Comparing Movement Learning in Man and Machine: Final Report

Tom Proctor (Q)

Fourth-year undergraduate project
in Group F, 2013/2014

"I hereby declare that, except where specifically indicated, the work submitted herein is my own original work."

Signed: Date:

Acknowledgements

I wish to thank the following people for their help throughout the course of this project:

Carl Rasmussen for his excellent supervision and insight on all aspects of the project. David Franklin for his expert guidance on human learning. Andrew McHutchon for his generous help with modifications in the PILCO environment.

Technical Abstract

Comparing Movement Learning in Man and Machine

Tom Proctor (Q)

This project compared movement learning in man and machine using simple reaching movements with the arm in a horizontal plane. There was already a wealth of academic literature on human learning experiments of this nature, and so the main technical requirement of this project was to set up the same problem in a machine learning environment.

In general, human learning must be relatively data efficient, as we cannot take hundreds of attempts to learn each new movement or task. Human control is also characterised by numerous difficulties, including delays, redundancy, uncertainty, noise and nonlinearity.

The machine learning environment concerned is PILCO, a reinforcement learning algorithm which builds a probabilistic dynamics model of the system with no prior knowledge. It exhibits unprecedented learning efficiency on challenging and high-dimensional control tasks, and thus is well suited to a human learning scenario.

A 2-joint, 6-muscle model of the arm was implemented in MATLAB in order for PILCO to run simulations of the control task, which was to perform a planar reaching movement, from just in front of the chest to almost full stretch directly in front of the starting point. The 2 joints are the shoulder and elbow, and the 6 muscles are: shoulder flexor, shoulder extensor, elbow flexor, elbow extensor, biarticular flexor and biarticular extensor. The state of the system at any point in time is described by 4 state variables: shoulder angle, elbow angle, shoulder angular velocity and elbow angular velocity. Initially, the model was implemented without any external forces applied, or stiffness or damping properties, and the controller was a linear controller which calculated each of the 6 control inputs as a weighted sum of state observations.

PILCO learns this task well for relatively low levels of observation noise (standard deviation $\approx 2^\circ$). It copes with higher noise levels, but performance is slightly worse.

In this initial control problem, the controller already has to deal with redundancy in the movement. The project then looked to implement various other features and difficulties from the human learning problem in order to observe how the strategy for performing the task changed.

First, the concept of feedforward control was analysed. Feedforward control corresponds

to learned dynamics and its effects have been shown in several human learning experiments. Theoretically, if the forward dynamics model is perfect, an entirely open-loop (feedforward) controller should be implementable which will achieve the task perfectly. In reality, there is a trade-off between feedback and feedforward control. The better the dynamics model, and the higher the noise, the more the controller will rely on feedforwad control.

Next, signal-dependent actuation noise was added to the muscle forces. This imposed a penalty on uncertainty in the forward dynamics model for high levels of activation of the muscles. Similarly to human trials, in which movements can be described as minimising variance in the action, it was seen that PILCO used lower levels of muscle activation but in a very similar pattern. The controller is therefore performing the same action, but marginally slower and with lower uncertainty.

Varying lengths of delay up to 95ms were added to the state feedback. This was to mimic the delay in sensory perception caused by the slow propagation of electrical signals through neurons. This destabilises the task, and in response, the controller has to lower its feedback gains in order to maintain stability.

Finally, the project looked at two important material properties of muscles, stiffness and damping. Both of these properties resist motion, and changing the force exerted by each muscle changes its level of stiffness and damping. As a result, unstable systems can be stabilised by co-contracting pairs of muscles (without a net change in joint torque) and therefore increasing stiffness and damping.

A viable implementation of stiffness could not be found within the PILCO framework, and so simulations were only run with damping added to the dynamics model. However, the results were encouraging, with the controller increasing muscle activations in order to stabilise the given unstable task. The controller then, is able to achieve a task that is not possible without damping. However, rather than increasing damping only in the direction of instability, damping is increased in all directions. This result is likely due to missing features in the simulation, such as an incentive to minimise the muscle activations.

Overall, the project found many parallels between human learning and PILCO. Much of the control theory used to make predictions in response to changes in the system apply equally to both systems, and thus similar behaviours can be extracted.

Contents

1 Introduction and Motivation

1.1 The project brief

"Humans and recent machine learning algorithms are capable of rapid learning adaptation. In this project, the two learning systems will be compared qualitatively and quantitatively, using simple arm motion learning. Human data will be acquired in the lab and compared to learning simulations using machine learning algorithms."

1.2 Motivation

Human and machine learning have developed very separately. Typically, machine learning algorithms have to model on a large (possibly $\mathcal{O}(10^6)$) training set before they can exhibit high accuracy on unseen data. Conversely, human learning is more data limited and learning must take place with very limited experience of the environment.

PILCO (Probabilistic Inference for Learning Control) acts as something of a link between the two counterparts, as it shares similarities with both the artificial and biological learning systems. It is a reinforcement learning algorithm that builds a probabilistic dynamics model. The algorithm is capable of 'unprecedented learning efficiency on challenging and high-dimensional control tasks' [1].

Although PILCO was not designed to mimic human learning, its similarities are interesting. How deep is the level of similarity? What are the key differences in the control methods when the two systems are trained on the same (or as similar as possible) task? Is it possible to infer anything about the optimality of human learning from these results? This project aims to address questions such as these, by setting up experiments in which human and machine learning can be directly compared.

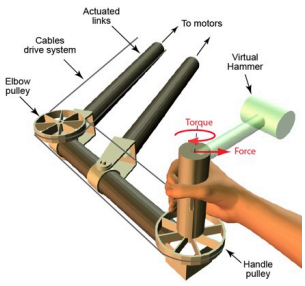
2 Background

2.1 Human Learning

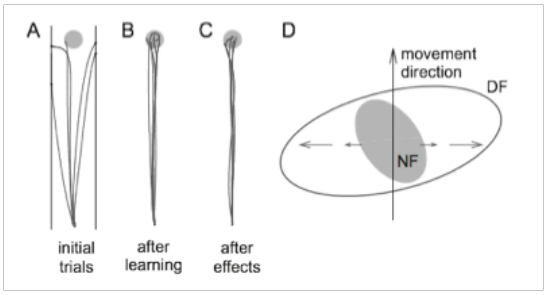
Human arm motion learning is a highly complex problem that has to cope with a long list of properties such as: non-linearity, delays, redundancy, uncertainty and noise [2]. Each of these properties presents a major challenge to the human nervous system and impacts the decision making process when controlling muscles. Therefore it is obvious that their effects must be considered in relation to more conventional control theory problems, when comparing human and machine learning.

