing the strip between two rollers which are forced together. The aim of this process is to improve certain physical properties of the strip, such as uniform stretchability. This means that much of the deformation due to the rollers occurs in the surface (or skin) of the strip, giving a highly non-linear relationship between force and elongation. The skinpass mill is therefore a non-linear plant with multiple inputs and outputs, with all of the inputs and outputs linked. The skinpass mill is designed to be relatively stable.

2 The Skinpass Mill

The reduction of the steel strip applied by the skinpass mill is very small, (usually less than 5%), and needs to be controlled to within fine limits. The result of this small reduction is to improve the yield point properties of the thin strip product. In terms of yield point flattening, a temper rolling mill is different to a hot rolling mill, which may perform reductions of 50% on very thick steel ingots or plate, with the aim of reducing the thickness of the strip, plate or ingot. Thus the skinpass mill is one of the final stages in the rolling of the steel strip and has different requirements to the earlier processes. In addition to this, the physical processes involved in skinpass rolling are not as well understood as either hot or cold rolling is, which makes the mathematical model needed for control purposes harder to find (Roberts 1972). Naturally, the mill stand is only a small part of the mill, but as this is the object of most of the control design, we will concentrate our attention on it. The primary aim of the control task is to keep the elongation as close as possible to the *setpoint*, or desired level of elongation, while keeping the other parameters (roll tilt and strip shape) within acceptable bounds.

The skinpass mill has three inputs and outputs. The outputs are elongation (related to the average of the main roll force), Roll Tilt (related to the difference in the roll forces) and the roll bending, (related to special roll bending actuators) as shown in Figure 1.1. The relationships between these three main systems is complex, and is difficult to describe analytically. Therefore, we modelled the mill using a static non-linear block in series with a simple linear second order dynamic system. The non-linear element was found using steady-state empirical data. The three sub- systems, elongation, crown and roll tilt were treated as separately as possible.

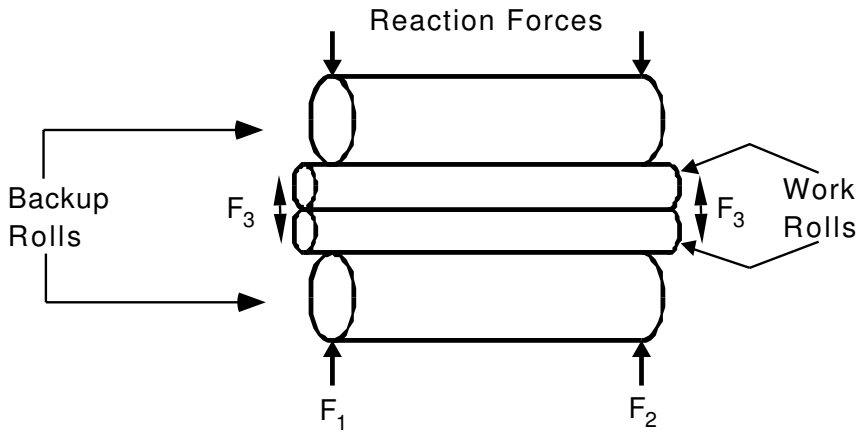

FIG. 1.1. Inputs and outputs of the mill

2.1 Actuators

The *actuators* apply pressure to the cylinders which, in turn, transfer force to the strip. The effect of the actuators to alter the strip shape is effectively instantaneous.

2.2 Measurements

Thinning the strip of steel results in *elongation* of the strip. Distortion in the strip can cause *roll tilt*, which will result in buckle. Heating of the strip as a result of rolling causes expansion and more thinning in the middle. This is called *negative crown*.

2.3 Noise

The noise is mainly due to roll eccentricities and strip irregularities and so the bandwidth of the noise is very closely linked to the strip speed.

Some mill dynamics are fast, with an open loop step disturbance lasting about 5ms. These are mostly hydraulic resonances (damped by gas cylinders) and are ignored in the current implementation. This is because we cannot control them without a dedicated controller and because they die out in between step inputs from the BOXES controller. Note also that in the commercial version of this mill controller, a dedicated controller implementing a PID controller is used to control fast valve and hydraulic dynamics.

The dynamics that are of interest to us concern the shape and elongation of the steel strip and are much slower. The strip runs through the mill at speeds between 30 metres/min and 400 metres/min. At the fastest speed (1.2 mm strip) undulations in strip thickness are caused by elliptical flattening of the rolls. The work rolls are smaller and so contribute a higher

frequency (although a lower amplitude). In the experiments described in this paper, a strip thickness of 4 mm is always used, with a corresponding strip speed of 150 metres/min. The work rolls have a diameter of roughly 400 mm (circumference of 1250 mm) which corresponds to a disturbance bandwidth of approximately 2HZ. The total disturbances introduced by the irregularities in original strip thickness are limited to about 2-3HZ. A sample time of 100ms is 4-5 times as fast as the fastest plant dynamics, and is therefore reasonable for most cases. The experiments are therefore conducted with a sample time of 0.1 seconds.

3 Boxes and the Skinpass Mill

This section deals with the performance of the BOXES algorithm while learning to control the skinpass mill and the changes that have been made to cope with the increased number of dimensions and noise. Our main aim with these modifications is to improve the robustness of both the controller and the learning agent.

There are three critical elements of a BOXES style algorithm:

- It succeeds by avoiding failure.
- BOXES avoids global failure by changing local variables
- Each local variable is changed independently from every other local variable

The BOXES algorithm relies on a state space representation, in which each input (including dynamic information such as derivatives and integrals) is divided into several partitions. Thus, a given input parameter might be divided into three categories, for example, large negative, near zero, and large positive. Each input may be divided into a different number of divisions. In this way, the divisions of the total space form 'boxes' within which all of components of the state space vector stay inside their respective boundaries.

When applied to the skinpass mill, there are four inputs to BOXES (i.e. outputs from the plant), these are the elongation (and its integral of error), roll tilt, and crown. Each of these four inputs is partitioned, giving $5 \times 3 \times 3 \times 3$ boxes.

