# Chapter 6

# Testing the FOX controller

## 6.1 Introduction

This chapter describes how to design FOX-based control systems. Various aspects of the design process are investigated, such as how to measure a system impulse response, how to construct the FOX eligibility profile model using this information, various eligibility profile modeling problems (and their solutions), and ways to cope with nonlinear systems.

Experiments were performed on one simulated system and three physical systems: a simulated gantry crane, a real gantry crane, an inverted pendulum, and a track-following wheeled robot. As each one of these experiments is described below, some new elements are introduced into the design process. In each hardware experiment the control algorithm was performed inside a real-time control process running under the Linux operating system on a Pentium class PC. The implementation details of this software system, which is a sophisticated and general purpose real-time controller, are given in Appendix K.

## 6.2 Design technique

This section describes the steps required to control a system using FOX.

### 6.2.1 Step 1: Feedback control

Many uncompensated systems will have unacceptable impulse responses, i.e. responses that can not be easily modeled as eligibility profiles. Some typical problems with an uncompensated impulse response are:

- It may not decay to zero (i.e. the system never reaches its reference value).
- It may take a long time to decay to zero (i.e. the system takes a long time to reach its reference).
- It may be too complicated, in the sense that a high order linear system model would be required for sufficient accuracy.
- It may be too nonlinear, in the sense that a linear model would not encompass some essential system dynamics.

| Error function name | Equation | Learning rates |
|---|---|---|
| Standard | $E = \sum_{i=1}^{T} \left( C\,\mathbf{y}_i - y_i^d \right)^2$ | $\alpha$ |
| Overshoot | $E = \sum_{i=1}^{T} \left( e_i' \right)^2$ ($e_i'$ is defined in section 6.2.2) | $\alpha_1$, $\alpha_2$ |
| Output limiting | $E = \sum_{i=1}^{T} \left( C\,\mathbf{y}_i - y_i^d \right)^2 + \frac{\beta}{2} \sum_{i=1}^{T} \mathbf{x}_i^T\,\mathbf{x}_i$ | $\alpha$, $\beta$ |
| Output derivative limiting | (see section 5.8.2) | $\alpha$, $\gamma$ |

**Table 6.1**: *Some common error functions and their learning rate parameters. Note that, except for the overshoot case, $\alpha$ is the main learning rate.*

- It may be too oscillatory.

An "internal" feedback controller (IFC) is added to the system to correct these problems (it is equivalent to the FBE feedback path). For mechanical systems it is usually some kind of position-reference linear controller. The IFC is included in the system block $F$ along with the original uncompensated system. The IFC embodies some of the "expert" knowledge of how the system should be controlled.

Selecting the IFC usually involves a tradeoff. A simple IFC may result in a complicated impulse response that will be hard to model. The impulse response can be made simpler at the expense of creating a more complicated IFC. But this may limit the ability of FOX to adapt to changing system parameters, or prevent it from accurately compensating for system nonlinearities. A balanced approach is required: select a "minimal" IFC that just covers the above requirements, so FOX can still discover the system dynamics for itself from the resulting eligibility profiles.

It will be shown in section 6.6 that an impulse response that does not decay to zero can be modeled by an eligibility profile that does. This is not too surprising when it is considered that an eligibility value measures the influence that a given weight has had on the system. In general, as a system operates *any* eligibility should decay to zero because a weight's influence on the system will become less important as other weights gain control (i.e. as more recent control outputs direct the system along new trajectories).

### 6.2.2 Step 2: Select error function

The error function determines how the trained system will behave, so it must be selected with care. Usually several training experiments will be necessary before a satisfactory error function is obtained. The error functions that have been described thus far, and their learning rate parameters, are shown in table 6.1.

Table 6.1 introduces a heuristic technique called "overshoot error", which is sometimes useful. This is similar to the standard error function, except that the error signal $e_i$ is redefined to be:

$$e_i' = \begin{cases} \alpha_1 \left( C\,\mathbf{y}_i - y_i^d \right) & \text{if } \left| C\,\mathbf{y}_i - y_i^d \right| - \left| C\,\mathbf{y}_{i-1} - y_{i-1}^d \right| < 0 \\ \alpha_2 \left( C\,\mathbf{y}_i - y_i^d \right) & \text{otherwise} \end{cases} \tag{6.1}$$

where typically $\alpha_2$ is 5–100 times larger than $\alpha_1$. If the reference error reduces over time then this is just the standard error function with learning rate $\alpha_1$. If the error increases (for example if the system

overshoots its reference) then the learning rate is increased to $\alpha_2$. This penalizes reference overshoots, requiring FOX to avoid them by applying a pre-emptive "braking force".

Combinations of the error functions in table 6.1 are possible, such as overshoot training plus output limiting. In fact, some amount of output limiting (or output derivative limiting) will usually be required in a practical system. The output limiting learning rate must be selected high enough to prevent over-training, but low enough so that FOX is still capable of controlling the system.

### 6.2.3 Step 3: Measure eligibility profiles

Measurements must be made on the system to determine the eligibility profiles. This is done as follows:

1. Set the system reference to zero.
2. Apply an impulse to each component of $\mathbf{x}$ in turn.
3. For each component, the measured error is the ideal eligibility profile.

Impulses are hard to apply in practice. A more practical method for linear systems is to apply a step function, then the impulse response will be the derivative of the measured error.

### 6.2.4 Step 4: Synthesize $A$,$B$,$C$

The impulse responses are used to construct the eligibility profiles (the linear system model $F^*$ given by the matrices $A$,$B$ and $C$). This model does not need to be perfect, as long as the modeled eligibility profiles are sufficiently close to the measured impulse responses. There are three main modeling techniques:

1. Theoretical: A known system model is used to obtain the form of $A$,$B$ and $C$. Some parameters are adjusted to make the model fit the measured impulse response.
2. Algorithmic: The matrices $A$,$B$ and $C$ are synthesized directly from the measured impulse response. An algorithm to do this was not created, see section 8.1.
3. Heuristic: A generic first, second or third order model is tweaked until it matches the measured impulse response. This was the method used most often in practice.

It is sometimes possible to reduce the *order* of the model with little loss in accuracy. This occurs if at least one eigenvalue in the system matrix $A$ is particularly small, so its contribution to the output decays quickly. The eigenvalue/eigenvector can be deleted from the eigen-expansion of $A$, and $B$ and $C$ are correspondingly modified. A procedure for doing this is given in Appendix G. An additional benefit of reducing the model order is that the FOX trace precision is better if the remaining eigenvalues are closer together.

### 6.2.5 Step 5: Select $\sigma$

The WIB buffer size $\sigma$ must be large enough that the eligibility profile is not truncated too much, and small enough that numerical imprecision does not manifest itself in the trace calculations. A full analysis of the problem and a method for selecting $\sigma$ is given in Appendix F. A guideline is too choose $\sigma$ to be the number of time steps it takes for the eligibility profile to decay to 1% of its maximum value, as long as this does not conflict with trace precision requirements for the floating point data type being used.

### 6.2.6 Step 6: Select the CMAC inputs

This must be done so that there is just enough sensor information to uniquely identify each point on all useful system trajectories. Less sensors will result in training correlation between two or more parts of the trajectory. More sensors can increase the CMAC noise factor, although this depends on the nature of the CMAC input. It can be useful to use CMAC sensors which are pre-processed versions of the systems sensors, to reduce the required number of sensors without compromising the uniqueness requirement.

The time $t$ can be used as a CMAC input in a feed-forward system. This has the advantage that the CMAC input is guaranteed to uniquely identify all input space points on the desired trajectory. However the system must be guaranteed to start from the same state at each training run (time $t = 0$) and have the same desired trajectory each time. Because there is no input space overlap, learning will have to occur separately at all times on the desired trajectory (even if that trajectory is periodic, with multiple identical parts).

### 6.2.7 Step 7: Testing

The system is operated and the FOX controller is trained. Training will only occur in those regions of the workspace that are actually seen. Thus the system reference should repeatedly traverse all the potentially useful paths during training.

The various learning rates must be adjusted to achieve stability. A higher learning rate results in faster learning, but can cause instabilities and training interference. The proportions of the main reference learning rate and the output limiting (or overshoot) learning rates need to be adjusted to get the desired "optimal" performance.

## 6.3 System approximation requirements

This section will give some insight into the question: how close must the approximation $F^*$ be to the real system $F$ for FOX training to succeed? If the weight changes for a training iteration are selected to follow the gradient $\mathrm{d}E/\mathrm{d}w_{ij}$ then the control signals $\mathbf{x}_i$ will follow the gradient $\mathrm{d}E/\mathrm{d}\mathbf{x}_i$. The backwards propagation rule for finding the $\mathrm{d}E/\mathrm{d}\mathbf{x}_i$ values can be easily derived (refer to Appendix B):

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,\mathbf{y}_i} = \frac{\mathrm{d}\,E}{\mathrm{d}\,\mathbf{y}_{i+1}} \cdot A + 2\,e_i\,C \tag{6.2}$$

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,\mathbf{x}_i} = \frac{\mathrm{d}\,E}{\mathrm{d}\,\mathbf{y}_{i+1}} \cdot B \tag{6.3}$$

These two equations define an ordinary linear filter, with $e_i$ as the input and $\mathrm{d}E/\mathrm{d}\mathbf{x}_i$ as the output, that operates backwards in time. This filter is not the quite the same as the system $F^*$, although both have the same system matrix $A$. Assume for the moment that $F$ is linear. Let $\nabla_i = \mathrm{d}E/\mathrm{d}\mathbf{x}_i$ where the $A$,$B$ and $C$ matrices of the original system $F$ are used. Let $\nabla_i^*$ be the output generated when the approximation $F^*$ is used. Training convergence requires that

$$\sum_{i=1}^{T} \nabla_i \cdot \nabla_i^* > 0 \tag{6.4}$$

(see Appendix A.3 for the reason why). In hand-waving terms this means that $\nabla_i$ and $\nabla_i^*$ must be "mostly of the same sign". This is demonstrated in figure 6.1. From this it seems that a large amount of

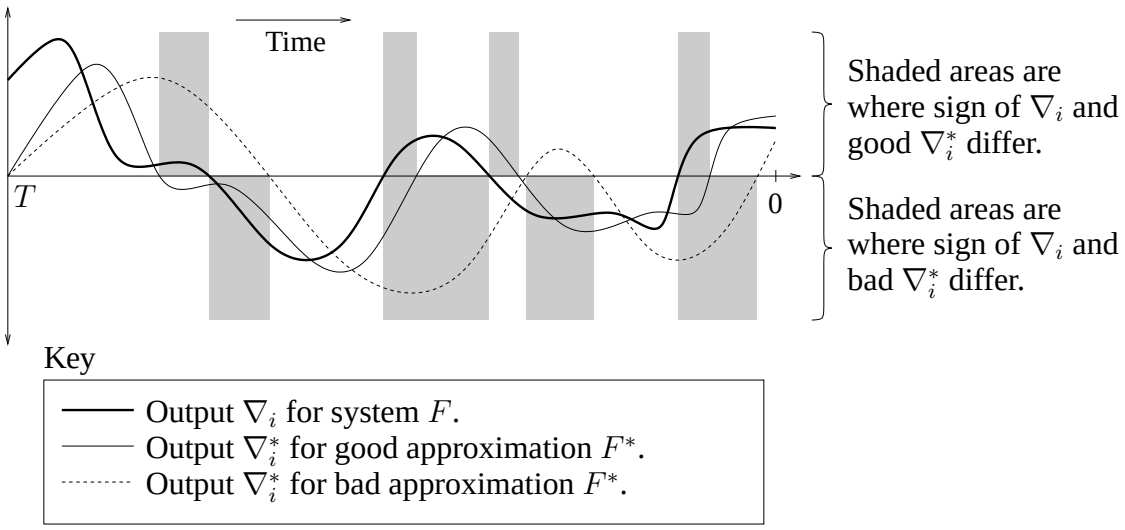



**Figure 6.1**: *Good and bad $\nabla_i^*$ examples. If $F^*$ is a good model of $F$ then $\nabla_i^*$ will track $\nabla_i$ more closely. The good $\nabla_i^*$ has less difference in sign from $\nabla_i$.*

deviation between $F$ and $F^*$ would be allowable—this is confirmed by experiment later in this chapter (but obviously the closer $F^*$ is to $F$ the better the chance of successful training).

This fact allows a degree of confidence that FOX can control nonlinear systems. Consider the case where $F^*$ is constructed to match a local linear approximation of a nonlinear system. As long as the linear approximations at all useful points of the state space are within the bounds of the allowable $F^*$ deviation then training should succeed.

Notice that the exact form of the input $e_i$ is a factor in determining if equation 6.4 is true. This fact makes it difficult to find a more formal condition for training convergence.

## 6.4 Testing: Simulation

The design and operation of a FOX based controller will now be illustrated using a simulated gantry crane control system. A variety of design and training strategies will be presented.