Delays are present throughout the nervous system, typically of the order of magnitude of 100ms [2]. This means that sensory information received by the brain is out of date, and control commands take an appreciable amount of time to produce force in the muscles. Noise is also present throughout the nervous system, affecting measurements of the current state, and commands issued to muscles. This means that the control signal is unlikely to be performed accurately, and one property of this noise is that it increases with muscle force [4]. This property is responsible for some of the characteristics of human movement, which can be described as minimising uncertainty or maximising accuracy, subject to boundary constraints [2][5].

Uncertainty comes from neural noise and delays, but also from sensory limitations and uncertainty about our model of the environment. This uncertainty is analogous to the probabilistic dynamics model of PILCO, discussed in section 2.2, in which the distribution over possible dynamics models represents its uncertainty in its model of the environment.



[6]



[4]

(a) Robotic arm (b) Hand path errors and stiffness ellipses

Figure 1: 1a The robotic arm used for human learning experiments. 1b A, B: hand paths before and after learning. C: hand paths after learning and in absence of field. D: Stiffness ellipse of end-point of the arm after learning

Figure 1 shows briefly the experimental setup and results for human learning experiments studying simple arm motion. The motion is restricted to a 2D plane, and the task is a simple reaching movement, which may be performed in a null field (NF) or a novel force field (applied by the robotic arm) such as a curl force field to ensure learning must take place to complete the task.

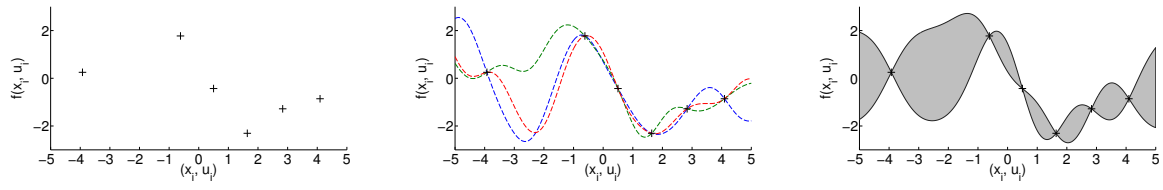
The results shown in Figure 1b are for learning the reaching movement in a Divergent Force field (DF), which is an unstable field where the force outwards becomes greater as the hand deviates further from the centre line. As expected, the hand paths are more accurate after training, though interestingly they are also more accurate once the DF is removed. This indicates that the muscles are not counteracting a specific force, but simply stiffening in order to overcome the instability. This is shown in part D of the

figure, where the stiffness ellipse of the end point of the arm is shown after training in the NF and DF. The stiffness has only increased in the direction of instability.

2.2 Machine Learning

PILCO is a reinforcement learning algorithm and learns a probabilistic dynamics model and control policy with no prior knowledge of the system. The fact that it learns a probabilistic, rather than deterministic, dynamics model is important for avoiding model bias. This concept is illustrated in Figure 2. The middle figure illustrates some of the possible smooth functions that could have generated the noiseless observations in the left figure. However, these are deterministic functions, and so if the observations diverge from the training data, the model will still make predictions with certainty but they will have no relation to the current state.

Instead, PILCO's dynamics model is more akin to the diagram on the right, which is a posterior over the transition function with uncertainty incorporated. This means at the beginning of training, uncertainty is very high and the model does not make very specific predictions, but this is the correct response to not having much information about the system.



[1]

Figure 2: Left figure displays some noiseless observations. Middle figure draws some of the possible smooth underlying functions. Right figure shows PILCO's probabilistic dynamics model based on these observations. A wide grey area indicates large uncertainty.

Figure 3 illustrates the basic operation of PILCO's internal simulations. Given the true state, x, noisy measurements, y, are observed and represented as a distribution with a mean expected state and an uncertainty. This observation noise is one source of uncertainty, and is explicitly estimated. The observed distribution of states y and the current probabilistic dynamics model are then combined to predict future states in response to control inputs u. As this is a non-linear mapping from a distribution of states, the new state is approximated by a Gaussian distribution with matched moments. Using these forward predictions, a control policy is trained which minimises some long-term cost function. The trained control policy is then applied to the real system in order to gather more experience. The measurements from this real trial are then used to update the proba-

bilistic dynamics model, which is then used to train a new control policy.

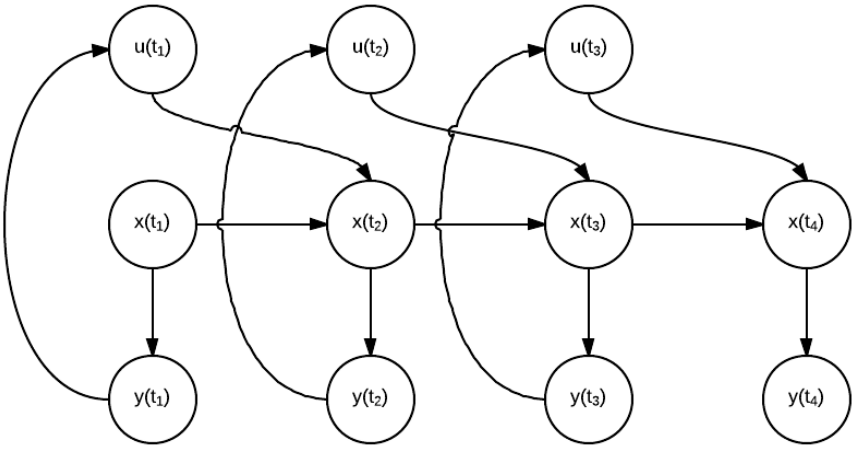


Figure 3: The simulation steps of PILCO

3 Experimental Setup

This project aimed to study the learning of arm movements. In order to limit the size of the problem, and make the results as directly comparable to existing human learning experiments as possible, the task was specified as a planar movement from a comfortable position in front of the chest to a target directly in front of the starting position. There are both human learning [7][8] and algorithmic learning [9] papers that look at learning this movement under various conditions.