The output of the control system is similarly quantised. Each box contains an output, and this output does not change during a control run. The action only changes when the whole system fails. The skinpass mill has three independent actuators: operator side pressure, drive side pressure, and bending pressure. Each of this is quantised into large negative, small negative, zero, small positive and large positive. Thus there is a total of 125 different combinations of actions. This represents a large increase in complexity over the pole and cart which only has two actions.

Time is quantised as well. The current action is treated as a constant output for the duration of each time step, so the model for the plant to be controlled needs to be step invariant. Adjustment of the sample or step time is not part of the learning procedure. During the control run, then, the BOXES algorithm is simply a lookup table.

The goal of learning is to coerce the performance of the closed loop into a heuristically defined specification or boundary of acceptable performance. The way this is done is very simple to describe, but it is difficult to guarantee convergence.

The algorithm performs a local search within a global failure definition. The underlying assumption behind this learning algorithm is that an action output by the boxes has a causal relationship with the success or failure of the global system. However, this relationship is usually not directly causal, instead, it is a probabilistic link. The strength of the link between a box and the outcome depends on the behaviour of the boxes around it and in the case of failure, the time between the activation of the box and the eventual failure. Since the behaviour of the surrounding boxes is difficult to predict and may be seen as somewhere between a random action and the 'correct' action, they must be treated stochastically. Thus the causal link between a given action and the ensuing success and failure would probably depend on the relative certainty with which the box holds its action, and so would change over the course of the learning process.

Sammut and Cribb (1990) claimed that a trade-off exists between speed of learning and the generality of the learned controller. The controller produced by BOXES is not guaranteed to be robust in the sense that it can control the same plant from different starting conditions. This is also true of other reinforcement learning algorithms.

In order to test the robustness of our algorithms, we run each of the modifications, with various noise levels, on two different plants: the skin-pass mill and the pole and cart as described by Anderson (1987). When running the algorithms, we continued for 10,000 trials before resetting the learning algorithm. In order to show the performance over this time, we recorded the highest number of successes *in a row* that has been achieved. By 'success' we mean that the system has been kept stable for 10,000 time steps.

subsectionBackground After each trial when the system fails, the algorithm collects the time indices at which each box is entered. They are collected into one number which indicates the proportion of the failure that is due to the action of currently set in the box. This number, termed $Life$, is a function of the elapsed time between use and failure of the box.

$$Life = \sum_{i=0}^{n} (T_{final} - T_i)$$

The $lifetime$, which gives some indication of the proportion of blame for failure, is an discounted accumulation of the past lifetimes for a particular action. This is done using a sliding average. The number of times that a box is entered during a trial is similarly accumulated.

$$Lifetime' = DK \times Lifetime + Life$$

$$Usage' = DK \times Usage + n$$

This second term is used as the divisor when working out the 'life expectancy' of a given action and as a measure of how much is known about this action.

$$Average\ Lifetime = \frac{Lifetime}{Usage}$$

The average lifetime can be seen as an estimate of the life expectancy of the entire system if this particular box chooses this action. In order to encourage exploration, this average lifetime is modified to bias the choice of action towards those actions with which BOXES has little experience. Thus, we define *merit* as:

$$merit = \frac{Lifetime}{Usage^k} \quad where\ k > 1$$

Several variations of this measure have been used for BOXES. The one above was described by Sammut (1994). These merits are then used to compare the various actions and the one which will most likely avoid failure for the controller, in the long run, is chosen. This choice may simply be taking the action with the highest score,

$$merit_{action} > merit_i \quad \forall i \neq action$$

or may be probabilistic.

$$Probability(action = i) \propto merit_i$$

The probabilistic strategy we use proceeds by choosing a particular action with a probability proportional to it's merit.

Deterministic action choices are based on the maximum score given by the appropriate scoring technique. That is, the action chosen would have the best balance, given the information known at the time, between experience and chance of success. A probabilistic choice of action would most often pick the same choice as the deterministic one, but would have some chance of choosing a different one.

4 Annealing

The deterministic method for choosing actions works well for the pole and cart. This may be because the range of actions is very limited, so the algorithm can obtain experience for the entire range of actions. However, the mill has a large number of actions available, 3 independent actions with 5 choices, giving 125 possible actions. Thus, there is ample scope for a complex decision surface to include local minima. Additionally, it is too large a surface to hope that the BOXES algorithm will gain global knowledge before a local minimum is found. For these reasons, we tried to use a probabilistic notion of action choice.

There are two ideas behind annealing. First, there is a need for constant excitation. It is important, when modelling a process from dynamic data, to excite all of the modes of the process so that the model includes these modes. This is often done by using a white noise input to an unknown plant after which the usual system identification procedures take place. In our simulation, plant disturbances are modelled by pseudo-random noise. However, annealing provides a second input of noise and can also be used for this purpose.

The second reason for using annealing is to prevent the algorithm being caught in local minima. As previously mentioned, each action available to a box must have a corresponding lifetime that is indicative of the action's average time to failure. Thus, each action must have a chance to find out what its time to failure is and with a large number of actions available, this may not be possible. Instead, one or two relatively successful actions take over, not allowing other actions a chance to obtain a statistically large number of example runs. Annealing is different from other ways of combating this problem in that instead of boosting the score of inexperienced actions, annealing simply chooses a random action. Each action has a probability proportional to its score and from this sample space an action is chosen. Thus, the higher scores are chosen more often, but any score can be chosen.

One variation that can be added is to provide a cut off level. Annealing is noise, so it should produce a deterioration in the performance of the plant. After annealing has done its job, namely, to give all of the actions a chance to find their own average time to failure, it can be reduced. This is done in the current simulation by only including those actions that have a score above a certain cut-off level. This level can be fixed or it can be raised as the learning procedure progresses, reducing the noise input by the annealing procedure.

In order to test these ideas, the pole and cart and the skinpass mill were tested with various annealing types and levels. In the three graphs shown, The columns represent fixed annealing at a certain level. The level being shown on the x-axis of Figures 1.2, 1.3 and 1.4. Two types of annealing are shown, constant annealing and reducing annealing .