Figure 6.2a shows a simple linear fourth order model of a gantry crane (mass $m_1$) with a load (mass $m_2$) suspended on a cable (represented by a spring with spring constant 1). The FOX controller structure is shown in figure 6.2b. The crane and load are at positions $p_1$ and $p_2$. A force $f$ is applied to the crane to get the load position equal to a reference value. The equations of motion are:

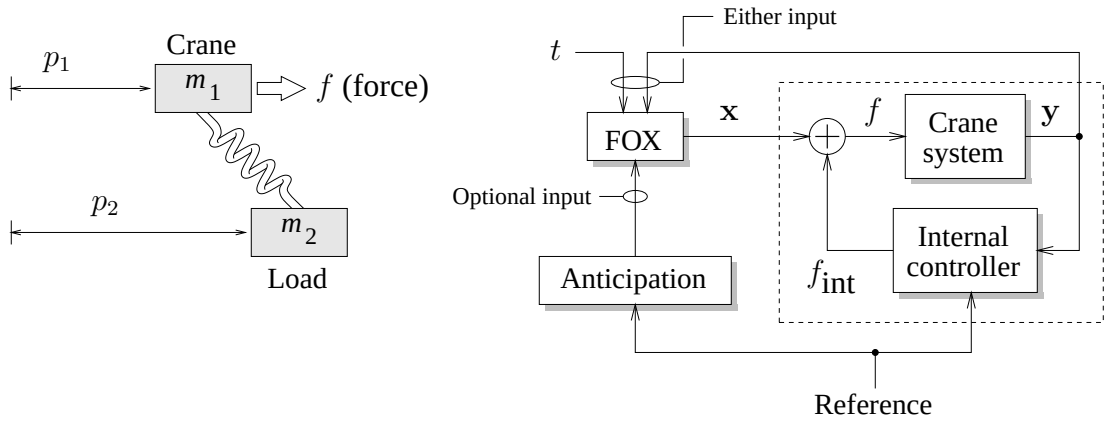
$$\ddot{p_1} = \frac{(p_2 - p_1 + f)}{m_1} \tag{6.5}$$

$$\ddot{p_2} = \frac{(p_1 - p_2)}{m_2} \tag{6.6}$$

$$f = \mathbf{x} + f_{\text{int}} \tag{6.7}$$

$$f_{\text{int}} = k_1\, p_1 + k_2\, p_2 + k_3\, \dot{p_1} + k_4\, \dot{p_2} \tag{6.8}$$

Note that the system includes an "internal" PD controller which adds a force $f_{\text{int}}$ to $\mathbf{x}$. The system state




(a) (b)

**Figure 6.2**: *(a) Simple linear gantry crane model. (b) FOX controller structure for controlling this system.*

vector is

$$\mathbf{y} = \begin{bmatrix} p_1 \\ p_2 \\ \dot{p_1} \\ \dot{p_2} \end{bmatrix} \tag{6.9}$$

The following parameters were used in all the experiments:

$$m_1 = 5 \qquad m_2 = 1 \tag{6.10}$$

$$k_1 = -3 \qquad k_2 = 2 \qquad k_3 = -15 \qquad k_4 = 10 \tag{6.11}$$

The resulting system matrices are (using an Euler approximation with time step $h = 0.1$):

$$A = \begin{bmatrix} 1 & 0 & 0.1 & 0 \\ 0 & 1 & 0 & 0.1 \\ -0.08 & 0.06 & 0.7 & 0.2 \\ 0.1 & -0.1 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0.02 \\ 0 \end{bmatrix} \tag{6.12}$$

Note that $A$ and $B$ represent both the gantry crane system and the internal controller.

### 6.4.1 Standard error function with the fourth order model

The second step of the FOX design process is to define an error function:

$$\text{error} = e = p_2 - \text{reference} \tag{6.13}$$

$$= C\,\mathbf{y} - \text{reference} \tag{6.14}$$

$$\text{where} \quad C = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \tag{6.15}$$

Then the system impulse response (desired eligibility profile) must be determined. This is particularly easy because the $A$ and $B$ matrices are already known. The impulse response, shown in figure 6.3, is

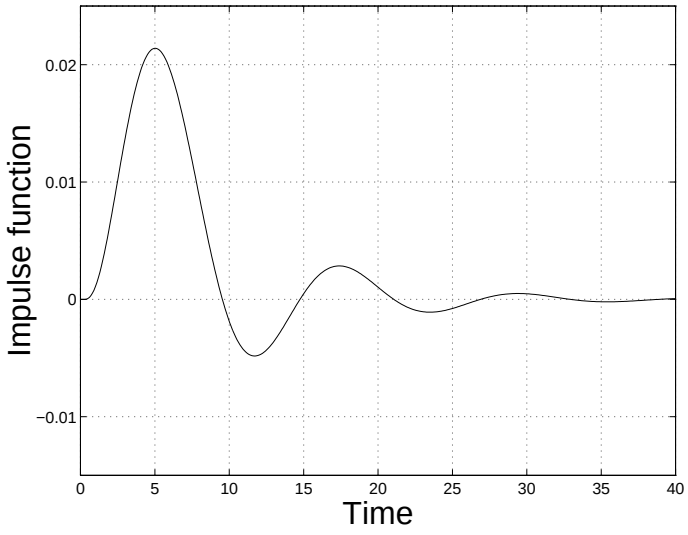



**Figure 6.3**: *Impulse response of the gantry crane system (including its internal controller).*

| | |
|---|---|
| System time step ($h$) | 0.1 |
| Total time steps per iteration ($T$) | 400 |
| Total number of iterations | 20000 |
| CMAC input resolution | 400 |
| CMAC input minimum | 0 |
| CMAC input maximum | 40 |
| CMAC number of weights | 100000 |
| CMAC $n_a$ | 20 |
| Main learning rate ($\alpha$) | 0.05 |

**Table 6.2**: *The CMAC parameters that were used for most of the simulated gantry crane experiments.*

$f(i) = CA^iB$. In this case the system matrices $A$ and $B$ will be used unchanged in the FOX controller, so no approximation step is necessary (in other words, the full fourth order eligibility model is used). The history size $\sigma$ is set to 350, because the impulse response has almost fully decayed after 35 seconds. The consequences of this choice will be explored below. Finally, the CMAC input is selected to be the time $t$ ($t = ih$ where $i$ is the time step number). This is only possible because the simulation starts at the same state and has the same reference (which is a function of $t$) in each iteration. The other CMAC and simulation parameters for the gantry crane experiments are shown in table 6.2.

Figure 6.4 shows how the error improves steadily with the number of training iterations. The total error in this case was computed by:

$$\text{total error} = E = \sqrt{\sum_{i=1}^{T} h\, e_i^2} \tag{6.16}$$

Figure 6.5 shows the reference signal and the trajectories that result after training is completed. The load

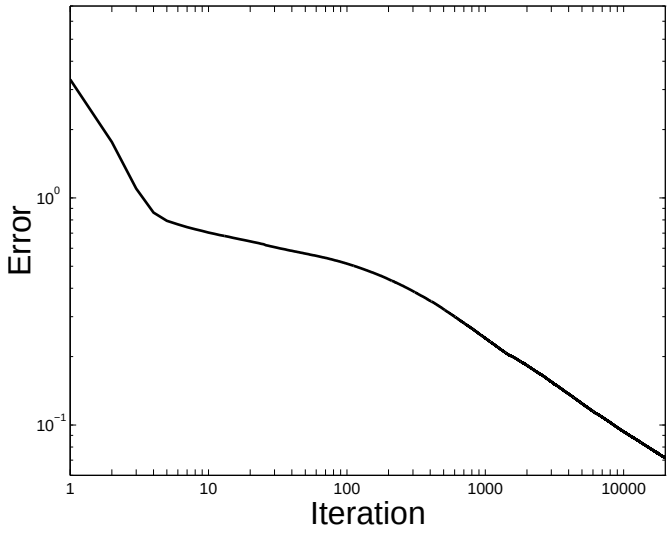



**Figure 6.4**: *Gantry crane controller simulation: Reference error improvement with number of iterations. The standard FOX error function with a fourth order eligibility model is used.*

position $p_2$ is almost exactly equal to the reference. There is nothing remarkable about this result—the FOX eligibility profile is exact and the system is linear and so convergence is ensured.

The fourth order eligibility model in this simulation required 320 bits of floating point precision[1] to achieve sufficient accuracy. By comparison, the IEEE single and double precision floating point numbers have 24 and 48 bits of precision respectively. A C++ high precision floating point number class was used instead of the usual C `float` or `double` data types. As a result the 20000 iterations took two days to compute on a Pentium II 300MHz machine. Needless to say, such extraordinary numerical precision is impractical.

The eigenvalues of $A$ can be examined to see why such accuracy is required:

$$\text{Eigenvalues of } A = \begin{bmatrix} 0.9847 + 0.052i \\ 0.9847 - 0.052i \\ 0.9718 \\ 0.7589 \end{bmatrix} \tag{6.17}$$

The smallest eigenvalue is 0.7589. Thus from equation 5.35 at least 42 decimal digits of precision are required. The 160 fraction bits give about 48 decimal digits precision, just a bit more than is required. If a standard IEEE floating point type has to be used then the only way to make it work is to use a small history size - but small history values will chop off most of the eligibility profile.

[1] Fixed point numbers were used with 160 bits in the integer part and 160 bits in the fractional part. The integer part needs the same number of bits so that all reciprocals are representable.

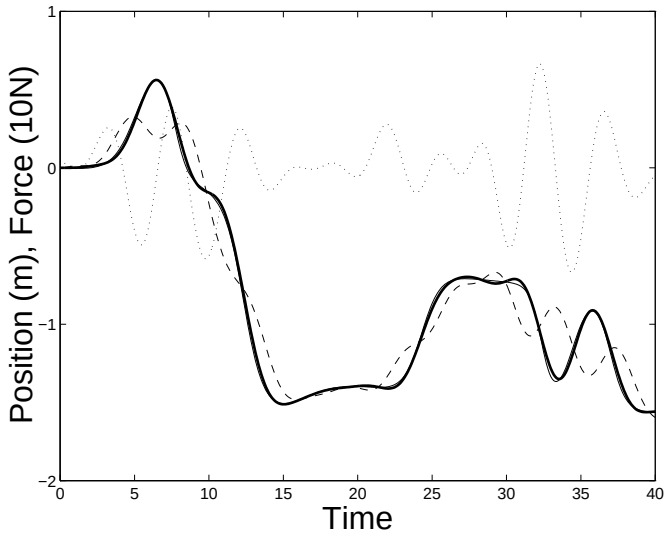



**Figure 6.5**: *Gantry crane controller simulation: The trajectories that result after training is completed (the standard FOX error function with a fourth order eligibility model is used). Key: reference (—), $p_2$ (▬), $p_1$ (– – –), $\mathbf{x}$ ($\cdots$). Note that the reference and $p_2$ are almost coincident.*

### 6.4.2 Reduced order models

In a practical system the matrix $A$ has to be reformulated to prevent such precision problems. The technique of Appendix G was used to get reduced third and second order eligibility models:

$$A^{(3)} = \begin{bmatrix} 1 & 0 & 0.1 \\ 0.095 & 0.918 & 0.161 \\ 0.110 & -0.103 & 1.023 \end{bmatrix} \quad B^{(3)} = \begin{bmatrix} 0.012 \\ 0.002 \\ -0.009 \end{bmatrix} \quad C^{(3)} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{6.18}$$

$$A^{(2)} = \begin{bmatrix} 0.739 & 0.193 \\ -0.326 & 1.230 \end{bmatrix} \quad B^{(2)} = \begin{bmatrix} -0.023 \\ -0.031 \end{bmatrix} \quad C^{(2)} = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{6.19}$$

The corresponding eligibility profiles are shown in figure 6.6. The third order model is an adequate approximation, as it is only substantially different in the first second or so. This is because the eigenvalue that was removed (0.7589) causes rapid decay compared to the others. The second order model is a bad approximation (it does not start near zero). Note that eigenvalue elimination can not give a first order approximation in this case, because the remaining complex eigenvalue will result in a complex $A$ and $B$.

To get the first and second order models a hill-climbing least squares technique was used to minimize the squared error between the models and the full fourth order trajectory (figure 6.7). The resulting models are:

$$A^{(2)} = \begin{bmatrix} 1 & 0.1 \\ -0.0113 & 0.97 \end{bmatrix} \quad B^{(2)} = \begin{bmatrix} 0 \\ 0.0113 \end{bmatrix} \quad C^{(2)} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{6.20}$$

$$A^{(1)} = \begin{bmatrix} 0.9888 \end{bmatrix} \quad B^{(1)} = \begin{bmatrix} 0.1 \end{bmatrix} \quad C^{(1)} = \begin{bmatrix} 1 \end{bmatrix} \tag{6.21}$$

The second order model is not a particularly close match, but it does manage to follow the major peaks of the fourth order model. The first order model is quite dissimilar to the fourth order model. A zero-

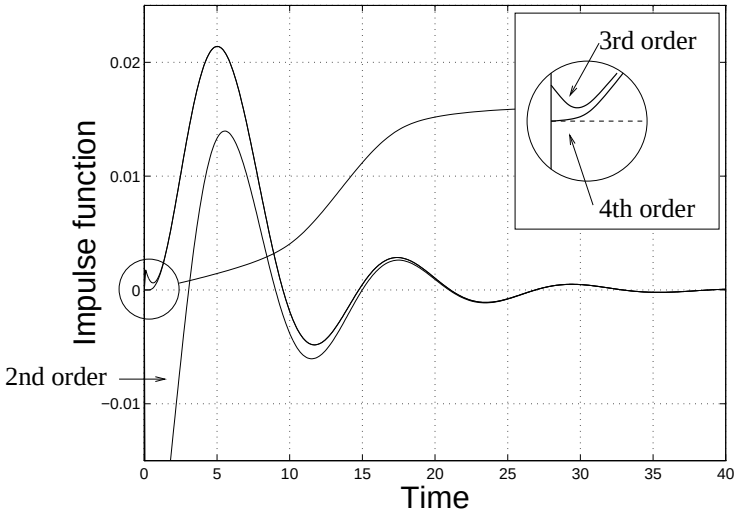



**Figure 6.6**: *Reduced order eligibility profiles of the gantry crane system. The fourth order function is the same as in figure 6.3. The third and second order functions have their first and second (respectively) smallest eigenvalues removed.*

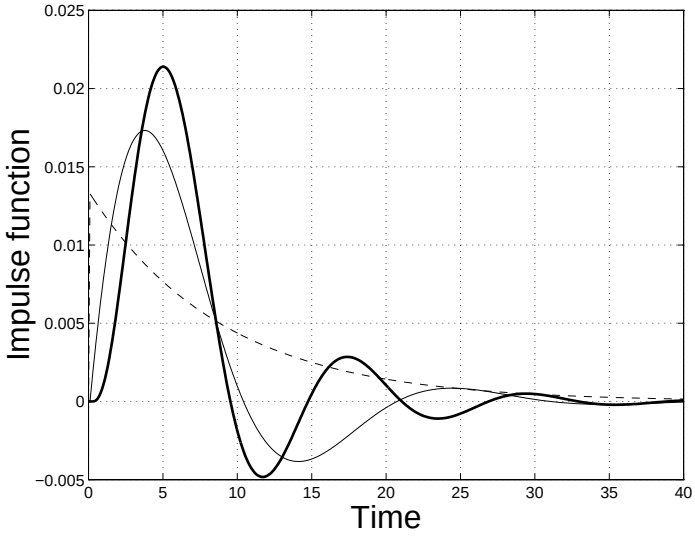



**Figure 6.7**: *The first and second order eligibility models for the gantry crane system, obtained by minimizing the squared error between the models and the full fourth order trajectory. Key: fourth order (—), second order (— —), first order (– – –).*

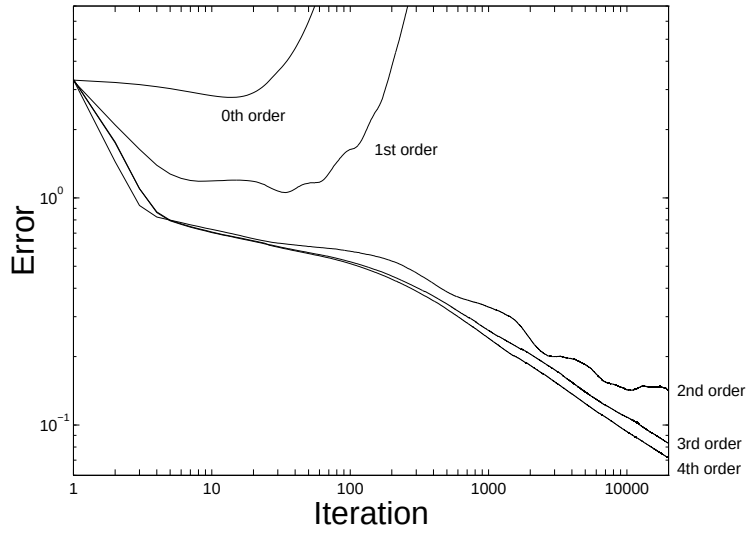



**Figure 6.8**: *Reference error vs number of iterations for eligibility model orders* $0 \ldots 4$.

order model is just a CMAC without any of the FOX eligibility machinery, but for this experiment it was emulated using $A = B = C = 1$ and a history buffer size of 1. A history buffer size of 300 is used for orders $1 \ldots 3$. For all computations involving the reduced order models, IEEE single precision floating point numbers with 7.2 decimal digits of precision were adequate (training times were on the order of several minutes for 20000 iterations).