PILCO is a form of predictive control, and as such requires several ingredients specific to each control problem. The first is a predictive dynamics model, which will predict future states given a current state and some sequence of future control inputs. In PILCO, the dynamics model and controller are trained on any given system with no prior knowledge, and so these elements do not require any modification when the control task is changed. Both the future states and future inputs are predicted stochastic distributions with a mean and variance.

Another element is the cost function, which defines the desirability of any possible state. Given the aim of PILCO to learn the dynamics and controller for a system with no prior knowledge, the cost function should support this philosophy and be as simple and intuitive as possible. This is to ensure it does not imbue the algorithm with knowledge that requires some previous insight into the problem. For example, the simplest way to

specify the cost function for a reaching movement is to define a cost function that is only a function of location, not velocity, i.e. the minimum loss is at the location of the target, irrespective of velocity.

In this case, to minimise the loss over the whole optimisation horizon, it would be desirable to both be at the target location and have zero velocity, but the controller should be able to generate a strategy that achieves this with a simple and very general spatial loss function. Thus the loss function is specified as follows, where W is a tunable parameter, $\mathbf{x}$ is the current state, and $\mathbf{z}$ is the target state.

$$\mathcal{L} = 1 - e^{-\frac{1}{2}(\mathbf{x}-\mathbf{z})^T W (\mathbf{x}-\mathbf{z})} \tag{1}$$

$$W = \texttt{diag([2 2 0 0])}$$

This loss function is similar to a quadratic function near the target state, but then saturates to a cost of 1 far away from the target. This is advantageous, as a quadratic cost function has no upper limit, and continues to penalise very far away states with increasing strength. This is not a good specification of the task, as we are primarily interested in completing the task, not failing in the least catastrophic manner. The quadratic cost function would strongly favour the closer state, even though it is still not performing very well, whereas the saturated cost function will only significantly differentiate between states that are meaningfully close to the target state (where 'meaningfully close' is defined by the weightings in W, i.e. width of the cost function). One potential drawback of this form of cost function is that if the random trials never get anywhere near the target state, the controller may never discover the region of low cost and get stuck in areas where the gradient of the cost function is almost flat (as gradient descent is used for optimisation). This is an important factor to consider when setting the weights in W, but the weights must also be high enough that the cost is only low when the intended task is achieved.

Each element of PILCO described so far was a pre-existing part of the PILCO environment, and thus required little or no work to set up. However, the final major element to be considered is the true dynamical behaviour in response to control inputs. This is one significant part of the technical work that had to be done for this project. Separate to the algorithm's own internal simulations, we also want to observe the true response of the system to a set of controls. These observations may be taken by measurement of an experimental rig, or simply a simulation which solves an ordinary differential equation based on the mechanical equations of motion. The simulation implemented for this project is discussed in section 3.1.

3.1 Model of the arm

In order to apply the PILCO algorithm to a model of the human arm, it was required to set up an appropriate simulation of the dynamics of the arm. While the algorithm has previously been applied to physical systems such as an inverted pendulum [1][10], this project was the first time it had been applied to any human control problems, and so it was more sensible to run the experiments as simulations, which are quicker and cheaper to set up and modify.

Several existing academic papers [9][11] examine a planar 2-joint, 6-muscle model of the arm for learning and control problems similar to the reaching movement this project is interested in. Fortunately, the MATLAB implementation of the model in Franklin's paper [9] was made available by David Franklin, and thus the physical parameters and dynamical equations were adapted directly from this source code.

The model state is described by shoulder angle (q_s), elbow angle (q_e) and the respective angular velocities($\dot{q}_s$ and $\dot{q}_e$). The 6 muscles acting on the arm in a horizontal plane are in three flexor/extensor pairs: for the shoulder, elbow and a pair of biarticular muscles acting between the two joints.

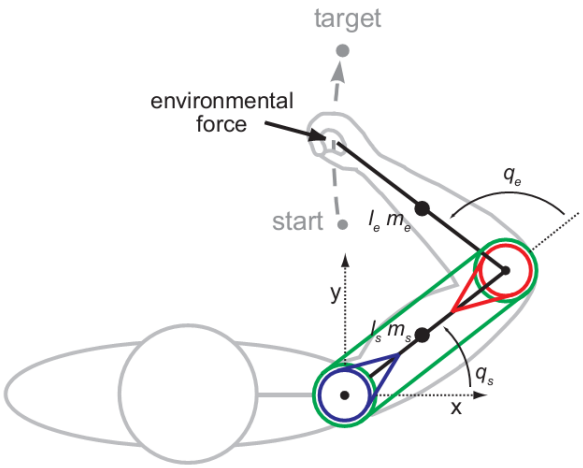


Figure 4: Diagram illustrating the geometry of the human arm model [9]. External forces will be zero until section 4.7.

Further details of the model of the arm can be found in the supplementary material for [9].

3.2 Considerations for applying PILCO to the model

In addition to setting up the mechanics of the simulation, there were also several factors to consider in order to run the simulations within the PILCO environment. These con-

siderations constituted the second significant area of work required to set up the initial experiments discussed in section 4.2.

One key difference between robotic control and muscle control, is the method for implementing opposite control signals. For a motor, it is simply run in reverse, and the control signal operates in the opposite direction. Muscles, however, can only exert force in tension, and opposite controls are implemented using muscle pairs. This was addressed in the PILCO environment by introducing an offset to the trained control signal. Although PILCO normally operates on motor actuators, and therefore uses control signals which can be equal magnitudes in opposite polarities, the magnitude of the maximum muscle force was set to half the desired maximum, 20N. Then, in the dynamics function, the received control signal, which is in the range [-20, 20], was offset by 20 to be in the range [0, 40]. This simulates actuators (muscles) that will only exert a force in one direction (tension). This maximum muscle force was another design choice, which was chosen to be fairly low in order to keep movements at a sensible speed, particularly on random trials.