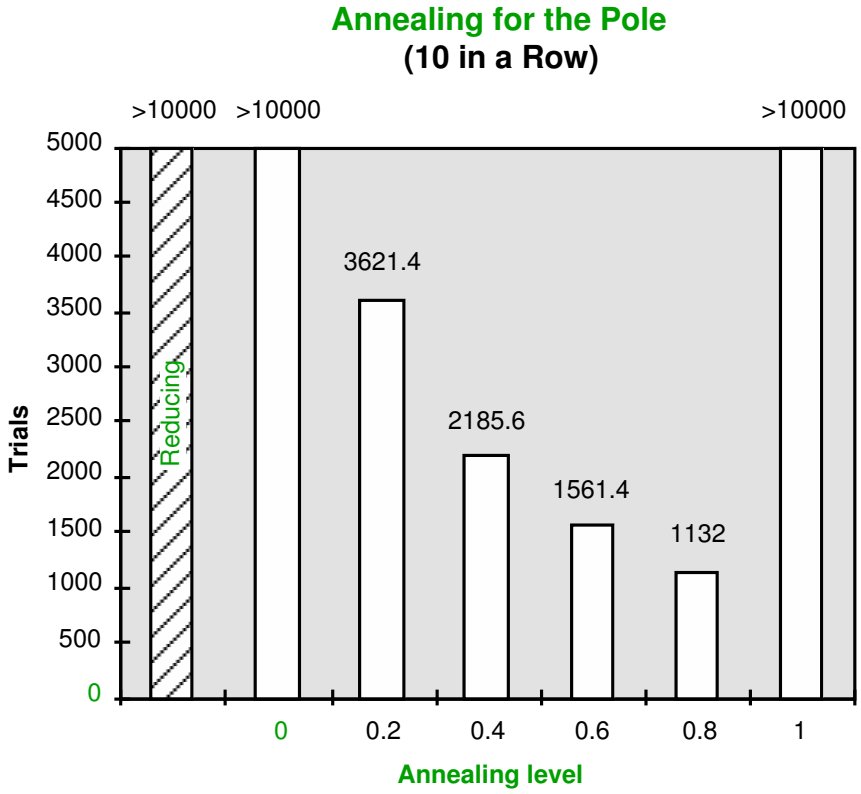

FIG. 1.2. The effect of annealing on learning to control the Pole and Cart. Note that the criterion for success is to balance the pole for 10,000 time steps and repeat that 10 times in a row. The number of trials plotted is the time taken to succeed for the first time.

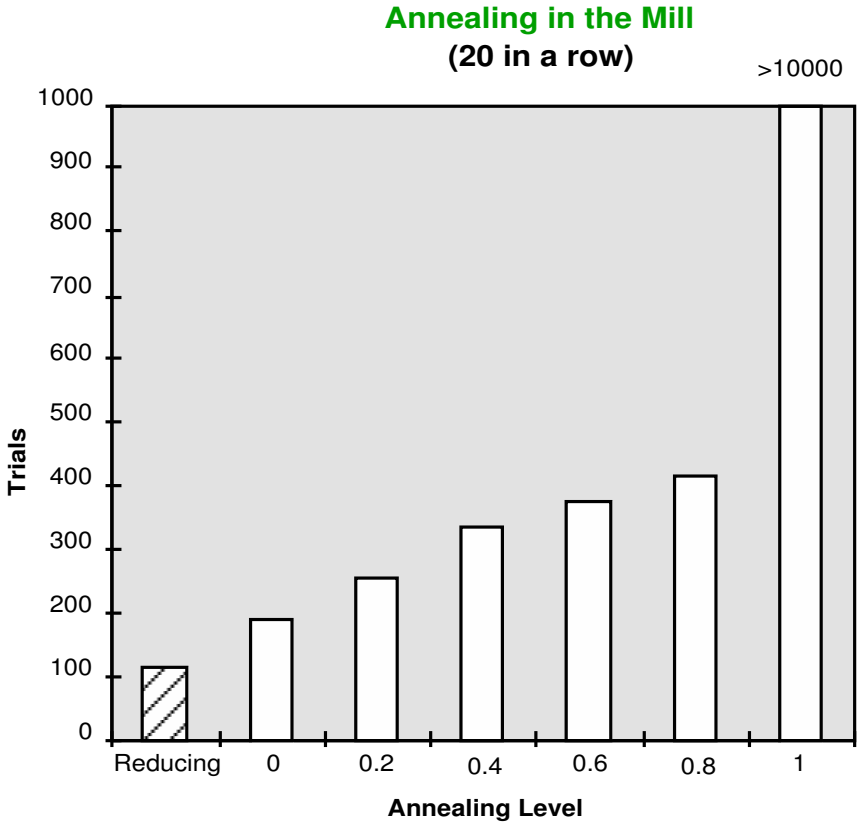

FIG. 1.3. The effect of annealing on learning to control the skinpass mill. The criterion for success is to balance the pole for 10,000 time steps and repeat that 20 times in a row. The number of trials plotted is the time taken to succeed for the first time.