How effective are the reduced order models? Figure 6.8 shows the error performance when each model is used in the FOX controller (the overshoot error function was used, with $\alpha_1 = 0$, $\alpha_2 = 0.1$). The third order model performs almost as well as the fourth, as would be expected because their eligibility profiles are so similar. The second order model also has a similar performance, though with a slightly slower and more varied convergence rate. The first and zero-order models fail to converge at all (the magnitude of $\mathbf{x}$ grows without bound). Note that the zero-order model corresponds to a standard FBE system, which means than FBE can not be successfully used in this situation. The conclusion is that the FOX eligibility profile must be roughly the same as the system's impulse response, but a large amount of variation is allowed. Other things being equal a lower order model is preferable because it requires less computation in the FOX algorithm. In all subsequent experiments the third order model will be used.

### 6.4.3 Training methods

Different error functions result in different training behavior. Three error functions will be considered here (in isolation and in combination). They are the standard and output limiting error functions controlled by $\alpha$ and $\beta$, and the overshoot error function controlled by $\alpha_1$ and $\alpha_2$. Figure 6.9 shows the effect of these error functions, separately and in combination, with second and third order eligibility models. A reference trajectory with a large discontinuity is used, as shown in figure 6.10.

In figure 6.9 the second order model using the standard error function (curve 6) suffers from overtraining after about 100 iterations—the error increases then starts to vary up and down. The resulting trajectory is shown in figure 6.10. The controller seems to have trouble damping the position $p_2$ after

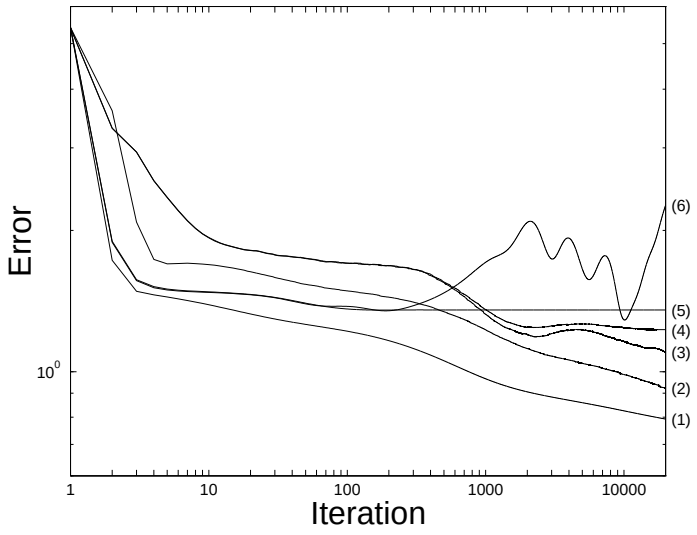



**Figure 6.9**: *Reference error vs number of iterations for different error functions. Key, curves from bottom to top: (1) Third order model, standard error,* $\alpha = 0.05$*. (2) Third order model, overshoot error,* $\alpha_1 = 0.1$*,* $\alpha_2 = 0.01$*. (3) Second order model, overshoot error,* $\alpha_1 = 0.1$*,* $\alpha_2 = 0.01$*. (4) Same as (3) but also with* $\beta = 0.00002$*. (5) Second order model, standard plus output limiting error,* $\alpha = 0.05$*,* $\beta = 0.001$*. (6) Second order model, standard error,* $\alpha = 0.05$*.*

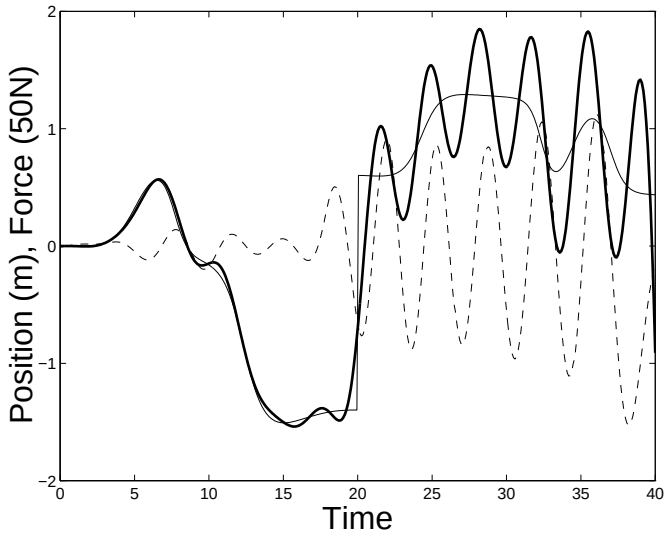



**Figure 6.10**: *The trajectory that results after training with a second order model and a standard error function. Key: reference (—), $p_2$ (**—**), $\mathbf{x}$ (– – –).*

the discontinuity occurs. This is because the second order model is an imperfect match for the crane's impulse response. Adding a small amount of output limiting ($\beta = 0.001$, curve 5) improves the situation—the error reaches its minimum value after about 100 iterations and no further training occurs. Output limiting prevents learning divergence, but it also prevents the error from going below a certain level.

Using the overshoot error function on the second order model provides better convergence (curve 3), although a small amount of over-training occurs. Adding some output limiting reduces the over-training (curve 4), although it also limits the minimum error that can be reached.

The third order model (curves 1 and 2) performs uniformly better than the second order model. The resulting trajectory for curve 1 is shown in figure 6.11. In this case the overshoot error function (curve 2) performs worse than the standard error function (curve 1)—in fact training takes longer but convergence is still achieved. This is because the third order model is a good match to the crane's impulse response.

In summary, more accurate models result in faster learning—less accurate models learn more slowly and can over train. Output limiting prevents over-training but limits the system performance. The overshoot error function seems to be a universally useful way to prevent over-training, and has better performance than output limiting. In a real system it might be preferable to perform output limiting even for an accurate model, to be consistent with the system's physical limitations (i.e. no real motor can generate an arbitrarily high force). But the above comments are only guidelines. In a real system a variety of different error functions should be tried, because performance is very dependent on the system dynamics and reference trajectories that are used.

### 6.4.4 Eligibility profile accuracy

How close should the eligibility profile model be to the system impulse response? As stated above, it is difficult to place formal limits on the eligibility model, but the following experiment will give an indication of the variation that is allowed.

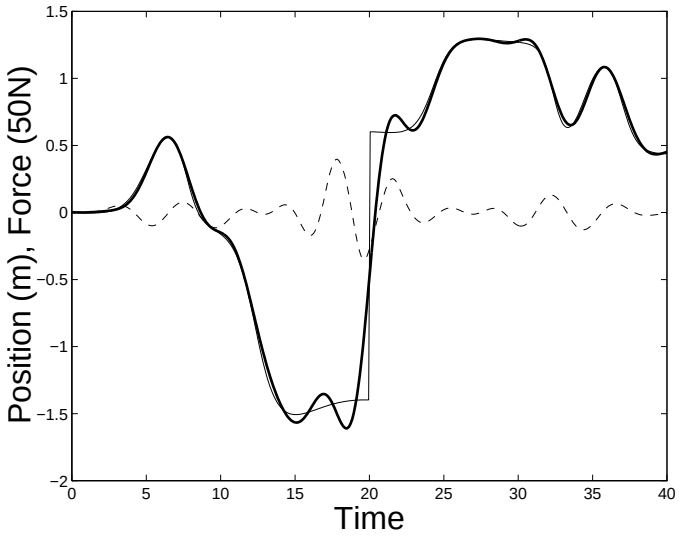



**Figure 6.11**: *The trajectory that results after training with a third order model and a standard error function. Key: reference (—), $p_2$ (▬), $\mathbf{x}$ (– – –).*

Figure 6.12 shows various third order eligibility profiles obtained by replacing the $A$ matrix with $A^k$, where $k$ (the "variation parameter") is in the range $0.5 \ldots 2$. This parameter controls the compression/expansion of the profile. Figure 6.13 shows the error that results after training when each of these profiles is used, with standard error ($\alpha = 0.05$) and overshoot error ($\alpha_1 = 0.1$, $\alpha_2 = 0.01$) training, and a discontinuous reference trajectory. Figure 6.14 shows the error performance of each profile for standard error training (the corresponding graph for overshoot training is similar). Standard error training is successful with $k$ in the range $0.7 \ldots 1.8$. Overshoot error training is successful with $k$ in the range $0.8 \ldots 2.0$. Other values of $k$ cause the error to diverge. Note that higher $k$ tends to result in lower error because the extra area under the eligibility profile curve effectively increases the learning rate. These results show again that a large amount of eligibility profile error is allowable.

### 6.4.5 History buffer size selection

How small can the history buffer size be without affecting training? This question is also related to the bigger question of how much eligibility profile mismatch is allowed, because a small history size truncates the eligibility profile in time. Figure 6.15 shows the error that results after training (for a third order profile) when history sizes from $50 \ldots 350$ are used, with the discontinuous reference trajectory. Any history size above about 200 produces the same training result, because after 200 time steps the eligibility profile has almost fully decayed (figure 6.3). A history size less than about 100 causes divergence as the truncated eligibility profile is too inaccurate. The conclusion continues to be that a large amount of eligibility profile error is allowable.

### 6.4.6 CMAC input configuration

Thus far all experiments have used a CMAC with the current time $t$ as the only input. This has the advantage that FOX can anticipate an arbitrary reference signal as long as the reference value is keyed to

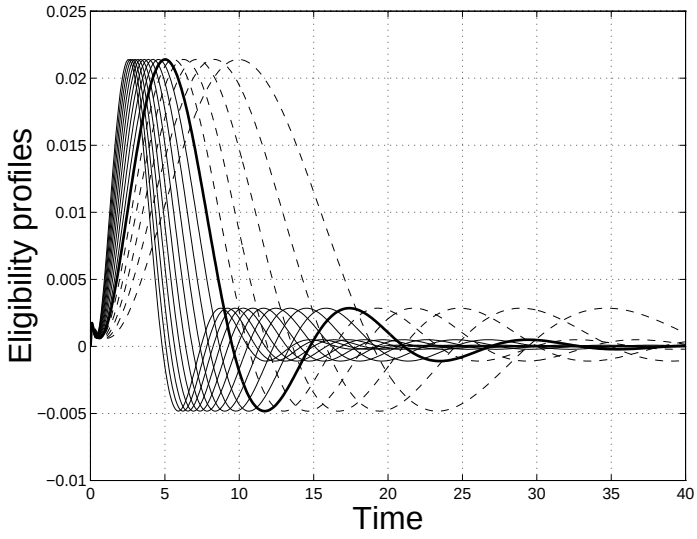



**Figure 6.12**: *Various third order eligibility profiles obtained by replacing the $A$ matrix with $A^k$, where $k$ (the "variation parameter") is in the range $0.5 \ldots 2$. Key: $k > 1$ (—), $k = 1$ (**—**), $k < 1$ (– – –).*

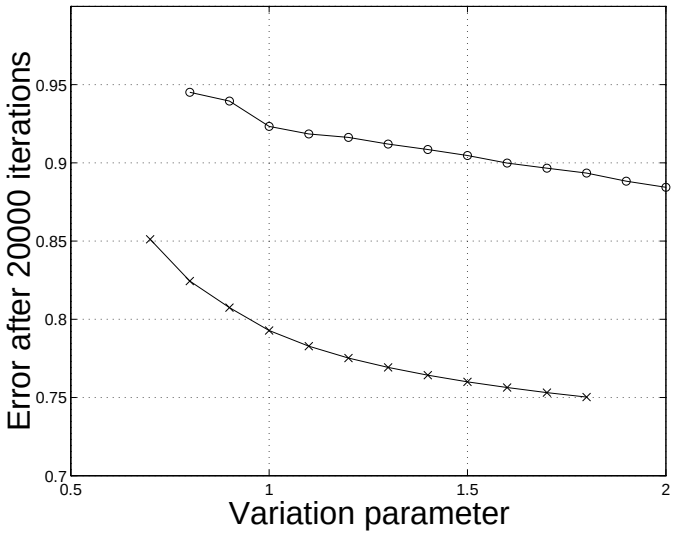



**Figure 6.13**: *Error versus variation parameter after 20000 iterations (third order eligibility profile). Key: standard error function ($\times$), overshoot error function ($\circ$). Unplotted points correspond to simulations with diverging error.*

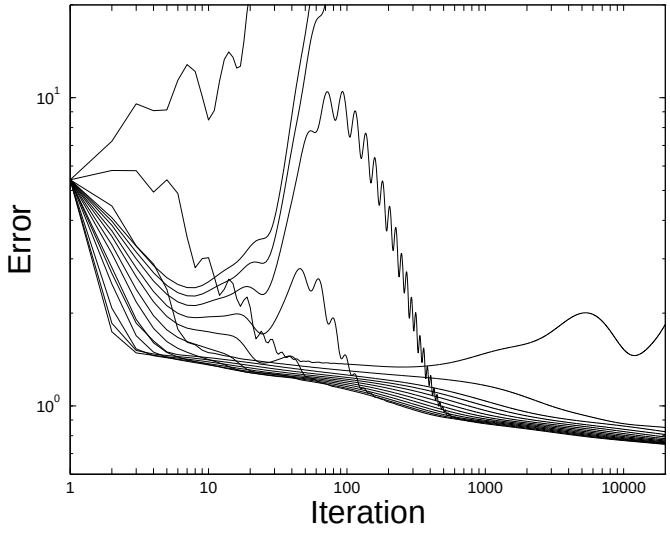



**Figure 6.14**: *Error performance with the various eligibility profiles shown in figure 6.12. The standard error function is used ($\alpha = 0.05$).*