It was also required to specify both the length of the control horizon and the length of time steps in simulations. A relatively long 3 second control horizon was chosen, as this is plenty of time to perform the movement and also incentivises being able to hold the hand at the target at the end of the movement. In order to minimise the computation time for simulations, a relatively coarse time step of 0.1s was chosen. Although the mechanics of the arm are simulated at a high enough speed to satisfy a stringent gradient smoothness constraint (and thus run with a much finer time step than 0.1s), the controller is only optimised for each time step in the control horizon. The control between time steps is then determined by the controller type, which in the case of this project, is a zero-order-hold controller. Thus, the control signal calculated for a time step is held continuously until the next time step.

Observation noise is another property that was important to all simulations, and had to be set up for the scenario. Observation noise determines the accuracy with which the state variables can be measured, and throughout this report is referred to by the standard deviation of angle measurements. For example, a simulation in which the observation noise level is described as 0.03, the measurements of the true angles q_s and q_e will be Gaussian distributed random variables with means q_s and q_e respectively, and standard deviation of 0.03 radians ($\approx 1.72°$). The observation noise for the angular velocities will then have standard deviation $\frac{0.03}{dt} = 0.3$, where dt is the sampling time, 0.1s. This is an attempt to ensure that observation noise across all the dimensionally inconsistent state variables contributes approximately uniform levels of noise to the overall system. Observation noise will be set to a standard deviation of 0.03 unless otherwise stated, as this was empirically

found to be a relatively low level of noise (such that the controller can easily achieve the task) for the scenario.

Some further experimental setup considerations are explained in section 4.2, such as the number of initial random trials and whether to implement physical model constraints.

4 Results and discussion

4.1 Interpreting the results

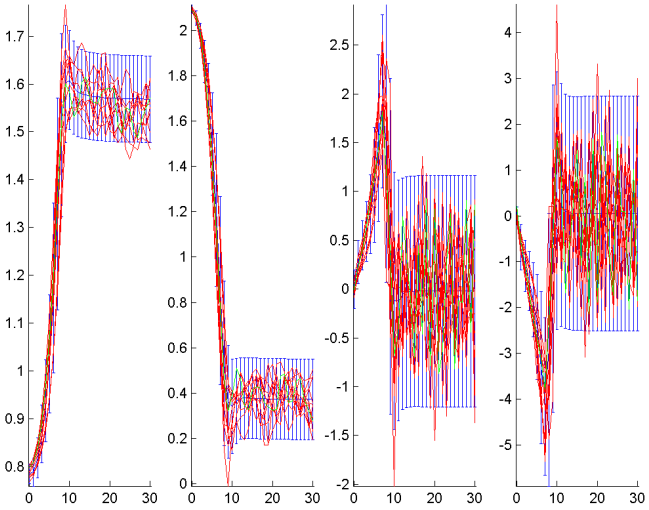
There are three standard graphs that will be used throughout this section to illustrate the results of various simulations. In all three graphs, the red lines show the results of multiple rollouts (to confirm the dynamics are stable), and the green line represents the only trial which the PILCO algorithm observes.

The first type of graph is the progression of state variables over the control horizon, as illustrated in Figure 5a. In order, the four sub-plots are shoulder angle, q_s, elbow angle, q_e, shoulder angular velocity, $\dot{q}_s$, and elbow angular velocity, $\dot{q}_e$. These angles are as illustrated in Figure 4. The x axis is the simulation step, going from 1 to 30 in all cases. The time step for all trials is 0.1s, and so all trials are over the course of 3s. The y axis is the angle and angular velocity, in rad and $rad\,s^{-1}$ respectively. The cost function specifies that the target is at $q_s = 1.5708$ and $q_e = 0.3713$, and thus in this example the task is being achieved.

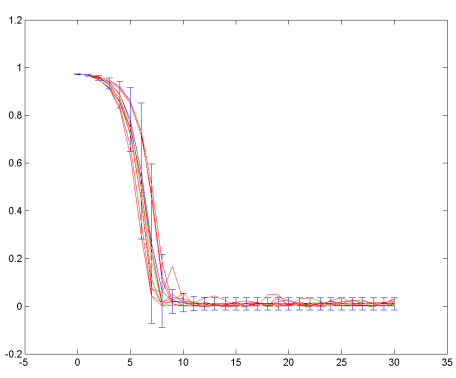
The second standard graph shows the loss function over the control horizon (Figure 5b). Again, the x axis is simulation step, and the y axis is cost. Any successful controller should result in a line which drops to near-zero and stays there for the rest of the control horizon.

For both of these graphs, in addition to the real rollouts in red and green, the predicted mean and error bars are shown in blue. The internal dynamics model is accurate if the real rollouts follow the predicted mean closely and the error bars are small. Both of these graphs are also standard outputs when running the PILCO environment.

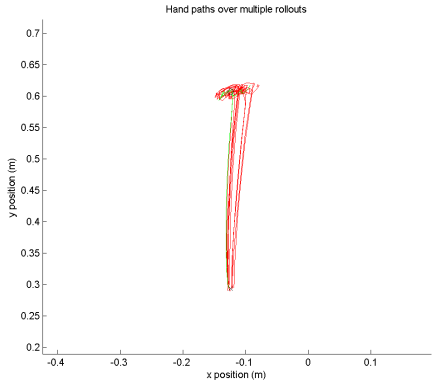
The third standard graph that will be used repeatedly is the Cartesian space hand-paths of the rollouts for a particular trial, as shown in Figure 5c. The x and y axes represent the position of the hand in metres. The shoulder is at the origin, $(0, 0)$, 'x' marks the mean of the hand's starting location, and 'o' marks the target location. In this example, the hand is moving almost directly to the target, and then maintaining its position there.



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.



(c) Hand-paths.

Figure 5: Example graphs for illustration of how results will be presented. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

One final note on graphs concerns plots of muscle activations and feedforward weights over the control horizon. The controller used is zero-order hold, and so the control calculated at the beginning of a time step is held at that value until the beginning of the next time step. However, drawing this accurately results in a great deal of overlap in the plots both horizontally and vertically, making the plots very difficult to interpret. As a result, these graphs are drawn as simple line plots instead, which gives an idea of the overall shape while maintaining readability. Both these graphs, and the hand-path plots, are specific to this arm movement scenario, and were thus part of the technical work carried out for the project.