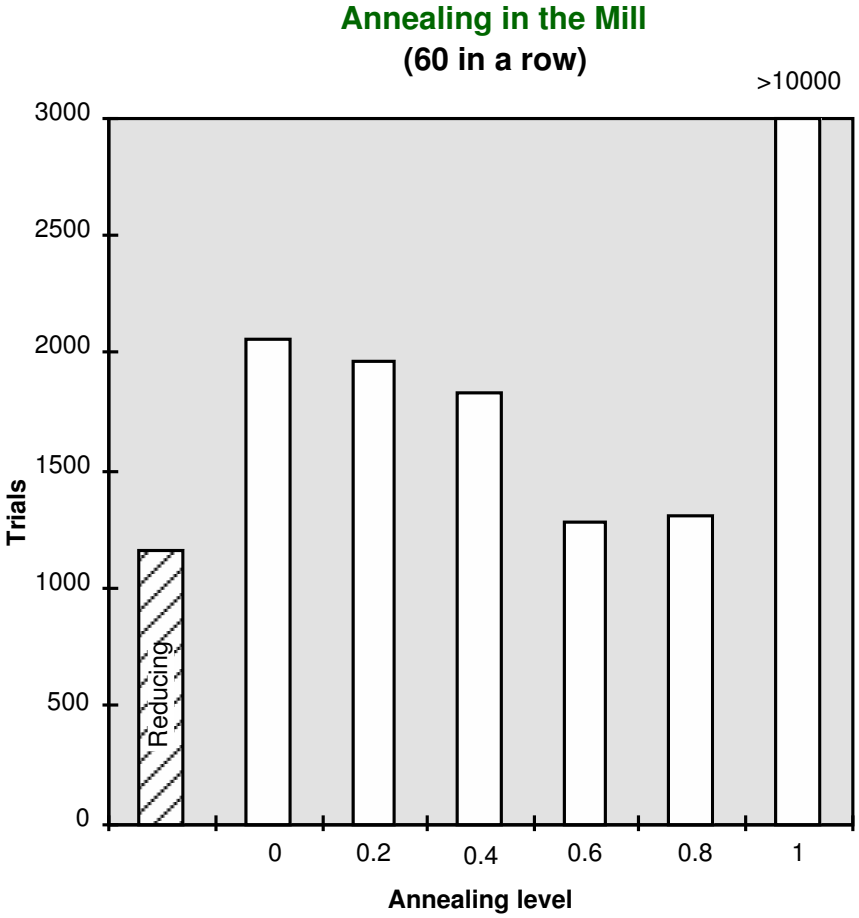

FIG. 1.4. The effect of annealing on learning to control the skinpass mill. The criterion for success is to balance the pole for 10,000 time steps and repeat that 60 times in a row. The number of trials plotted is the time taken to succeed for the first time.

The effect of annealing on learning to control the skinpass mill. The criterion for success is to balance the pole for 10,000 time steps and repeat that 60 times in a row. The number of trials plotted is the time taken to succeed for the first time.

Constant annealing chooses the action according to the following rule:
$Probability(action = i) \propto AvLifetime_i$
if $Av\ Lifetime_i > \beta \times Max\ Av\ Lifetime$
$Probability(action = i) = 0$ otherwise

That is, the probability of choosing action i, is proportional to the average lifetime for that action in a particular box. In addition, a cut-off level is defined such that if the average lifetime is less than β times the highest average lifetime of an action in the same box, then that action will never be chosen.

In the constant annealing scheme, the value of β is constant throughout a complete learning sequence. Under reducing annealing, the value of β changes according to the formula:

$$\beta = \frac{Global\ Lifetime}{Target\ Lifetime}$$

where the global lifetime is the current lifetime of the system, as a whole, and the target lifetime is the success criterion of 10,000 time steps (in the present experiments). With this method the cut-off level is raised as performance increases.

Interestingly, the versions of the BOXES algorithm described by Sammut (1994) consistently failed the robustness tests used here. While that algorithm learns to control the pole and cart system quickly, it cannot achieve a consistent level of performance by retaining the box statistics from one learning sequence to the next as annealing does. We have previously proposed a method of voting (Sammut and Cribb 1990) to construct a robust controller for the pole and cart. Unfortunately, this method does not scale to problems that have a large number of control actions. The combination of annealing and not resetting the statistics kept in each box after a successful sequence appears to be more promising.

5 Training for Noise

Previous research in machine learning (Quinlan 1986) suggests that it is necessary to train a learning system in a noisy environment if the final system is to be used in a noisy environment. As can be seen in Figures 1.5 and 6, the performance of the algorithm when trained in this way is in accord with our expectations.

The average time to failure of the mill with a noise ratio of 0.1 is about 1000 seconds when using actions learned with no noise. Actions learned with the same noise ratio of 0.1 achieve an average time to failure of about

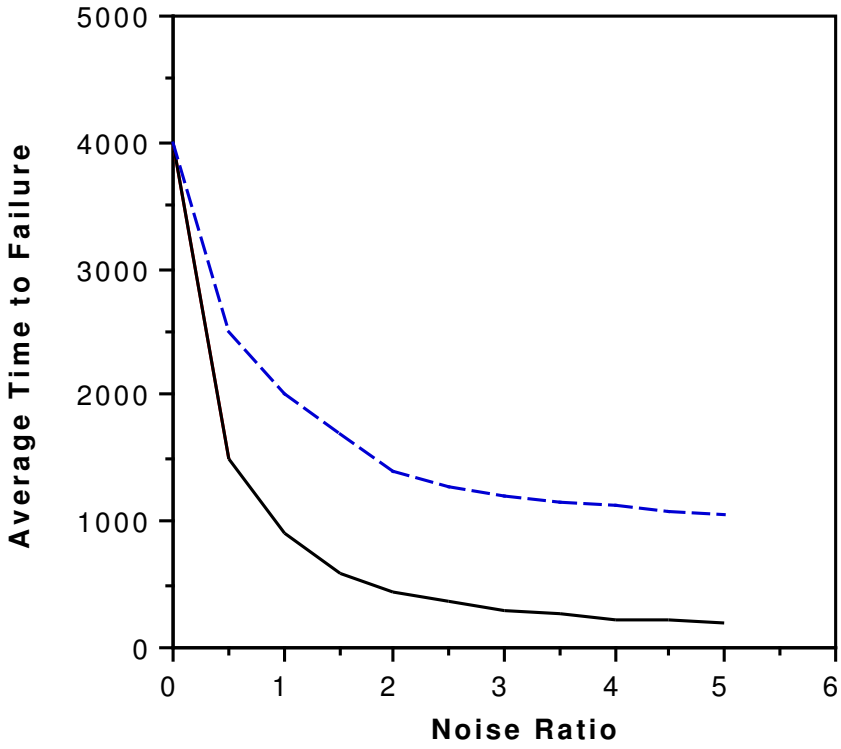

FIG. 1.5. Performance on a zero noise plant after training on the zero noise plant. The dotted line shows the performance on a noisy plant.