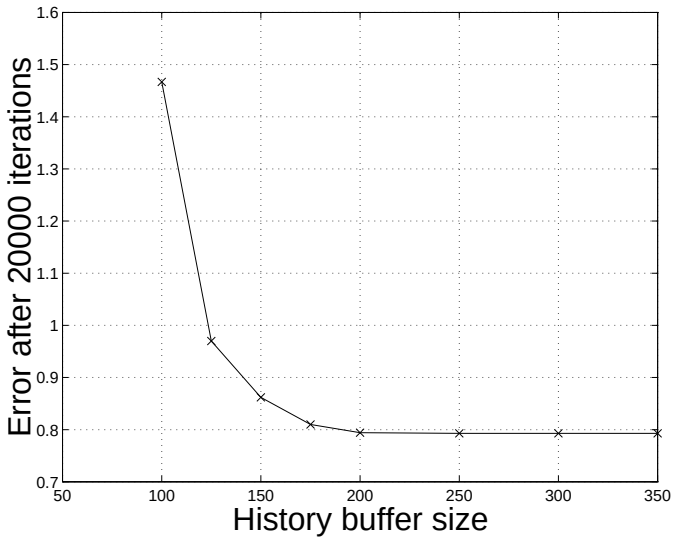



**Figure 6.15**: *Error versus history size after 20000 iterations (third order eligibility profile). A history less than about 100 results in divergence.*

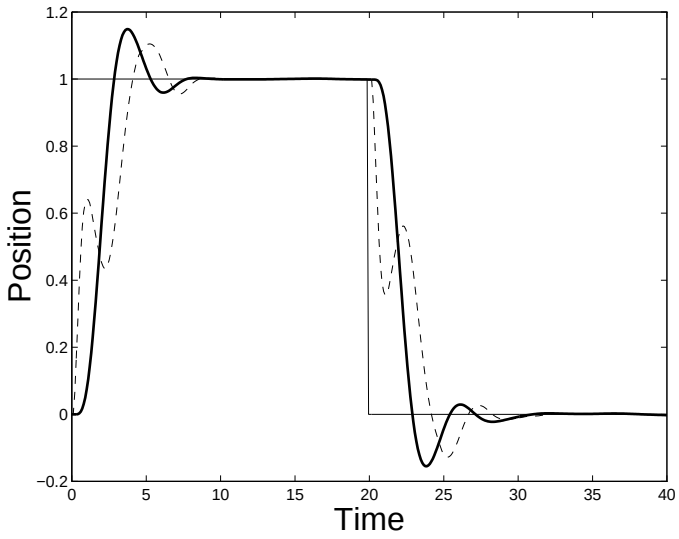



**Figure 6.16**: *The trajectory that results after training, with system state CMAC inputs and reference transition resets (third order eligibility model, standard error). Key: reference (—), $p_1$ (– – –), $p_2$ (▬).*

the time. However this has several disadvantages (outlined above) so in many situations it is desirable to use other input configurations.

A common alternative is to use the system state $\mathbf{y}$ as the CMAC input. Figure 6.16 shows the trajectory that results after 20000 training iterations with a third order eligibility model, and a standard error $\alpha = 0.05$. In this case an arbitrary reference can not be used because the CMAC input does not contain enough information to uniquely identify the current time. A constant or piecewise-constant reference value is commonly used (a two-level reference is used in figure 6.16). Whenever the reference is changed the FOX eligibility information must be zeroed (i.e. the FOX must be reset) as training is effectively being restarted from a different state. As a result the FOX controller can not anticipate the reference change at all. The CMAC input must be reference-independent, i.e. it must contain the position error relative to the current reference value not the position itself. Figure 6.17 shows what happens if a reference transition reset is not performed. Proper convergence can not be achieved because training for times $20 \ldots 40$ interferes with training for times $0 \ldots 20$ even though the two time periods are unrelated (in terms of cause and effect). As a reference, figure 6.18 shows the same system with a time CMAC input.

An alternative hybrid approach is to add an "anticipation" signal to the CMAC state inputs. The anticipation signal somehow indicates that a change in the reference signal is coming. This is demonstrated in figure 6.19, where the anticipation signal goes from -1 to +1 in the 5 seconds before the reference change. In this case resets are not required (in fact they would be disruptive). This combines advantages of both the time-input and state-input approaches (e.g. anticipation is achieved), but it is not always a practical solution.

Figure 6.20 shows the error performance of these three techniques. Time-input is the best in terms of convergence rate, state-input is worst, and the anticipation-signal approach is intermediate between the two.

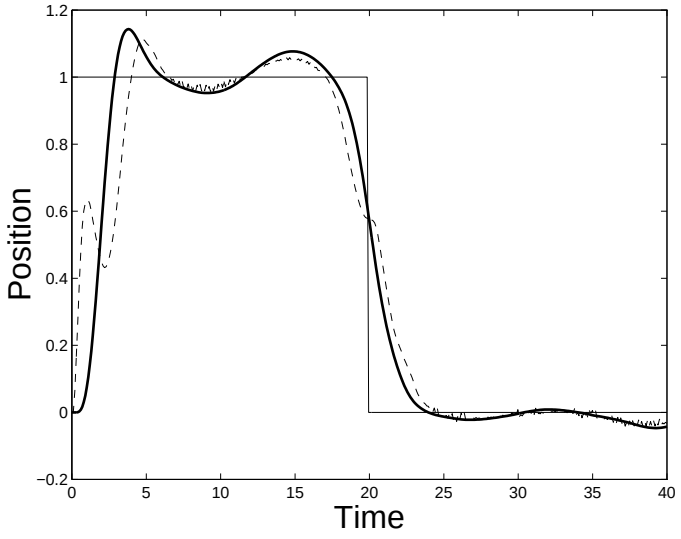



**Figure 6.17**: *The trajectory that results after training, with system state CMAC inputs and no reference transition reset (third order eligibility model, standard error). Key: reference (—), $p_1$ (– – –), $p_2$ (▬).*

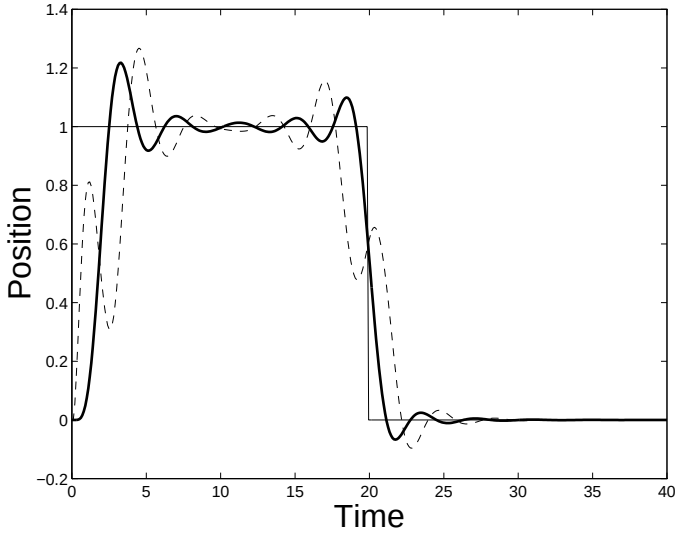



**Figure 6.18**: *The trajectory that results after training, with time CMAC input (third order eligibility model, standard error). Key: reference (—), $p_1$ (– – –), $p_2$ (▬).*

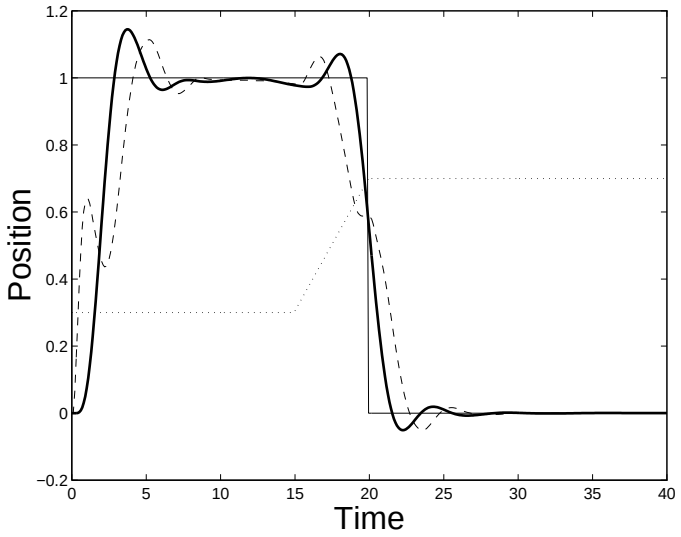



**Figure 6.19**: *The trajectory that results after training, with system state plus anticipation CMAC inputs (third order eligibility model, standard error). Key: reference (—), $p_1$ (– – –), $p_2$ (**—**), anticipation input ($\cdots$). Note that the anticipation input goes from $-1 \ldots 1$ but is scaled to fit the graph.*

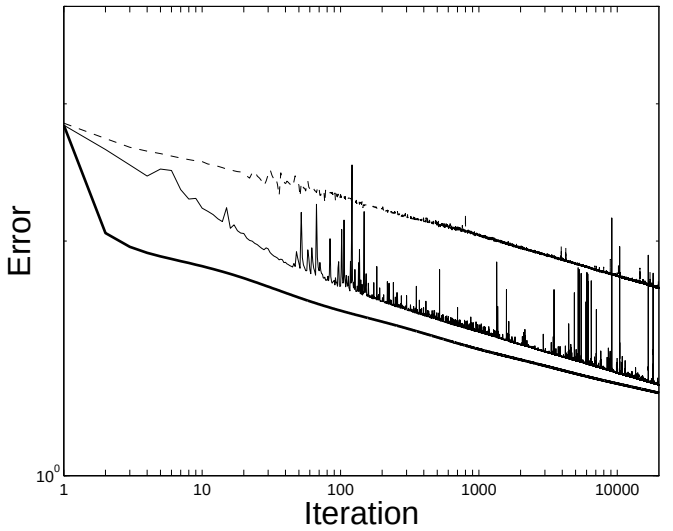



**Figure 6.20**: *Error performance with various CMAC input configurations (third order eligibility model, standard error). Key: time input (**—**), state input (– – –), state and anticipation input (—).*

### 6.4.7 Concluding notes

The following conclusions can be drawn from the crane simulation experiments:

- The eligibility model can not contain a mixture of large and small eigenvalues if a reasonable history size is required. Various techniques must be used to remove or replace bad eigenvalues.
- A large difference between the system's impulse response and the eligibility profile can be tolerated. Such differences arise from reduced order or inaccurate eligibility models, or short history buffers. The problem remains to quantify the exact amount of tolerable difference. Note that model accuracy does not necessarily equate with model order, although higher order models have the potential to be more accurate.
- Lower order eligibility models are good, because they consume less memory and computing power, and also because they are less likely to contain conflicting eigenvalues. However, models should not be simplified to the point where they are too inaccurate.
- More accurate eligibility models result in faster training. Less accurate eligibility models are susceptible to over-training. This can be corrected by using an output limiting or overshoot error function. The overshoot error function is generally useful—it makes many systems more stable and has a minimal cost in convergence rate.

## 6.5 Testing: Gantry crane

A FOX controller was constructed for the control of a model gantry crane. A picture of the experimental hardware is shown in figure 6.21, and a schematic is shown in figure 6.22. The crane can move horizontally on a set of rails. It is connected by a toothed belt and pulleys to a small DC motor via a gearbox (ratio 25:1). The crane supports a load (a lead weight) by a steel cable which is free to swing left and right.

An analog sensor measures the cable angle $\theta$ (in degrees) and a shaft encoder on the gearbox measures the crane position $p_c$ (in meters). The load position $p_\ell$ is calculated from the cable length $\ell$ (assuming a small angle) by

$$p_\ell \;=\; p_c + \ell\,\frac{\pi}{180}\,\theta \qquad \text{(m)} \tag{6.22}$$

An electronics box contains the high power motor driver circuit which accepts a bipolar voltage from a D/A converter. There is also an A/D converter and shaft-encoder decoder circuit for sensor digitization.