4.2 Initial experiments

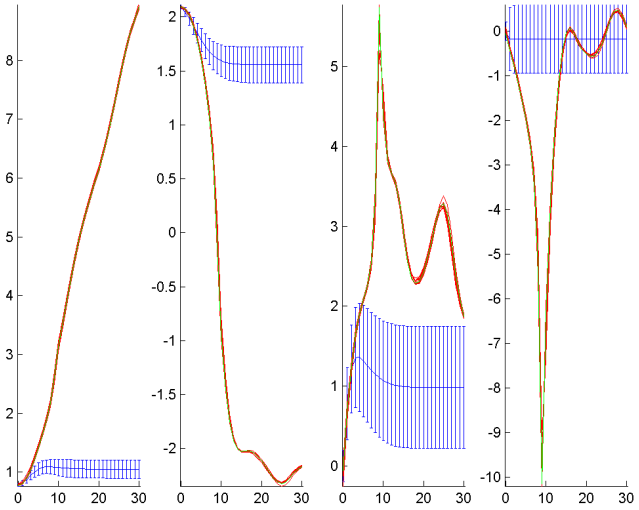
The first simulation to be run was as simple as possible. The model was as described in section 3, and had to perform the reaching task specified with purely feedback control and no external forces applied.

Figure 6 shows the rollouts for the controller having been trained on the observations from five rollouts in which random control inputs were applied. This is the usual process at the beginning of the algorithm, as it has no prior knowledge and so must gather some observations of the system before it can optimise any policy. Five random rollouts is a relatively high number compared to other scenarios in which PILCO has been applied, but this was specified in response to the high complexity of the system, and therefore higher level of experience required before a controller can be reasonably optimised.

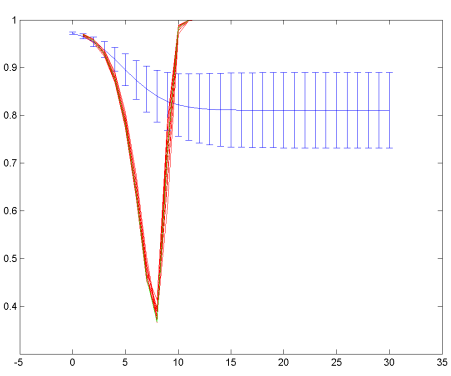
In this trial, the simulation is highly non-physical, as the shoulder has rotated nearly a full 360°. However, it was decided early on in the project to ignore this concern. The alternative would be to impose a strict constraint on the allowable angles of the shoulder and elbow, but the effect on PILCO is to cut off the simulation at the point the constraint gets violated and take no more observations after. In a system where it only requires a small deviation from the starting state to violate a constraint, this method can make it difficult to build up enough experience of the dynamics to significantly improve the controller. Furthermore, it is clear that any good controller for this problem will not violate the angle constraints of the human arm , as any reasonably efficient path will move relatively directly to the target. Thus, it was decided to leave the system without constraint in this case, under the assumption that upon learning a better dynamics model of the system, the controller would achieve the task in a sensible fashion that would naturally stay within the real constraints.

Indeed, this is seen to be a good assumption after just five controlled simulations, the results of which are shown in Figure 7. Figure 7c clearly shows a feasible path for the hand (it is not violating any of the physical constraints of the arm), and that the task is being achieved, though not particularly quickly or efficiently. Additionally, the internal dynamics model is not yet very accurate, as shown by all of the rollouts falling outside of the predicted error bars for shoulder angle.

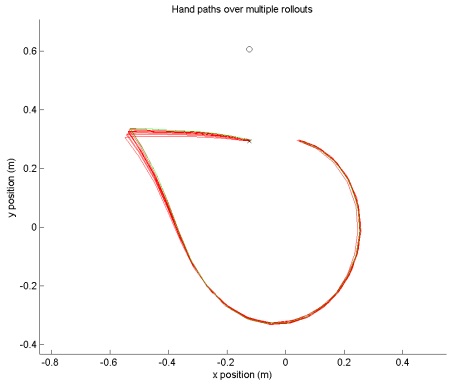
By the eighth controlled trial in Figure 8, the path is much more efficient, and as a result the cost function (Figure 8b) drops to a near-zero value slightly sooner. By definition then, the task is being performed better, and this can be confirmed by visual inspection of the state variables and hand-paths in Figures 8a and 8c. For the first time, the con-



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.



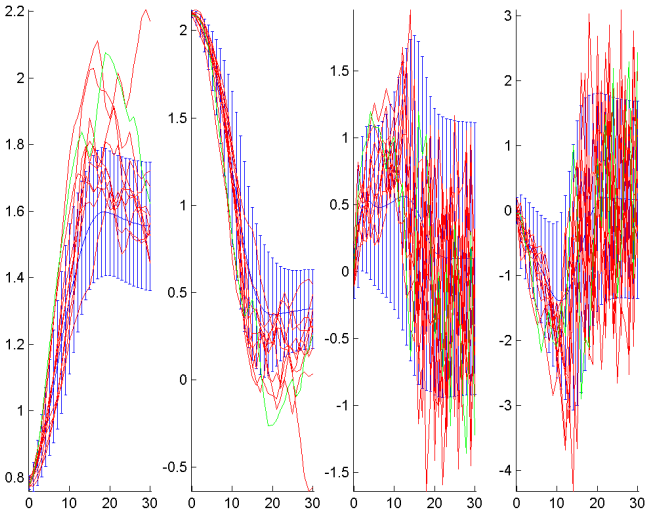
(c) Hand-paths.

Figure 6: Performance on 1st trial. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

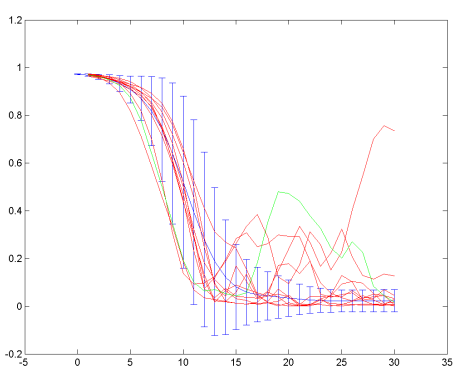
troller has performed the task quickly enough such that on reaching the target, it has to hold the target position until the end of the control horizon. It performs this reasonably well.