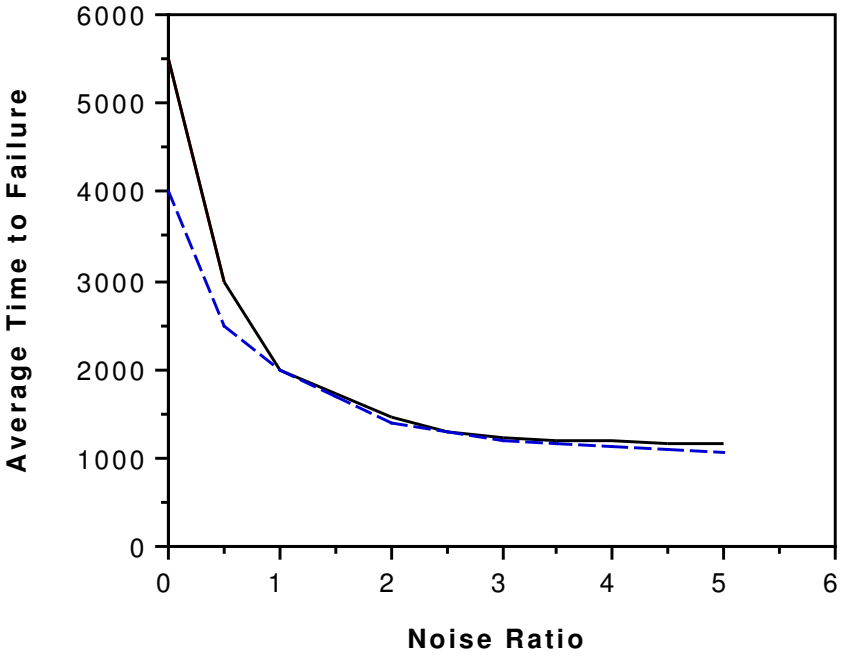

FIG. 1.6. Performance on a zero noise plant after training with a noise level of 0.1. The dotted line shows the performance on a noisy plant after training with the same noise level.

2000 seconds. It seems from this result that learning on a low noise plant does not improve performance on higher noise plants. However, further tests were conducted, this time with the initial training being done on a low noise plant with a noise ratio of 0.1. The controller now performs better at zero noise levels than the controller learned on zero noise levels. Thus, far from being an impediment to learning, introducing a small amount of noise actually helps the controller to learn more about the plant. These graphs show that training on a zero noise plant produced a poor controller for a noisy plant, while conversely, a controller trained on a noisy plant produces a robust controller useful for all noise levels that is actually better for the zero noise plant than the zero noise controller. This supports the earlier suggestion that noise, or excitation of all modes of the plant, is important for good modelling of the plant.

6 Actuator Output

The BOXES algorithm requires that the actuator output be quantised. The problem with this is that coarse quantisation leads to unnecessarily large actuator changes. This would be highly detrimental to a commercial plant, coming with the attendant maintenance problems. Three methods were investigated in an attempt to alleviate this problem.

6.1 Smoothing the Output

An attempt was made to filter the output to the actuator in the time domain. Such a filter is usually a running average of previous actuator outputs. This type of filtering introduces a delay and so to minimise the effects of the delay, the filter is generally first-order. That is, the new output is only a function of the immediately preceding output and the input.

$$u_t = u_{t-1} + \alpha(u'_t - u_{t-1})$$

where u' is produced by BOXES and $u_i$is output to the plant.

By varying the filter coefficient, α, a smoother response can be obtained. However, substantial drop off in performance occurs when the filtering is present, as shown in Figure 1.7.

In order to explain why the performance is reduced, we looked at the response of BOXES in the time domain. The large damped oscillations in Figure 1.8 provides one explanation. As can be seen, filtering the output in this way, while producing a smoother controller, also results in delays in the control loop, and large, slow oscillations in the controlled variable.

6.2 Weighting the Output with Error Magnitude

This type of smoothing relies on weighting the output with a function of error (difference from a setpoint). In this way, the controller should respond to large errors with large control actions, bringing the plant under control.