A block diagram of the crane controller algorithm is shown in figure 6.23. Table 6.3 gives the experiment and CMAC parameters. A linear PD controller provides the "internal" feedback loop. It tries to keep the load angle at zero by implementing:

$$\mathbf{x}_\ell \;=\; -0.02\,\theta - 0.003\,\frac{\mathrm{d}\theta}{\mathrm{d}t} \tag{6.23}$$

These PD coefficients, which provide adequate control, were found by trial and error. The main feedback loop contains a FOX controller which is trained to make the load velocity equal to a desired velocity computed from the crane position error as follows:

$$\text{desired load velocity} \;=\; -0.6\left(\frac{1}{1+e^{-10(p_c-\text{ref})}} - 0.5\right) \qquad \text{(m/s)} \tag{6.24}$$

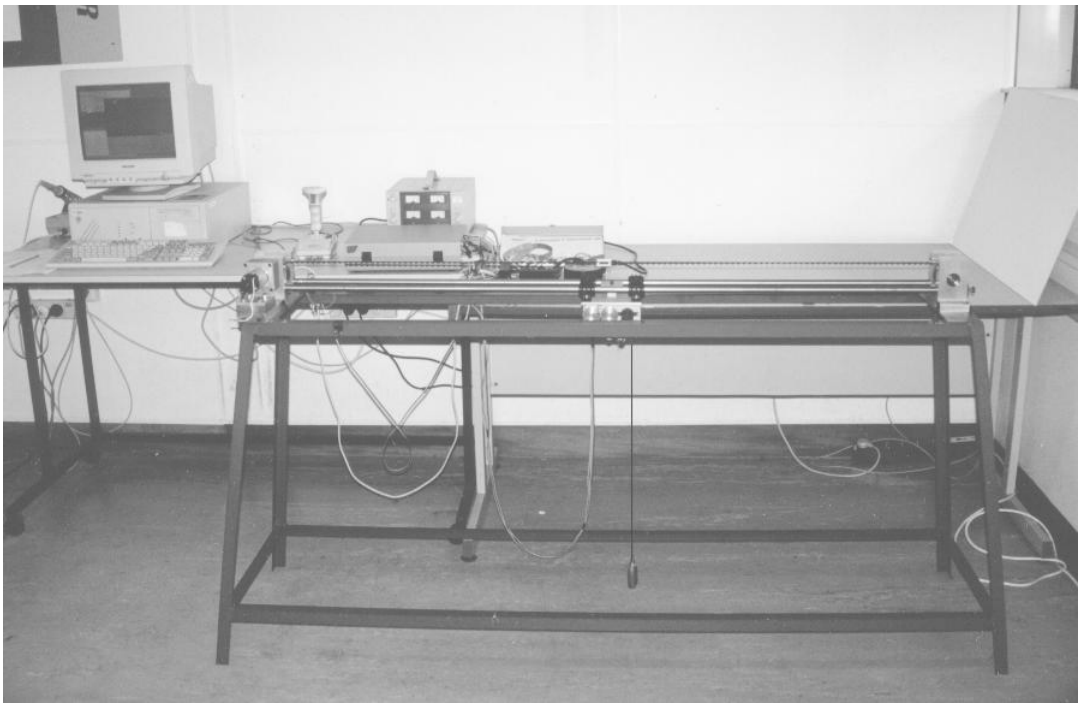


**Figure 6.21**: *A picture of the gantry crane experimental hardware.*

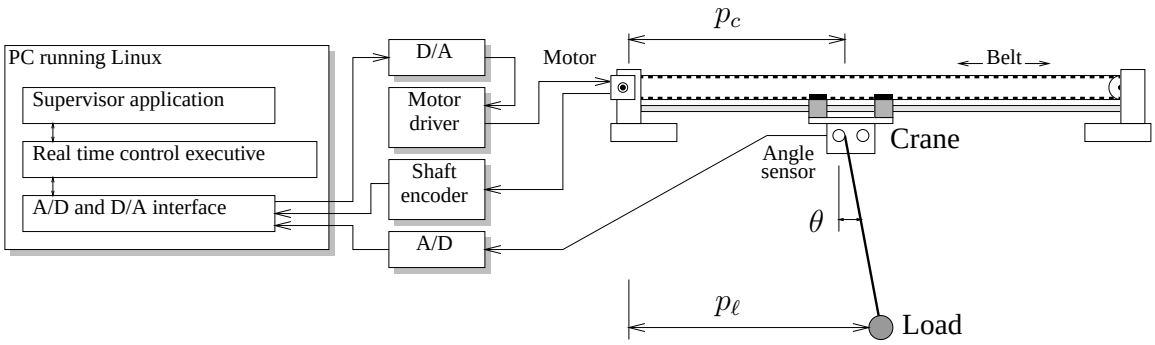



**Figure 6.22**: *A schematic of the gantry crane experimental system.*

| | |
|---|---|
| System time step ($h$) | $1/64$ s |
| Standard error FOX learning rate ($\alpha$) | 0.0005 |
| CMAC number of inputs | 4 |
| CMAC resolution on each input | 200 |
| CMAC number of weights | 100000 |
| CMAC $n_a$ | 20 |
| FOX default eligibility model order | 2 |

**Table 6.3**: *Standard gantry crane experiment parameters, including CMAC parameters. Most of these parameters are used for most of the experiments.*

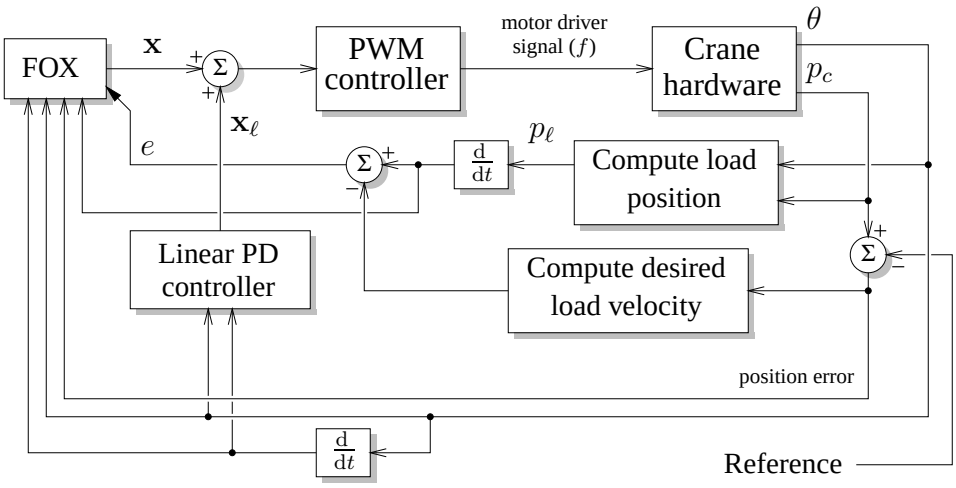



**Figure 6.23**: *A block diagram of the crane controller algorithm.*

Thus when the crane is far away from its reference position the desired velocity will be $\pm 0.3$ m/s, and as it gets closer to the reference the desired velocity will smoothly ramp down to zero.

The time derivatives are computed from the sensor data using third and fifth order digital Butterworth IIR filters. A software PWM driver is used to try and compensate for stickiness within the gearbox, which prevents low motor drive signals from moving the crane. If the motor drive signal is above a threshold it is sent directly to the D/A converter. If it is below this threshold it is used to modulate a PWM signal whose duty cycle is calibrated to push the motor along at the appropriate speed. This provides a more linear low speed motor response, although it does cause the motor to emit an audible rattling sound at the system sampling frequency of 64Hz.

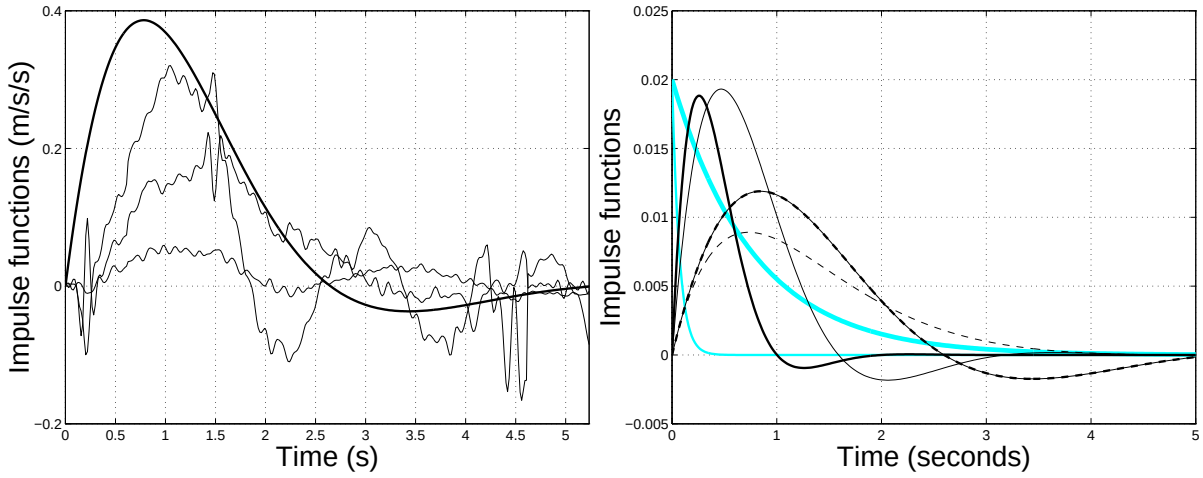
The crane mechanical system has four state variables[2]. The FOX inputs represent each state variable. Note that the position variables are reference independent. The reference signal is a squarewave that alternates between $\pm 3$ every six seconds. The FOX is reset on each reference change.

### 6.5.1 Eligibility profiles

The gantry crane system is very nonlinear. Even an "ideal" crane model is intrinsically nonlinear, and this system also has nonlinear friction (mainly in the gearbox) and the drive electronics have a saturating response.

Strictly speaking, the impulse response of a nonlinear system is not a well defined concept—but the simplest approach is just to treat it as a linear system and try to measure the characteristic time constants of the system's step response. Figure 6.24a shows the error derivative for three different force steps. The responses have different forms because of system nonlinearities, though each one looks roughly like a decaying oscillation. The thick line in this plot shows a second order eligibility profile which roughly matches the experimental data. Its parameters were selected "by eye" (they are $k_a = 6$, $k_b = 3$, refer to Appendix E for the meaning of these constants). This profile was used for most of the following experiments.

[2] At *least* four—more accurate models will have more state variables.




(a) (b)

**Figure 6.24**: ***(a)*** *The three thin lines are measured impulse responses of the gantry crane system, equal to* $\mathrm{d}e/\mathrm{d}t$ *for a step force input. Each impulse was measured for a different force value. The impulses have different forms because of system nonlinearities. The thick line is the second order eligibility model used for most of the crane experiments.* ***(b)*** *Black lines: the four second order eligibility profiles used in figure 6.28a. Grey lines: the two first order eligibility profiles used in figure 6.28b. Key: second order* $k_a = 6$ *and* $k_b = 3$ *and* $\sigma = 200$ *(—), second order* $k_a = 18$ *and* $k_b = 6$ *and* $\sigma = 100$ *(—), second order* $k_a = 2$ *and* $k_b = 2.5$ *and* $\sigma = 250$ *(– – –), second order* $k_a = 2$ *and* $k_b = 1.5$ *and* $\sigma = 310$ *(– – –), first order* $k_a = 0.98$ *and* $\sigma = 200$ *(—gray), first order* $k_a = 0.8$ *and* $\sigma = 20$ *(—gray).*

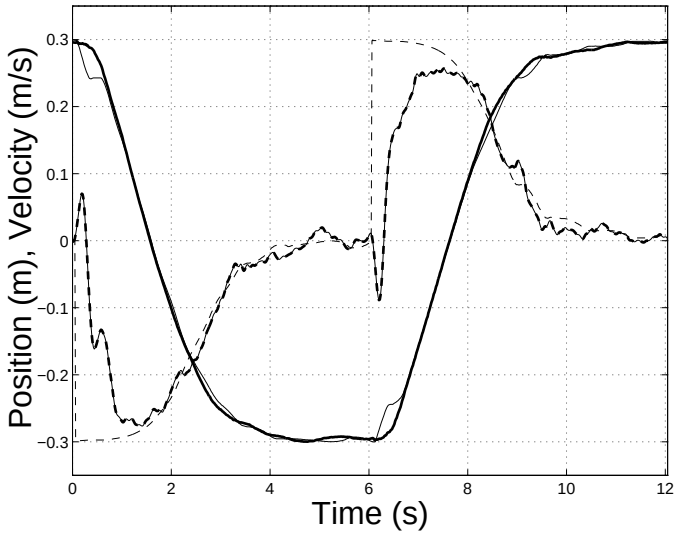



**Figure 6.25**: *Gantry crane experiment results. The best time domain performance achieved, with the overshoot error function,* $\alpha_1 = 0.0005$ *and* $\alpha_2 = 0.005$*. Key: cart position* $x$ *(—), mass position* $p$ *(—), desired* $\mathrm{d}p/\mathrm{d}t$ *(– – –), actual* $\mathrm{d}p/\mathrm{d}t$ *(– – –).*

### 6.5.2 Experiments

As before the following learning rate parameters were used: main rate ($\alpha$), output limiting ($\beta$), output derivative limiting ($\gamma$), and the overshoot learning rates ($\alpha_1$, $\alpha_2$).

The best system performance was achieved using the overshoot error function with $\alpha_1 = 0.0005$ and $\alpha_2 = 0.005$. Figure 6.25 shows the time domain response that was achieved after 100 learning iterations, or 600 seconds (each iteration corresponds to one transition in the squarewave reference signal).

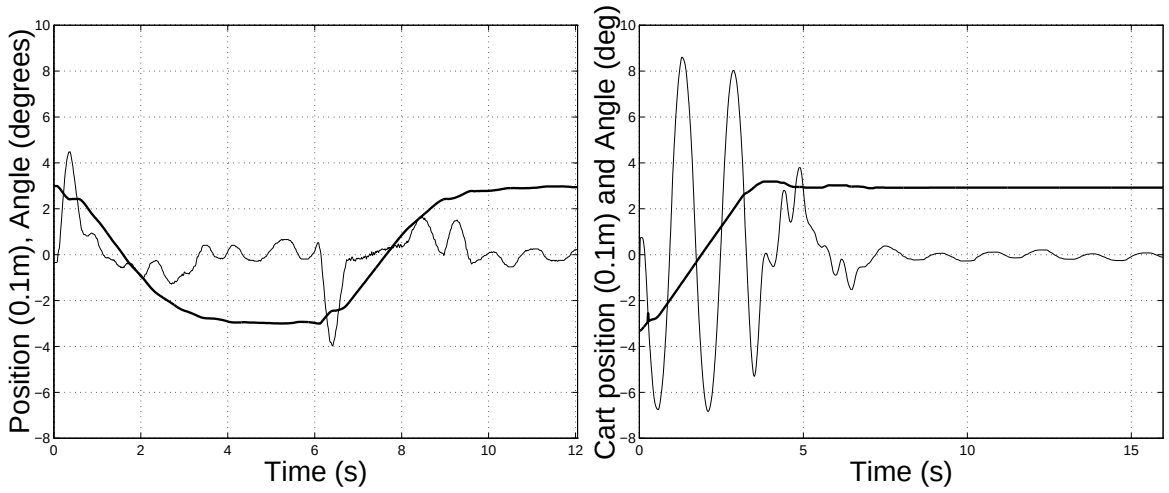
There are several points to note: the load velocity quickly reaches the desired velocity and then tracks it reasonably closely as the crane approaches its reference position. The crane has an initial quick movement to get the load up to speed. The short initial negative spike in load velocity is a quirk of the way load position is measured. The crane smoothly approaches and decelerates towards the reference. There is almost no reference overshoot. At all points the crane position and load angle are well controlled.

Figure 6.26 compares this performance with the best that was achieved using a linear PID controller [16]. The FOX based system achieves better control of the load angle and has less crane position overshoot.