The internal dynamics model is significantly better, as Figure 8a shows each state variable in each rollout stays almost exclusively within the predicted error bars.

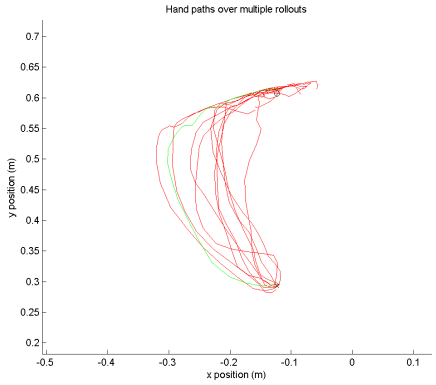
Figure 9 shows the eighteenth controlled trial, in which the task is performed very well. The cost function in Figure 9b drops to near-zero even sooner than in trial 9, and the hand-paths go almost directly to the target in relatively straight lines (Figure 9c). This intuitively seems likely to be the optimal path. The dynamics model, as illustrated in Figure 9a is clearly very good at predicting future states within the space that the con-



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.



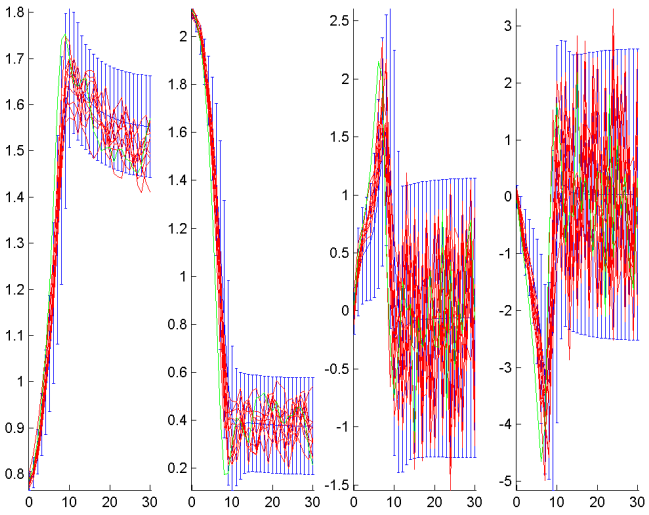
(c) Hand-paths.

Figure 7: Performance on 5th trial. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

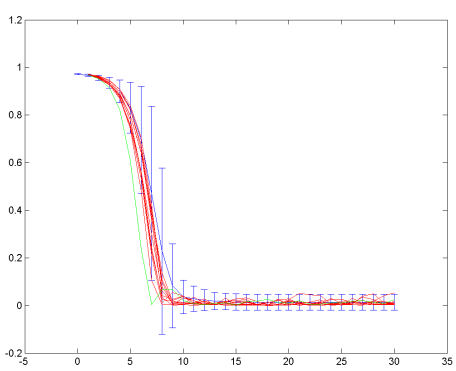
troller operates, and the Figure also best demonstrates the long period of time that the controller spends regulating the hand at near-zero velocity at the target, i.e. the movement has been achieved within the first second of the trial.

Figure 10 shows the muscle activations at the end of training for this scenario. As might be expected, the shoulder flexor and elbow extensor are activated strongly for the beginning of the reaching movement, and then all the muscles are actuated in a manner analogous to bang-bang control as the hand gets closer to the target.

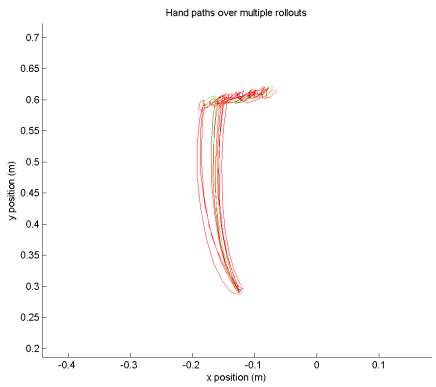
The following sections from 4.3 to 4.6 will each change just one feature from this simulation, and study its effect.



(a) Progression of 4 state variables over the control horizon.

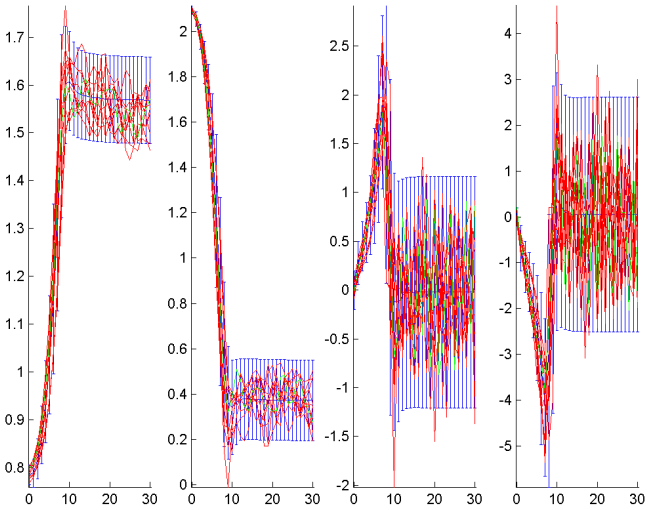


(b) Cost function.

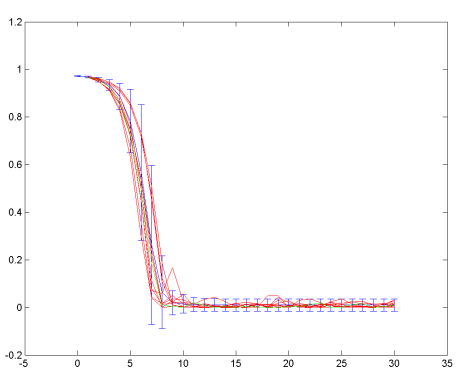


(c) Hand-paths.