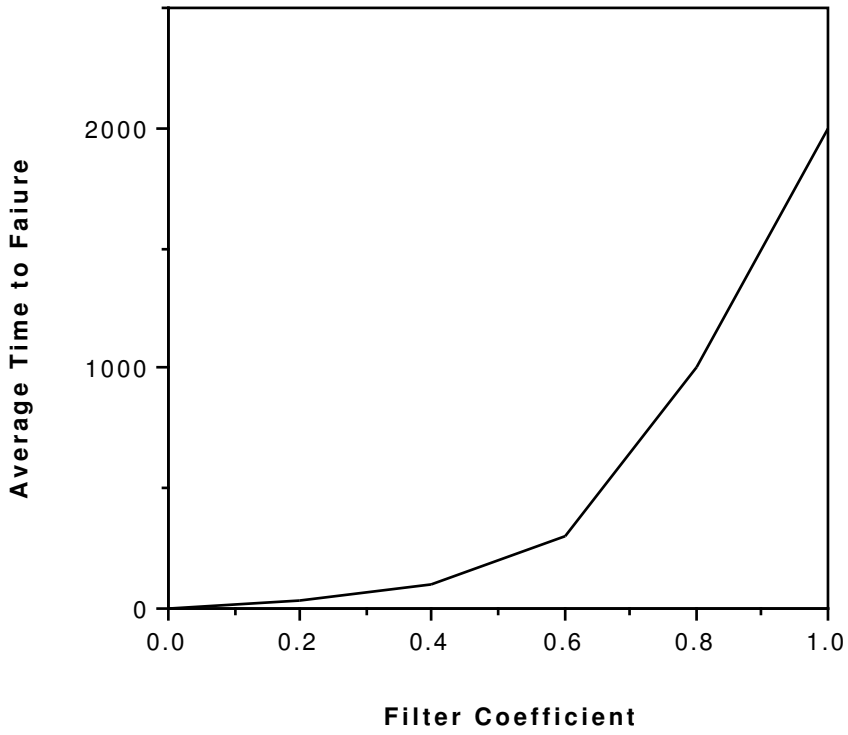

FIG. 1.7. Performance as actuator output is filtered

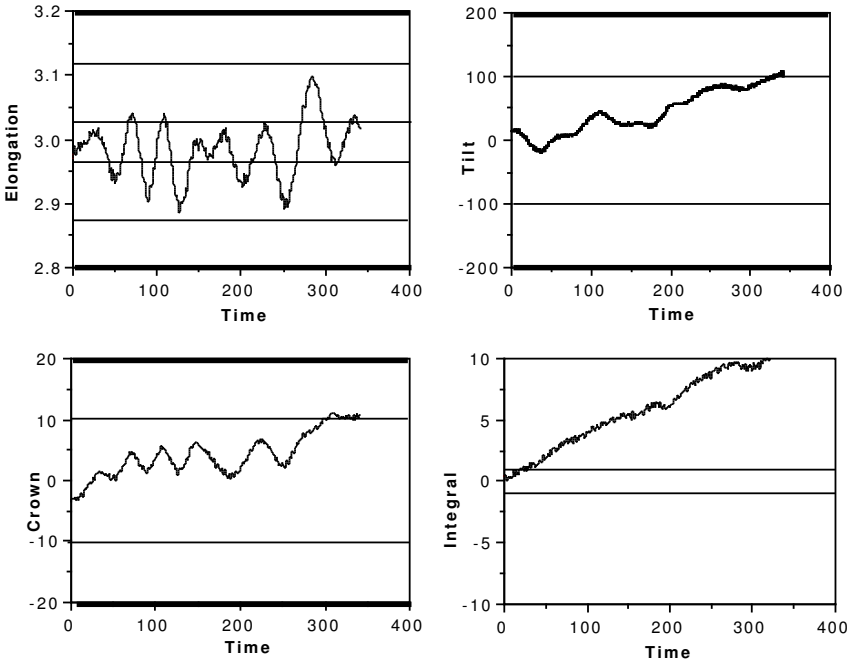

FIG. 1.8. Response of one run with high filtering. Note the long damped oscillations. X-axis in seconds.

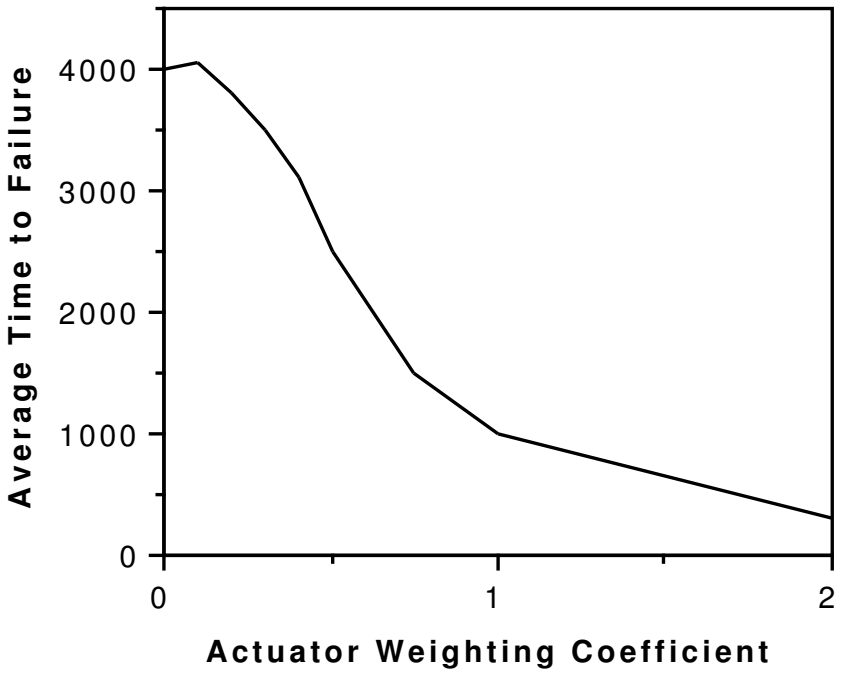

FIG. 1.9. Actuator Weighting vs Performance. No significant best value is observed.

Likewise, as the error becomes smaller, the actuator changes also become smaller, and a smoother controller results. To test this theory, the following function of actuator weighting was used.

$$Action'(e) = \left| \frac{2e}{e_{max}} \right|^{\alpha} \times Action(e)$$

where $e_{max}$is the error at the failure boundary.

The output of the actuator is thus weighted by a function of the absolute value of the error. To test the effectiveness of weighting the output in this way, and perhaps to find a good weight curve, 20 test runs of selected algorithms with different weighting parameters were allowed to learn from 30,000 trials. The average final value of the time to failure was recorded for selected weighting parameters. Note that the results from this example, like many in these experiments, may be specific to the skinpass mill. The results are not meant to be useful for all plants, but simply to show the viability of the idea. The results from this test are given in Figure 1.9.

These are disappointing, in that no significant best weighting curve was found. However, the sharp decline in performance around the actuator weighting coefficient of 0.5 warranted further attention. Again, we turn

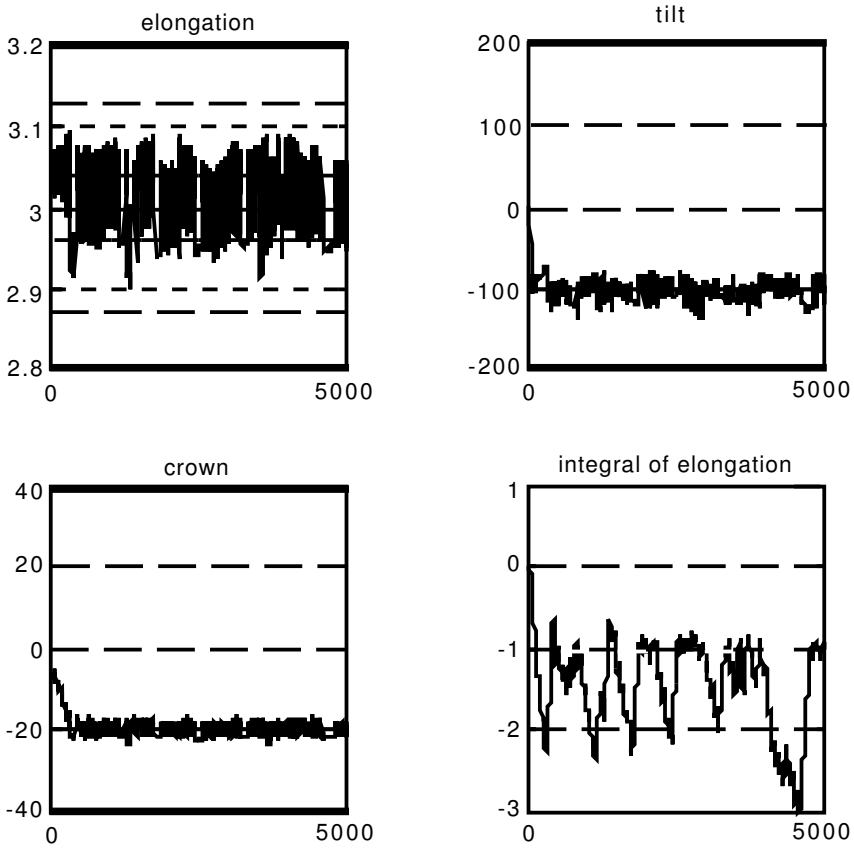

FIG. 1.10. Large oscillations due to wide zero.