Figure 6.27a shows the error performance for different overshoot learning rates. The total error for each six second iteration is computed as:

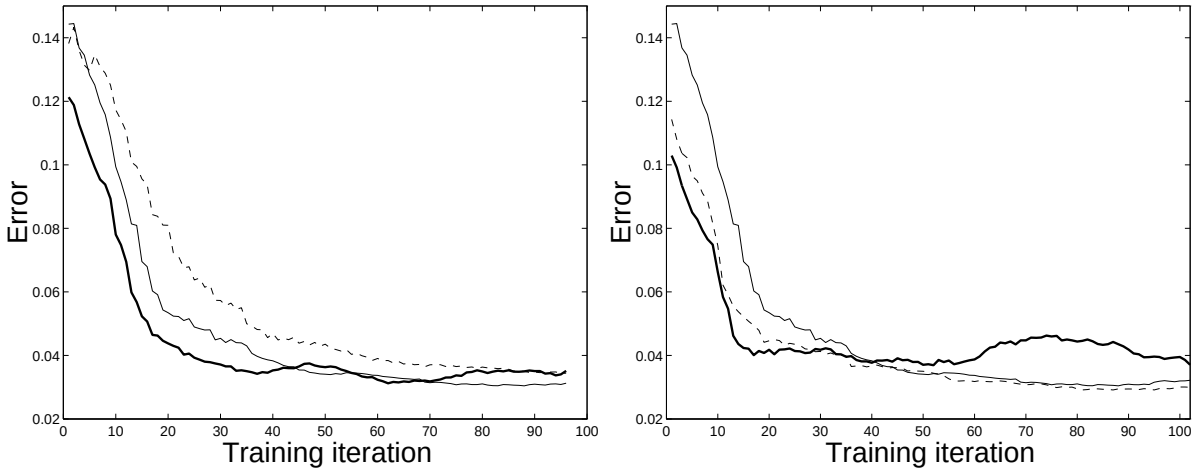
$$E = \sum_i h\, e_i^2 \tag{6.25}$$

This plot shows the expected behavior—increasing $\alpha_2$ improves the training speed. Figure 6.27b shows the overshoot error performance when both $\alpha_1$ and $\alpha_2$ are varied. It shows that higher learning rates increase the training speed, but if they are too high ($\alpha_2 = 0.025$) then over-training will occur (the error eventually increases instead of monotonically decreasing). The time domain response in this case is shown in figure 6.30a. Notice how the crane position and load velocity oscillate instead of smoothly tracking their references—this is the hallmark of over-training.



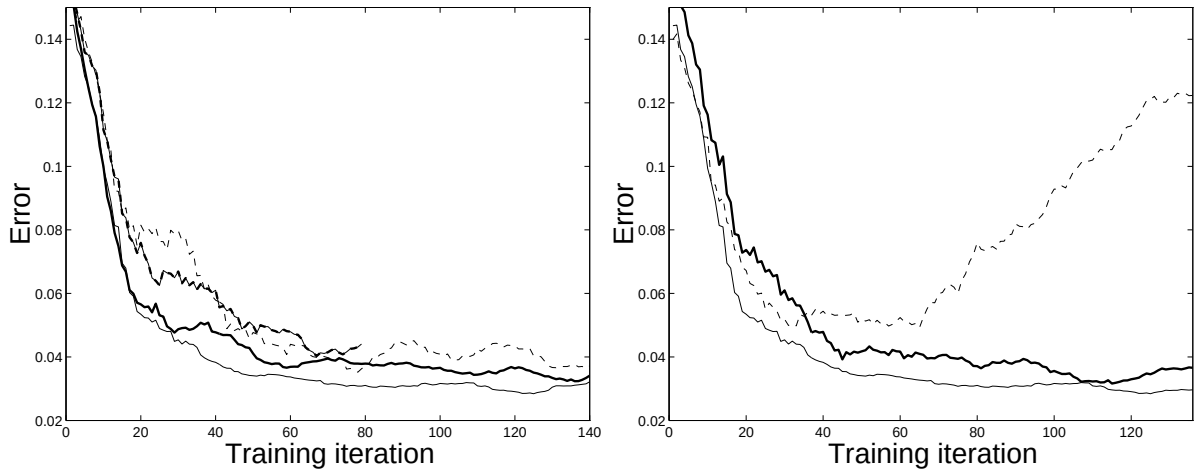

(a) (b)

**Figure 6.26**: *Gantry crane experiment results (time domain).* ***(a)*** *The same as figure 6.25 but only showing the cart position and load angle.* ***(b)*** *The best response achieved with a PID controller [16] (note that there is only one step change in the reference). Key: cart position $x$ (—) load angle $\theta$ (—).*




(a) (b)

**Figure 6.27**: *Gantry crane experiment results.* ***(a)*** *Error performance for different overshoot error learning rates (all other constants have their standard values). Key: $\alpha_2 = 0.01$, $\alpha_1 = 0.0005$ (—), $\alpha_2 = 0.005$, $\alpha_1 = 0.0005$ (—), $\alpha_2 = 0.0025$, $\alpha_1 = 0.0005$ (– – –).* ***(b)*** *Error performance for different overshoot error learning rates (all other constants have their standard values). Key: $\alpha_2 = 0.025$, $\alpha_1 = 0.0025$ (—), $\alpha_2 = 0.005$, $\alpha_1 = 0.0005$ (—), $\alpha_2 = 0.001$, $\alpha_1 = 0.0001$ (– – –).*




(a) (b)

**Figure 6.28**: *Gantry crane experiment results.* ***(a)*** *Error performance for different second order eligibility profiles. The line styles correspond to the second order eligibility profiles shown in figure 6.24b.* ***(b)*** *Error performance for the first order eligibility profiles shown in figure 6.24b. Key: The best second order eligibility profile, as a reference (—),* $k_a = 0.98$ *(—),* $k_a = 0.8$ *(– – –).*

Figure 6.28a shows the error performance for different second order eligibility profiles, corresponding to the profiles shown in figure 6.24b. These profiles have a range of different decay constants but roughly the same form. In each case the learning rates were adjusted to achieve the best performance. They all result in roughly the same training behavior, although profiles that take longer to decay show slightly reduced performance.

Figure 6.28b shows the same thing for first order eligibility profiles. A first order profile with $k_a = 0.98$ results in worse performance than the best second order profile (although it is still acceptable). The profile with $k_a = 0.8$ is effectively zero-order, because it decays so quickly. It results in bad over-training after about the 60th iteration. The time domain response in this case is shown in figure 6.30b. The crane and load are subject to large oscillations, which only get worse with more training.

Figure 6.29 shows the error performance for the four different types of error function. In each case the learning rates were adjusted to achieve the best performance. The overshoot error function has the best performance, standard training (with no variety of output limiting) has the worst performance (with a significant over-training problem). Output limiting and output derivative limiting have intermediate performance, with little over-training.

### 6.5.3 Concluding notes

The following conclusions can be drawn from these experiments:

- FOX is capable of controlling nonlinear systems as well as linear ones.
- The overshoot-error training method provided the best performance, while standard-error training was the worst.
- Over-training occurred in all cases if the learning rate was too high.

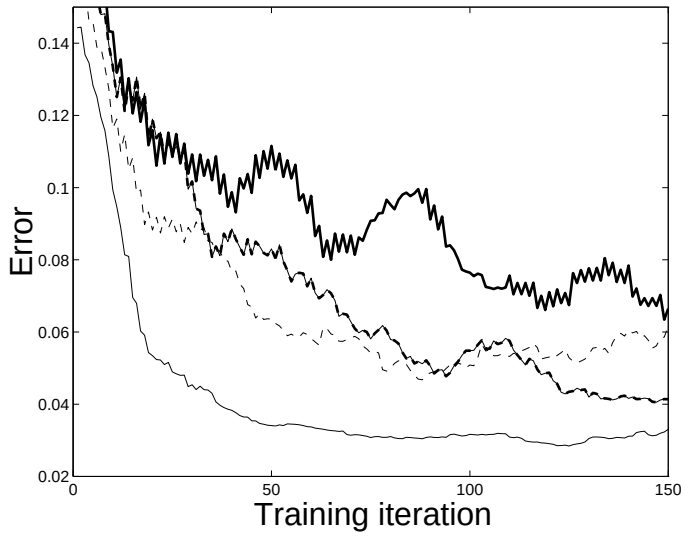
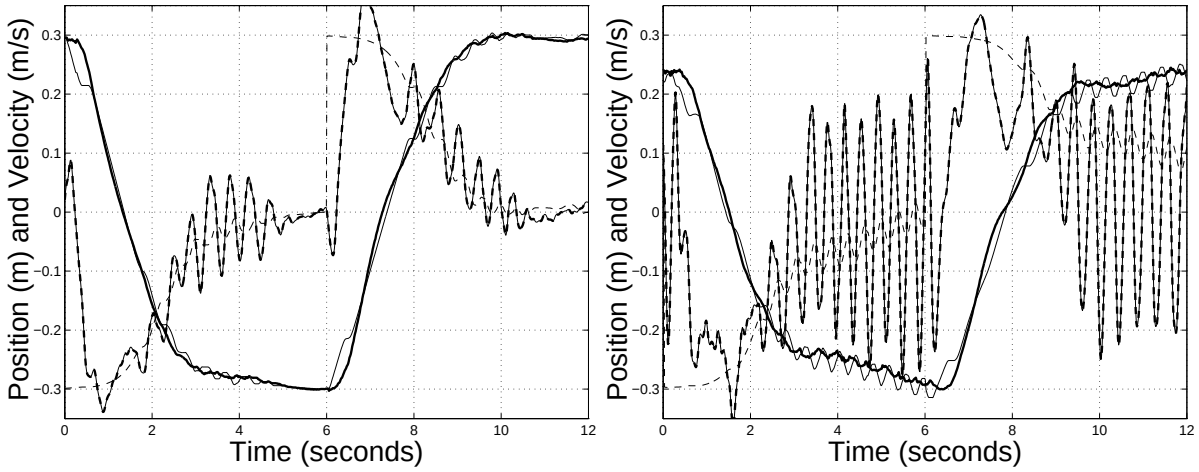



**Figure 6.29**: *Gantry crane experiment results. Error performance for different error measures. Key: overshoot training with $\alpha_2 = 0.005$ and $\alpha_1 = 0.0005$ (—), standard training with $\alpha = 0.0005$ (– – –), output limiting with $\alpha = 0.0005$ and $\beta = 0.0005$ (—), output derivative limiting with $\alpha = 0.0005$ and $\gamma = 0.0002$ (– – –).*




(a) (b)

**Figure 6.30**: *Gantry crane experiment results (time domain).* ***(a)*** *Response with the standard error function, $\alpha = 0.0025$.* ***(b)*** *Response with first order eligibility profile, $k_a = 0.8$. Key: cart position $x$ (—), mass position $p$ (—), desired $\mathrm{d}p/\mathrm{d}t$ (– – –), actual $\mathrm{d}p/\mathrm{d}t$ (– – –).*

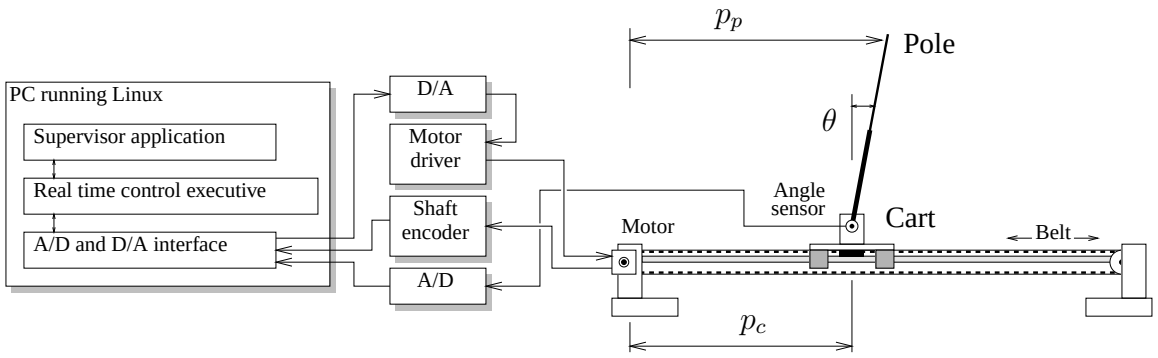



**Figure 6.31**: *A schematic of the inverted pendulum experimental system.*

- Large variations in the eligibility profile are allowed. Even a first order profile will work adequately for this system, but a zero-order profile will not. Thus eligibility is an important factor in the system's success—a straight FBE controller would not work.

## 6.6 Testing: Inverted pendulum

The fourth order inverted pendulum problem involves keeping a pole balanced on a "cart" by applying a horizontal force to the cart (figure 6.31). This is a widely studied "hard" problem in control theory, because although it can be solved adequately with standard non-adaptive linear control techniques, some extra elements of "intelligent control" are usually necessary for a good solution. Techniques using neural networks [47], reinforcement learning [10] and feedback-error with an expert rule base [42] have all been applied to it.

Two experiments were conducted to determine how easy it is for FOX to control this system, using the minimum amount of dynamical information. An AVI movie is provided on the CDROM that accompanies this thesis (`inverted.avi`) which shows most of the experimental results.

A schematic of the experimental system is given in figure 6.31 and two photos are shown in figure 6.32. The inverted pendulum system was based on the same hardware as the crane system, except that the crane platform was replaced with a cart-and-pole. A potentiometer sensor on the cart measured the pole angle $\theta$, in degrees. As before, a shaft encoder measures the cart position $p_c$ (in meters). The pole position $p_p$ was calculated from the pole length $\ell$ (assuming a small angle) by

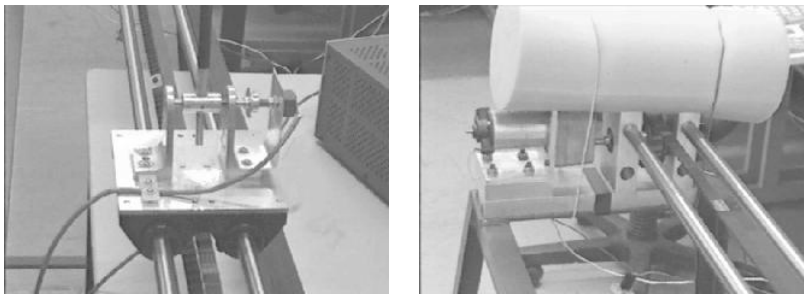
$$p_p \;=\; p_c + \ell\,\frac{\pi}{180}\,\theta \qquad \text{(m)} \tag{6.26}$$

The 25:1 gearbox was changed to a 5:1 ratio to get faster cart movement.