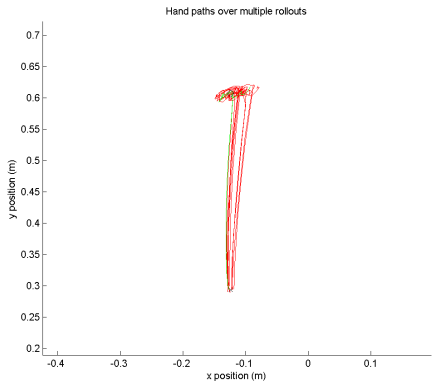
Figure 8: Performance on 8th trial. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.



(c) Hand-paths.

Figure 9: Performance on 18th trial. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

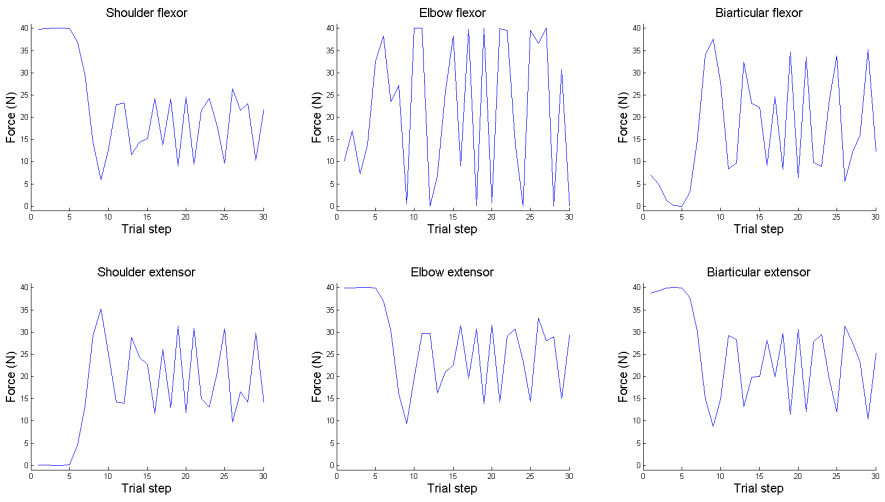


Figure 10: Muscle activations enacted to achieve the task as shown in Figure 9c.

4.3 Feedback vs. Feedforward control

Feedforward, or predictive, control corresponds to learned dynamics. Theoretically, learning the dynamics of a system perfectly should mean that a precise series of controls can be planned in advance to achieve the task. This would then be an open-loop controller, with no control inputs affected by measurements of the current state.

The literature on human learning [2] studies both feedback and predictive control. While the dynamics of a system can be learned to varying levels of accuracy, the forward model is unlikely to ever be accurate enough to train a purely feedforward controller which performs the task well. Instead, efficient feedforward controls can be learned to implement much of the required inputs, but feedback must be used to fine-tune the movement at the points where the uncertainty in the forward predictive model is too high.

This theory implies that there is a trade-off between the use of feedback and feedforward control. The more accurate the forward dynamics model is (i.e. the lower the uncertainty is), the more open loop feedforward controls can be used to perform the task. Conversely, if the dynamics model has very high uncertainty, it cannot be used at all to predict the future inputs required, and the controller will have to rely on feedback control exclusively.

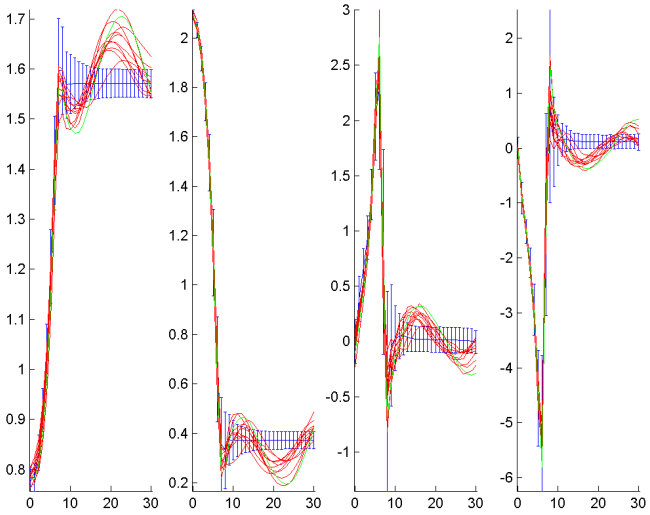
In addition to this, it would also be expected that as uncertainty in the observations increases, feedback weights will decrease and the feedforward controls will be more dominant. This is intuitive, as if the observations were so noisy as to not be reliable indicators of the current state at all, it would be impossible to implement any sensible feedback strategy based on those measurements.

In order to test these theories in a machine learning environment, it was first necessary to implement some notion of feedforward control in the policy for PILCO. The existing linear controller worked by optimising a set of weights between each of the 4 state variables and each of the control inputs (the 6 muscles) to give 24 weights. There are also 6 bias terms. The policy is then simply the linear combination of these weights multiplied by the state variables, plus a bias. The proposal was to introduce an additional set of $E \times H$ parameters, where E is the number of controls (6) and H is the number of time steps in the control horizon (30). Thus, there are a set of feedforward controls planned for each time step. At each time step, the feedforward and feedback controls are added together before being transformed into valid muscle activations by the saturation function.

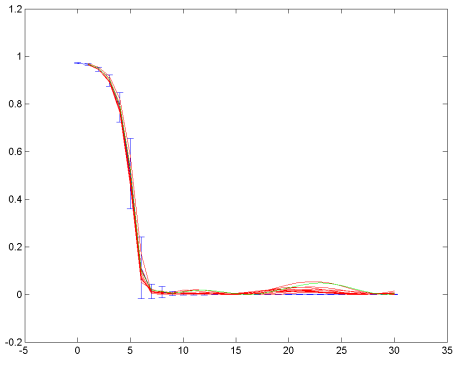
This strategy allows the controller to plan future controls based on its own prediction of future states, but it also introduces a very large number of additional parameters to

optimise. As a result, the policy was optimised for up to 10 times as many function evaluations, but the rest of the settings were otherwise unchanged from the initial experiments in section 4.2.