to the time domain behaviour of BOXES controlling the mill, and take a typical example from weighting parameter of 0.3, and one at 0.5. Both examples are at a noise ratio of 0.5, which is quite high. These are shown on Figures 1.10 and 1.11 respectively.

We speculate that the drop-off is due to larger oscillations occurring around the setpoint These oscillations are in turn due to the lack of control and the large noise amplitude. If this is the case, then smaller weighting parameters produce narrower regions where the controller has little effect, which leads to smaller oscillations. Conversely, a large weighting parameter would produce larger regions where the controller has little effect. If the oscillations became larger than the size of the boxes, this would possibly produce instability and poor performance.

Figure 1.2 shows a successful controller with larger oscillations around the setpoint. The amplitude of these oscillations is about $\pm 0.08\%$, or about

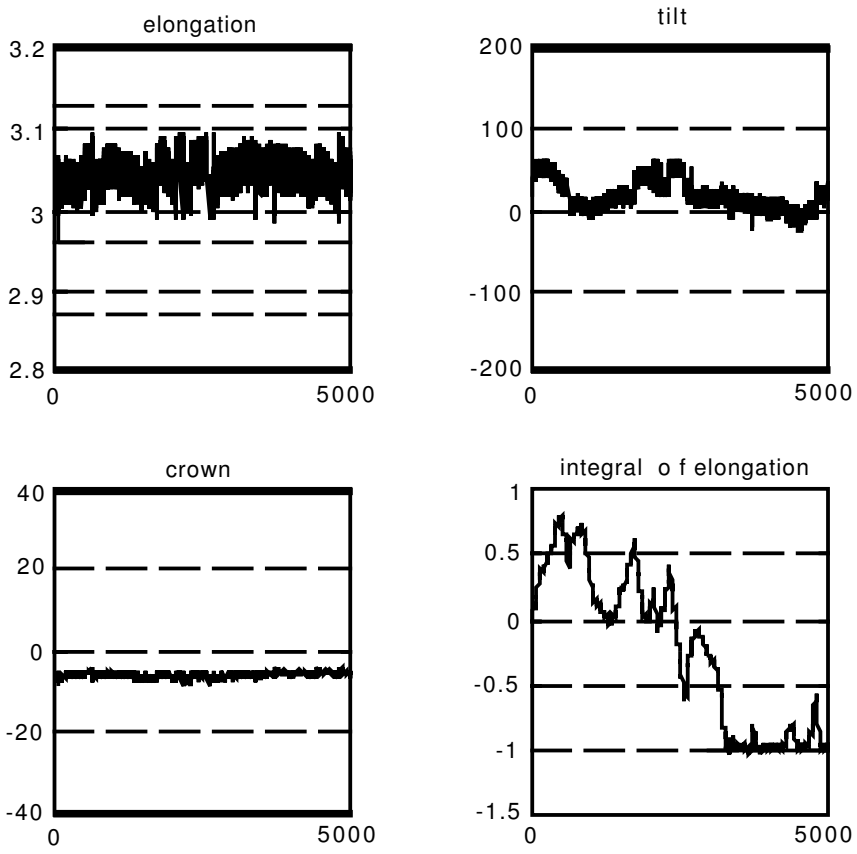

FIG. 1.11. Smaller oscillations due to narrower zero. Both runs (Figures 1.10 and 1.11) had the same external noise level.

40% of the failure boundaries. This algorithm has a weighting parameter of algorithm of 0.5, so the weighting function should have a value of about 0.65 at he limit of oscillations. In Figure 1.11, the size of the oscillation is smaller, about 25% of the failure boundary, and the weighting parameter is 0.3. This gives a similar weighting function value, about 0.65. While this hardly convincing proof that a direct relationship between the value of the weighting function and the size of the oscillations exists, it does support to some extent the idea that reducing the actuator output around zero may reduce performance. Also note that the actual values where the performance drops off are related to the choice of gains available to each box. These were not chosen for any good reason in the original formulation of the problem, and have not been included in the learning procedure in any way. Thus, in order to produce a gentler controller, these gains could be chosen (hopefully as part of the learning scheme, but perhaps by a human designer) to produce a smoother but still useful controller.

6.3 Control Effort Cost

When designing more conventional controllers, a common way of compromising between controller fluctuations and other measures of control quality such as setpoint following is to place a cost on the chosen action. This cost may be related to the magnitude of the action or to the size of the change of the action depending on the desired behaviour of the controller. We have used this idea in BOXES to smooth the output of the controller for the pole and cart with better results than either of the first two methods. The way we have done this is to modify the merit equation, as shown below.

$$merit = weight \times \frac{Lifetime}{Usage^k}$$

A 'do nothing' action was introduced into the pole and cart system. This action was given a large weighting ($w = 3$) in comparison to the push-left and push-right actions ($w = 1$). Figure 1.12 shows how the original, unweighted BOXES controller performs on this task, with the solution being characterised by jerky, unnecessary actions.

Figure 1.13 shows how BOXES performs with weighting. This second graph shows a marked difference in control strategy (which is also evidenced in the rules developed by the learning agent). In comparison with the earlier methods of actuator smoothing, there is only a minimal increase in learning time to reach the same level of average time to failure for the pole and cart.

Because the mill has 125 actions, providing a weighting is somewhat more complicated and experiments are still proceeding.