A block diagram of the inverted pendulum controller algorithm is shown in figure 6.33. Table 6.4 gives the experiment and CMAC parameters. A linear PD controller provides the internal feedback loop. It tries to keep the pole angle at zero and the cart position at a reference by implementing:

$$\mathbf{x}_\ell \;=\; -0.5(p_c - \text{ref}) - 0.25\,\frac{\mathrm{d}p_c}{\mathrm{d}t} - 0.06\,\theta - 0.000625\,\frac{\mathrm{d}\theta}{\mathrm{d}t} \tag{6.27}$$

The main feedback loop contains a FOX controller which is trained to get the cart position equal to the reference with the error $e = p_c - \text{ref}$. The desired pole angle is unspecified, but this is not a problem



(a) (b)

**Figure 6.32**: *Two pictures of the inverted pendulum experimental system. (a) The cart on its rails, showing the base of the pole and the angle sensor. (b) The motor, gearbox and drive-train.*

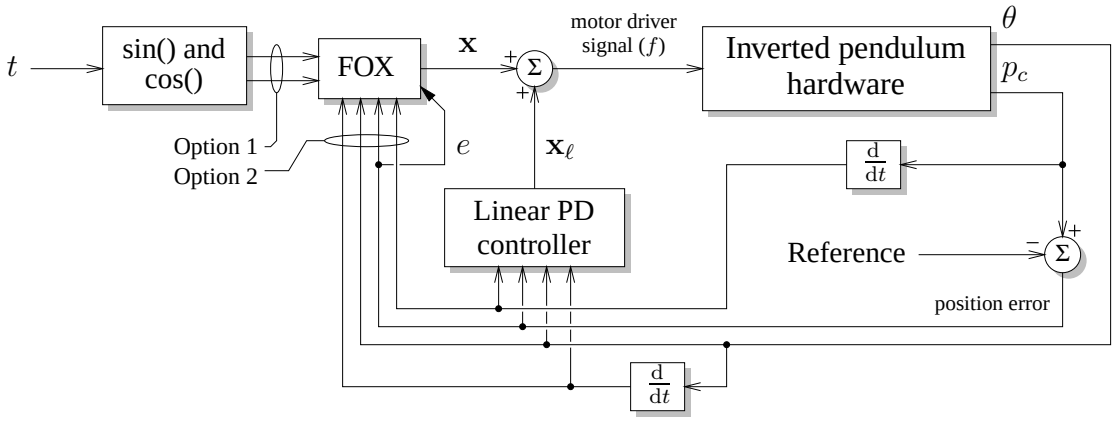



**Figure 6.33**: *A block diagram of the inverted pendulum controller algorithm.*

because the cart can not attain a reference position without the pole angle being simultaneously well controlled. Hardly any effort was made to choose good linear controller coefficients, on the assumption that the FOX controller would be able to compensate. Different CMAC input configurations were used for different experiments—they are described below.

### 6.6.1 Finding the eligibility profile

The impulse response of the inverted pendulum (with its linear controller) was measured by moving the cart to $p_c = 0$, carefully balancing the pole at $\theta = 0$, then gently tapping the top of the pole. As shown in figure 6.34, the cart quickly moved to compensate for the changing pole angle, but over-corrected and started to oscillate. The system converged to a limit cycle (due to its nonlinearity). No effort was made to tune the linear controller to reduce or eliminate this oscillation. Note that the flat spots in the cart position at its extremities are due to mechanical slip in the drive-train.

FOX can not model such an impulse response. And even if it could, it is unclear whether this is desirable because of the nonlinear nature of the system. Instead, a second order eligibility profile was chosen that roughly matches the time scale of the oscillations (figure 6.34). This eligibility profile decays

| | |
|---|---|
| System time step ($h$) | $1/64$ s |
| CMAC number of inputs | 2 or 4 |
| CMAC resolution on each input | 256 or 100 |
| CMAC number of weights | 100000 |
| CMAC $n_a$ | 20 |
| FOX second order eligibility model parameters | $k_a = 1.5$, $k_b = 1.7$, $\sigma = 384$ |

**Table 6.4**: *Standard inverted pendulum experiment parameters, including CMAC parameters.*

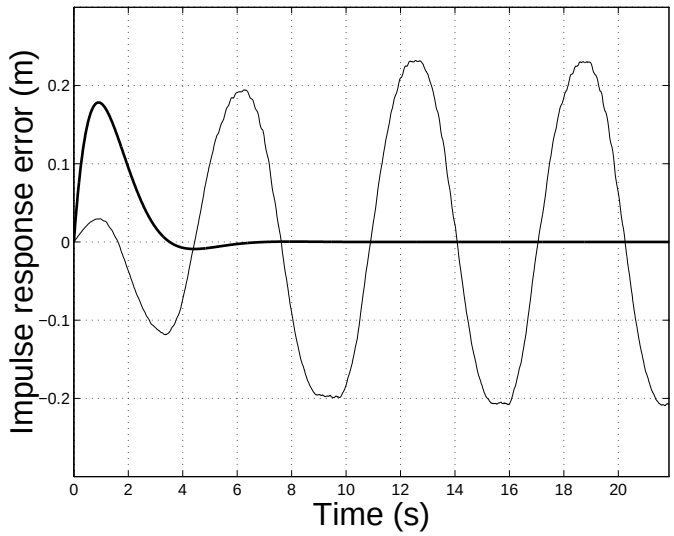



**Figure 6.34**: *Key: Impulse response (error $e$) of the inverted pendulum system (—). Eligibility profile model (**—**).*

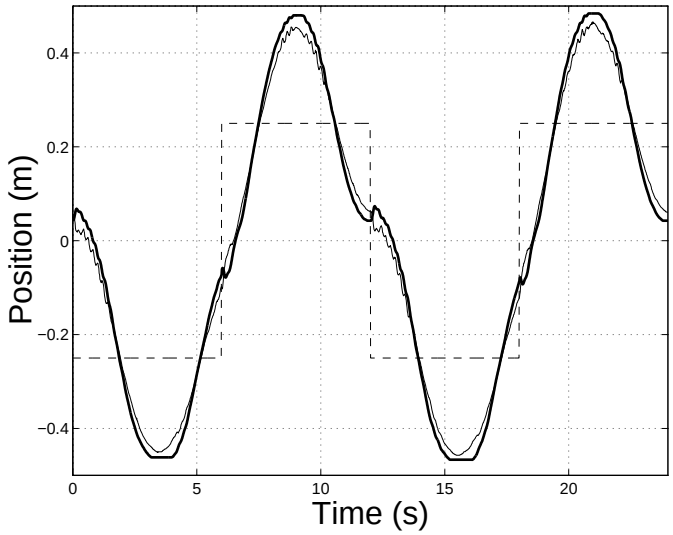



**Figure 6.35**: *Response of the feedback FOX inverted pendulum system to a squarewave reference, at the start of training. Key: cart position $p_c$ (—), pole position $p_p$ (—), reference (– – –).*

to zero, which is probably more appropriate for control than an oscillatory profile, as pointed out in section 6.2.1.

### 6.6.2 Learning feedback control experiment

In this experiment FOX was used in a feedback control mode, i.e. the CMAC inputs were the four system state variables (reference independent). A square wave reference was used that changed between $\pm 0.25$ every 6 seconds. The FOX was reset on each reference change. It was determined that the standard error function with output derivative limiting gave a good response ($\alpha = 0.005$ and $\gamma = 0.003$).

Figure 6.35 shows the response at the start of training. The cart position occasionally crosses the reference, but it is clear that the linear controller is insufficient to the task. Figure 6.36 shows the response after 60 training iterations (360 seconds—each iteration is a transition of the reference). The response has improved greatly: the cart position smoothly approaches the reference and there is little overshoot. The pole angle is similarly well controlled. Convergence has been achieved at this point, as further training does not improve the response much (various electrical and mechanical nonlinearities mean that a perfect response is difficult to achieve). The output derivative limiting means that over-training is not a problem.

Of the 100000 CMAC weights, only 12359 were nonzero after training (12.4%). This implies that a much smaller weight set could have been used without affecting performance.

### 6.6.3 Learning feed-forward control experiment

In this experiment the task was to follow the reference signal

$$\text{ref} \quad = \quad \sin\left(\frac{\pi}{2}t\right) + \cos(\pi t) \tag{6.28}$$

which repeats every four seconds. In this case the CMAC inputs must somehow encode the reference position, so the FOX was used in a feed-forward control mode, with two inputs representing time $t$ like

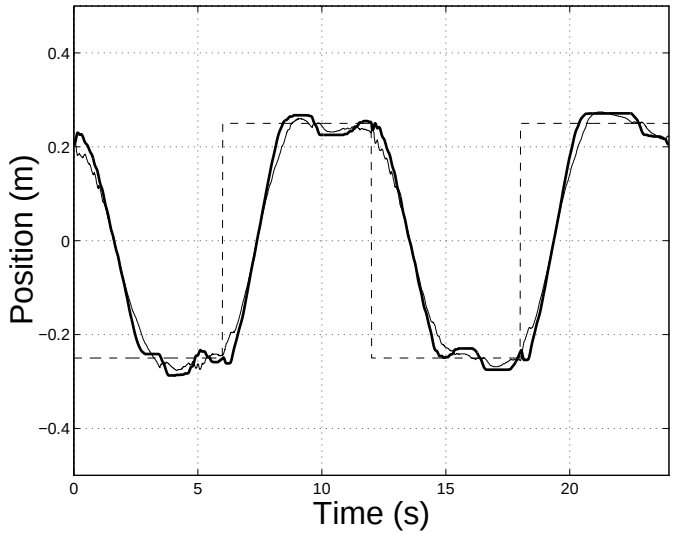



**Figure 6.36**: *Response of the feedback FOX inverted pendulum system to a squarewave reference, after 60 training iterations. Key: cart position $p_c$ (—), pole position $p_p$ (—), reference (– – –).*

this:

$$\text{input 1} \quad = \quad \sin\left(\frac{\pi}{2}t\right) \tag{6.29}$$

$$\text{input 2} \quad = \quad \cos\left(\frac{\pi}{2}t\right) \tag{6.30}$$

The time $t$ by itself can not be used because then the CMAC would not generalize over those parts of the reference that are the same. The following "wrapped time" is also inappropriate:

$$\text{input 1} \quad = \quad \frac{t}{4} - \left\lfloor \frac{t}{4} \right\rfloor \qquad (\text{in the range } 0 \ldots 1) \tag{6.31}$$

because the input will jump discontinuously from 1 to 0 at the end of each cycle which will interfere with the CMAC's generalization ability. The CMAC input contains no information about the actual state of the system, so whenever the trained controller is used the system must start close to a known state or the behavior will be unpredictable. The standard error function was used, with $\alpha = 0.01$.

Figure 6.37 shows the response at the start of training. The cart is unable to follow the reference at all. Figure 6.38 shows the response after 120 cycles of training (480 seconds). The cart position now matches the reference closely. The pole position was unspecified by the error function, but training has given it a trajectory appropriate to the cart position. The error performance during training is shown in figure 6.39.

Of the 100000 CMAC weights, only 3762 were nonzero after training (3.7%). The weight usage is a lot smaller than for the first experiment because the range of input trajectories seen is far more restricted.

### 6.6.4 Concluding notes

Hougen *et al* [47] describe the inverted pendulum problem as so difficult that it has not been adequately solved by a learning controller until recently. However, their version of the problem gives the controller

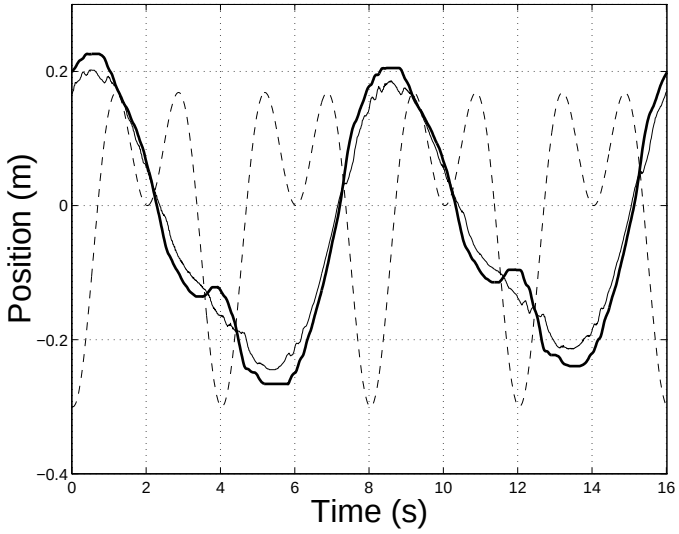



**Figure 6.37**: *Response of the feed-forward FOX inverted pendulum system to a non-constant reference, at the start of training. Key: cart position $p_c$ (—), pole position $p_p$ (—), reference (– – –).*

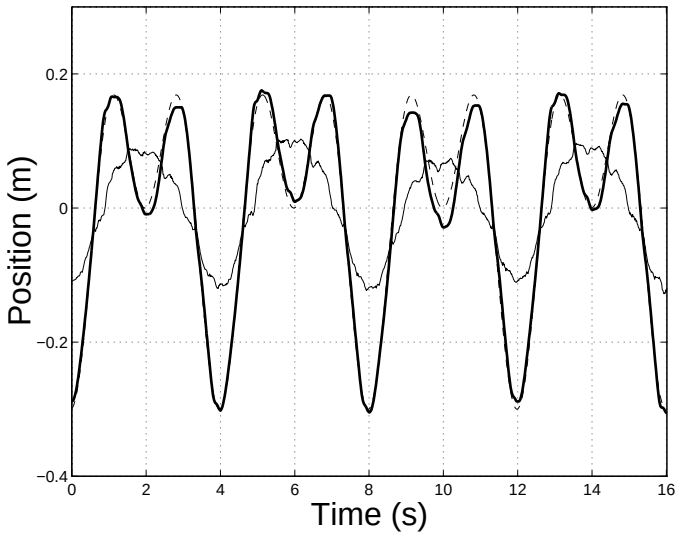



**Figure 6.38**: *Response of the feed-forward FOX inverted pendulum system to a non-constant reference, after 120 training cycles. Key: cart position $p_c$ (—), pole position $p_p$ (—), reference (– – –).*

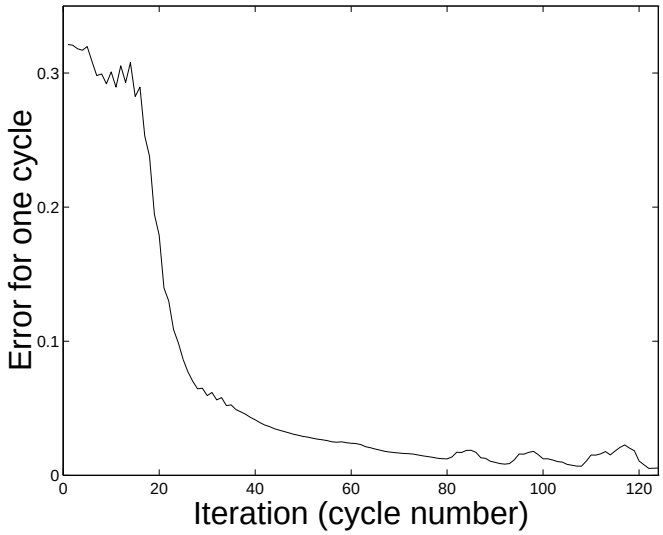



**Figure 6.39**: *Error performance during training for the feed-forward FOX inverted pendulum system.*

much less information about the system being controlled (only a binary reward/penalty signal is provided). In contrast the FOX controller has expert control knowledge split between the internal feedback controller and the continuous error signal.