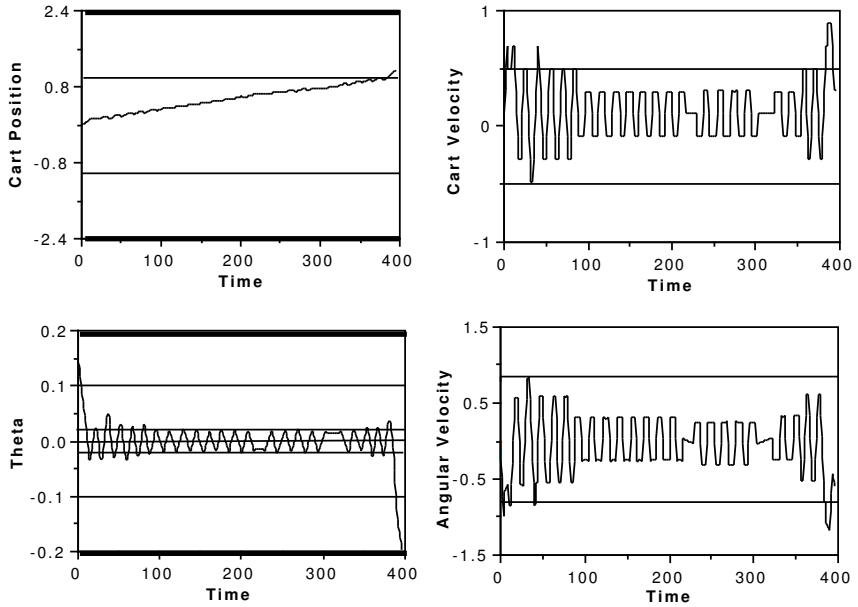

FIG. 1.12. The behaviour of the pole and cart using a controller without a cost on the action

7 Conclusions

Reducing annealing allows most of the actions in a box to gain experience. This means that a more complete model of expected time to failure can be built up for each action. As a result a more robust controller for the mill could be constructed.

It was noticed that adding noise to systems with no annealing or no noise improved the performance of the mill. It was suggested that this is because the noise excites the plant, enabling the BOXES model of the plant to be made more complete. This effect was not evident for the pole and cart, probably because the instability of the plant caused enough excitation by itself to make it possible to model the plant.

Three methods were attempted to improve the quality of the BOXES control. Filtering the output had the effect of introducing a delay into the control loop. As might be expected, this produced a marginally stable controller, exhibiting long, slow oscillations. Very poor performance was found. It was hoped that by weighting the output to be smaller near to zero error, a smoother controller might result. Instead, the small weighting near zero error produced a zone where the controller had little effect, and the oscillation usually found in a BOXES controlled plant increased in amplitude to fill this zone. Only very steep weighting showed any sign of improving performance. Placing a cost on a control action was the

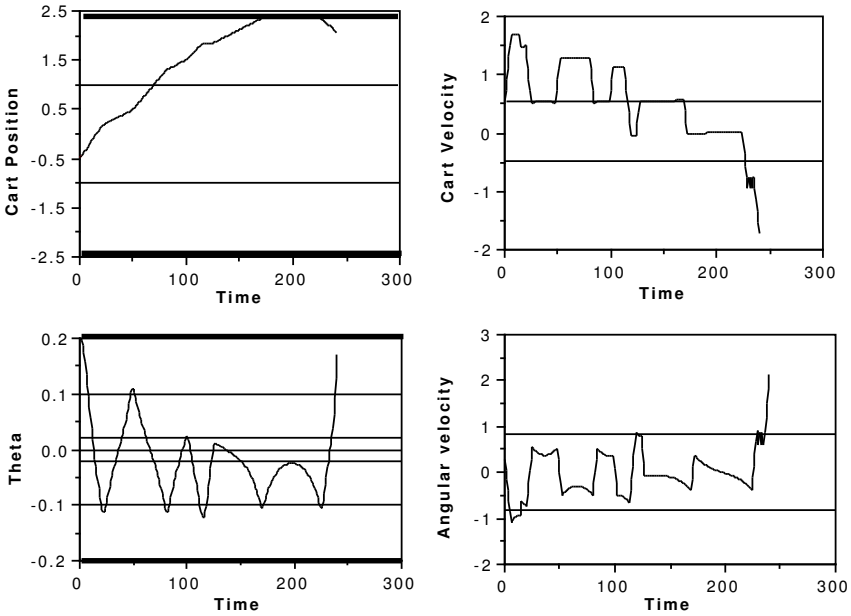

FIG. 1.13. The behaviour of the pole and cart using a controller with a cost on the action

most useful of the three. Using a BOXES learning agent that places a cost on action magnitude, we found learning times were not significantly effected. The resulting controller, however, was far more economical with its outputs, resulting in a controller which produced a output only when really necessary. Unfortunately, no systematic way of choosing the cost for each action has yet been found, but our results do show that this technique is worth pursuing.

Bibliography

1. Anderson, C. W. (1987). Strategy Learning with Multilayer Connectionist Representations. In P. Langley (Eds.), *Proceedings of the Fourth International Workshop on Machine Learning.* (pp. 103–114). Los Altos: Morgan Kaufmann.
2. Michie, D. and Chambers, R. A. (1968). Boxes: An Experiment in Adaptive Control. In E. Dale and D. Michie (Eds.), *Machine Intelligence 2.* Edinburgh: Oliver and Boyd.
3. Quinlan, J. R. (1986). The Effect of Noise on Concept Learning. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach, Vol. 2.* Los Altos: Morgan Kaufmann Publishers.

4. Sammut, C. A. (1994). Recent Progress with BOXES. In K. Furakawa, S. Muggleton and D. Michie (Eds.), *Machine Intelligence 13*. Oxford: The Clarendon Press, OUP.
5. Sammut, C. and Cribb, J. (1990). Is Learning Rate a Good Performance Criterion of Learning? In B. W. Porter and R. J. Mooney (Eds), *Proceedings of the Seventh International Machine Learning Conference*. (pp. 170–178). San Mateo, CA: Morgan Kaufmann.
6. Roberts, W.L. (1972). An approximate Theory of Temper Rolling, *Iron and Steel Engineer YearBook*, pp 530–542.