The inverted pendulum controller was very easy to design. The most time consuming part was selecting an appropriate error function and learning rate by trial and error. Both experiments resulted in successful control being learned in a reasonable time. The control fidelity is very good considering the many mechanical deficiencies of the system. This system is competitive with the best currently available inverted pendulum learning controllers.

## 6.7 Testing: Mobile robot

A simple two wheeled mobile robot was constructed (see figure 6.40). The robot's task is to follow a thin guide cable laid out on the floor. The cable carries a 12kHz high-current signal. The robot has four pickup coils mounted in front of it to pick up the magnetic field from the cable. The position of the cable between each pair of sensors is inferred from the relative field strengths. The robot has two small DC motors for the left and right wheels. With the saturating motor driver electronics and sticky gearboxes these are two highly nonlinear actuators.

A schematic of the mobile robot experimental system is shown in figure 6.41. The 12kHz sensor signals are filtered by hardware on the robot to produce steady voltages that correspond to the field strengths present at each sensor, then they are digitized and passed to the software controller. The software controls two motor driver circuits which feed the left and right wheel motors.

A block diagram of the controller algorithm is shown in figure 6.42. The controller output is interpreted as a "turning" signal which is added and subtracted from a constant speed reference to get the motor driver signals. The raw sensor signals are processed by digital filters and two values are produced: an estimate of the cable position between the front pair of sensors, and similarly for the rear pair. The zero position for both sensors occurs when the cable is centered under the robot. The rear sensor position

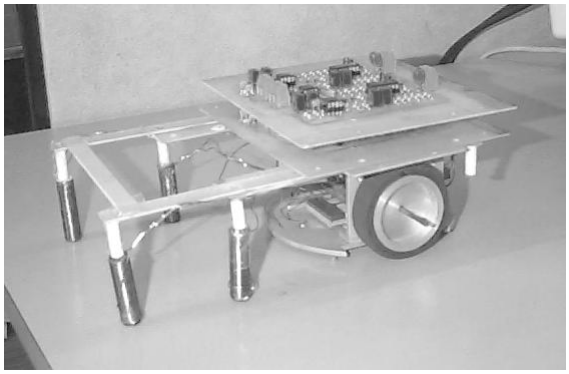


**Figure 6.40**: *The mobile robot. The four vertical poles mounted in front of the robot are sensor coils to pick up the magnetic field of the guide cable.*

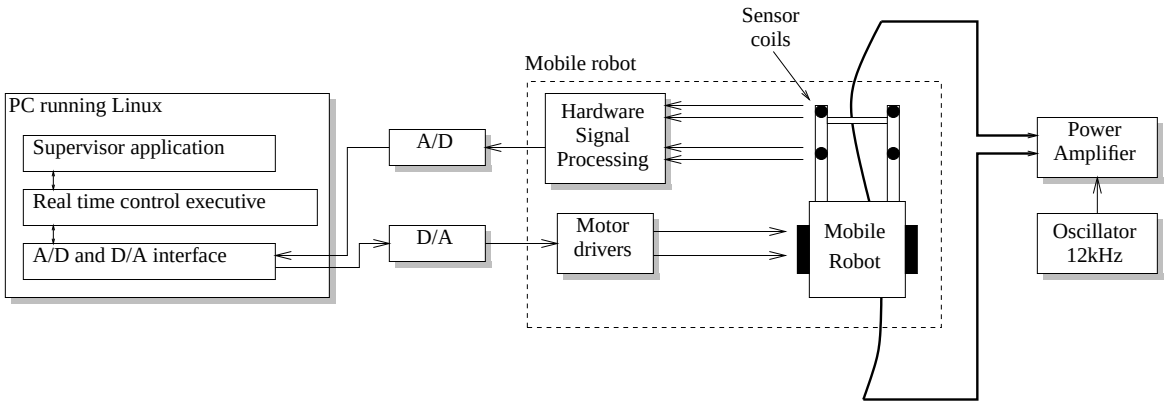



**Figure 6.41**: *A schematic of the mobile robot experimental system.*

is used as the error signal. When this is minimized the robot should be tracking the cable accurately.

The rear sensor position (and its derivative) are used in the internal linear feedback loop. The front sensor information is deliberately excluded, even though it can be usefully used in the linear controller. The reason for this is that FOX is expected to be able to *anticipate* the correct turning commands from the front sensor information, because the front sensor naturally sees bends in the cable before the rear sensor. The standard error function was used. The rest of the controller is configured similarly to previous examples.

With just the linear controller operating the robot has barely acceptable performance. Two typical robot trajectories are shown in figure 6.43. The robot can mostly keep itself on the track, but it overshoots every corner. About half the time it fails to take the sharp turn shown in figure 6.43b. Note that there is a magnetic anomaly present at the cable junction (where the two ends of the cable come together and lead off to the power amplifier) which causes brief sensor glitches.

The "impulse response" of the robot was measured in an extremely ad-hoc way: the robot was set to run up the center of a straight section of cable. Part way along it was nudged out of alignment and the error was measured as it sought the center again. The result is shown in figure 6.44a, along with the second order eligibility profile that was (arbitrarily) chosen to approximate it.

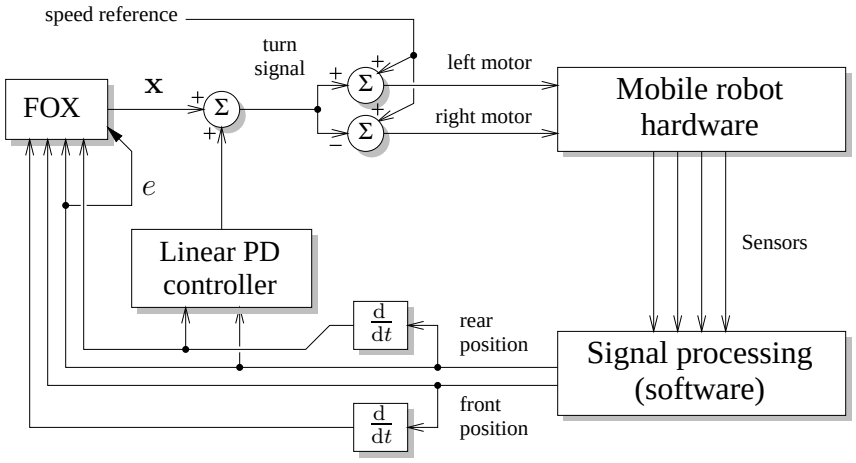
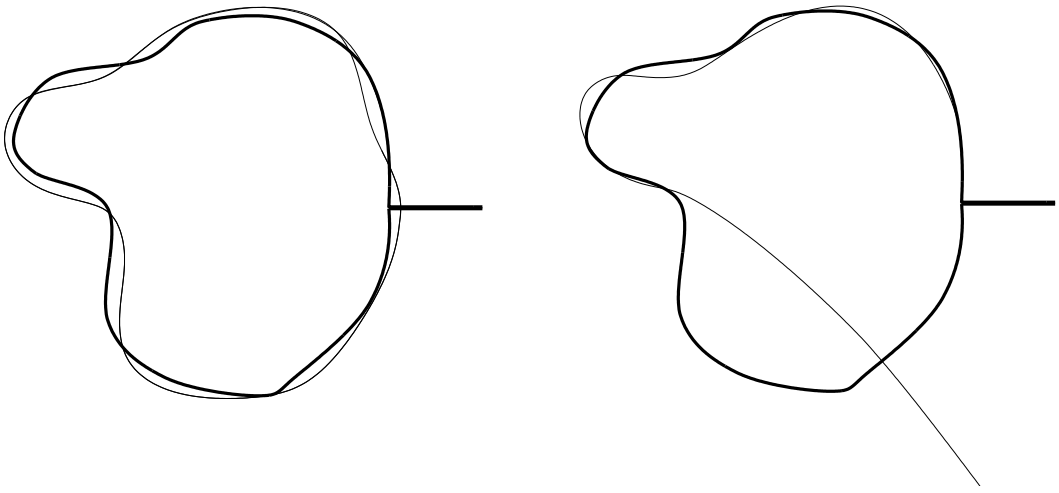
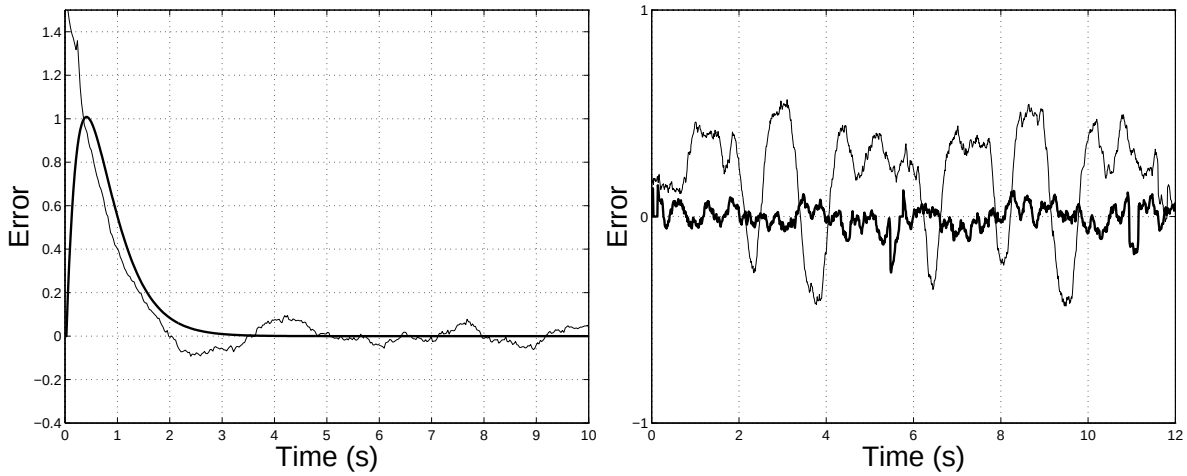



**Figure 6.42**: *A block diagram of the mobile robot controller algorithm.*



(a) (b)

**Figure 6.43**: *Mobile robot experiment: Two typical robot trajectories before training. The thick line is the position of the guide cable, the stalk at the right is where the two ends of the cable come together and lead off to the power amplifier. The thin line is the position of the robot (partially inferred from error data, and smoothed). The robot starts at the cable junction and travels anti-clockwise around the cable. In (a) the robot manages to stay on the track, in (b) it gets confused at a sharp turn and leaves the track. The robot is traveling anti-clockwise in both cases.*




(a) (b)

**Figure 6.44**: *Mobile robot experiment.* ***(a)*** *The (very) approximate impulse response of the mobile robot (—), and the eligibility profile chosen to match it (—), with $k_a = 0.44$ and $k_b = 1.33$.* ***(b)*** *The error trajectory before (—) and after (—) ten minutes of training using FOX.*

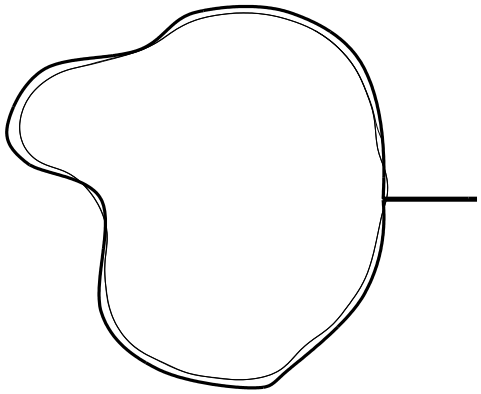


**Figure 6.45**: *Mobile robot experiment: The robot trajectory after training. The robot starts at the cable junction and travels anti-clockwise around the cable. Key: guide cable (—), robot trajectory (—).*

Figure 6.44b shows the results that are achieved after ten minutes of training with FOX. The error signal was greatly reduced in magnitude. Note that when the robot left the track during training the system was stopped and the robot was repositioned before continuing. This was done to prevent anomalous data from interfering with training. Figure 6.45 shows the robot trajectory after training. The track overshoots are reduced, and the robot never leaves the track. FOX successfully learnt to anticipate turning commands from the front sensor values.

FOX can only successfully operate in those areas of the workspace in which it was trained. This was illustrated by making the robot travel in a clockwise direction after it had been training in an anti-clockwise direction. The robot performance was instantly degraded, especially at the sharpest turn in the track.

## 6.8 Conclusions

It has been shown that FOX-based controllers are *easy* to design. Some control-systems experience is necessary to design the internal feedback controller (IFC), but this is a relatively easy task because the IFC only needs "acceptable" performance, so little time is taken up with parameter tuning. Once this is done, FOX requires only easy-to-measure dynamical information about the resulting system, i.e. the error impulse response. An eligibility profile model must be created from the impulse response, but a large amount of approximation is acceptable. Thus the eligibility profile can be designed "by eye" in many cases, so lower order models can be used (which are computationally smaller and faster).

The concept of CMAC weight eligibility was important for the success of the experiments described here. In all cases a zero-order FOX, equivalent to a feedback-error controller using a straight CMAC, did not work.

The selection of the CMAC inputs depends on whether the FOX is being used in a feedback or feedforward mode, and on the nature of the reference signal. The selection of the other CMAC parameters (for example $n_a$ and $n_w$) is largely a matter of engineering experience, as each selection involves a variety of tradeoffs. As shown in the inverted pendulum example, the weight table occupancy can be used as a rough guide to the selection of $n_w$.

FOX is not just limited to linear systems. It can be applied to any nonlinear system where the concept

of the impulse response is sufficiently well defined. That is, the system's behavior when presented with an impulse should be sufficiently consistent over its useful state space, so that the eligibility profile always remains a reasonable approximation of the system's dynamics. Note that this depends on the fact that FOX can handle large eligibility profile / impulse response deviations. FOX will even work with non-decaying impulse responses.

The principle design decision is the selection of the error function. Different functions result in different system behavior, which is not always easy to predict. The most time consuming aspect of the design is trying various error functions and learning rates to see which is most effective. The learning rates must be selected to prevent over-training, while at the same time providing a quick enough convergence rate. The overshoot error function appears to be the most universally useful.