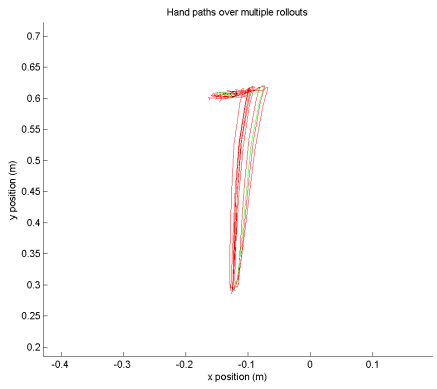
The result of running PILCO on this new system is similar to before for a task that was easily achievable before, and performance is perhaps even slightly worse, as seen in Figure 11. This is understandable as a great deal of additional redundancy has been added to the system without making the task any more difficult to achieve. Therefore, the optimisation of the control policy is even more prone to getting stuck in local optima. This effect is discussed in further detail later in this section.



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.



(c) Hand-paths.

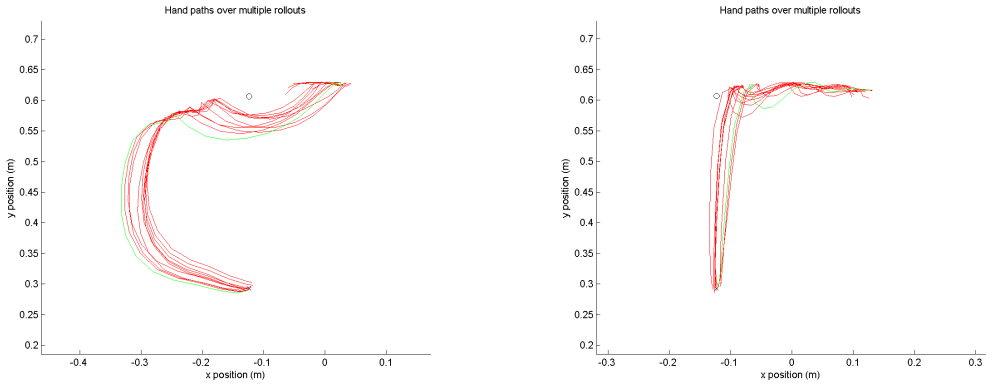
Figure 11: Performance with feedforward control and 0.03 observation noise on 20th trial. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

The performance in a variety of observation noise levels is analysed in Figure 13. The

task is considered to be achieved in each of the first 4 cases, where the total cost over all rollouts is around 50. For observation noises of 0.09 and 0.11, this total cost rises sharply to over 70. However, the hand-paths shown in Figure 12a for observation noise 0.09 show that the hand is still reaching the target, and so although the trial cannot be compared quite as directly with the simulations in lower noise, there is still a level of success as it is reaching the target in the end.

Obs. noise	0.01	0.03	0.05	0.07	0.09	0.11
Total cost	42.4	45.0	53.7	54.6	111.2	74.0

Table 1: Total cost incurred over 11 rollouts of controller (using feedback and feedforward control) at the end of training. Theoretically, total cost may range from 0 to 330, but typical performances range from 40 (excellent) to 260 (almost random).



(a) Cost function in obs noise: 0.09 (b) Cost function in obs noise: 0.11

Figure 12: Hand-paths for observation noise levels 0.09 and 0.11. Both have relatively high total costs, but Figure 12a still reaches fairly close to the target by the end of the trial.

4.3.1 Decreasing feedback weights under increased observation noise

Figure 13 shows the trained feedforward and feedback weights under increasing observation noise. As discussed, the first 4 cases are achieving the task well, and in similar fashion. Under 0.09 observation noise, the task is somewhat less successfully performed, and in the last case, the controller is very much unsuccessful.

Figure 13a shows the feedforward weights, which seem to be increasingly utilised from trials 2 to 5, with significant changes in weights occurring later in the control horizon. The feedback weights in Figure 13b shows very clearly the corresponding decrease in feedback weights for increased observation noise, where black and white correspond to the most extreme valued weights. This behaviour is exactly as predicted. Increasing the noise on

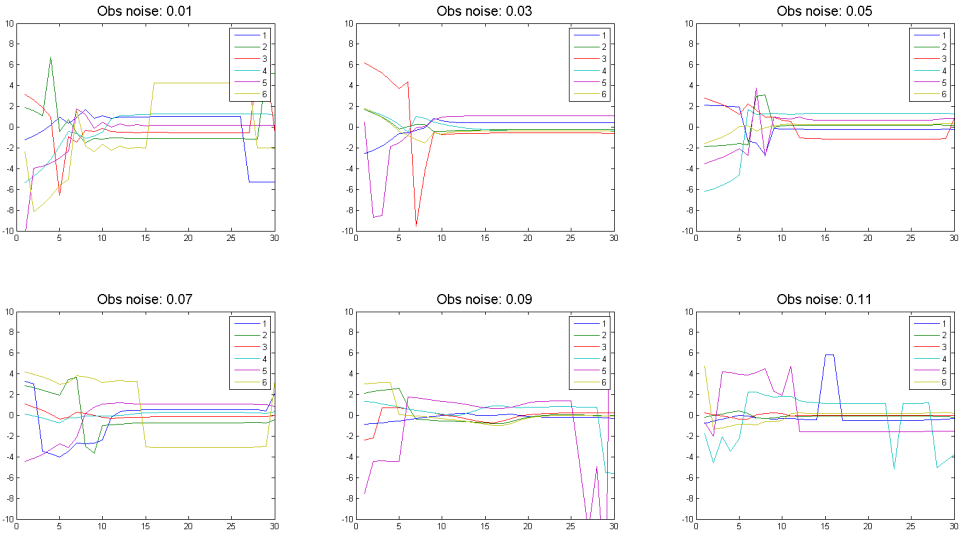
observations makes feedback less reliable, and as the controller is directly modelling the observation noise, it is able to turn down the feedback weights in response.

Despite the increased noise, the dynamics model can still be trained reasonably well by averaging over many trials. This means the open-loop feedforward weights can still be trained and they become more important as feedback becomes less reliable.

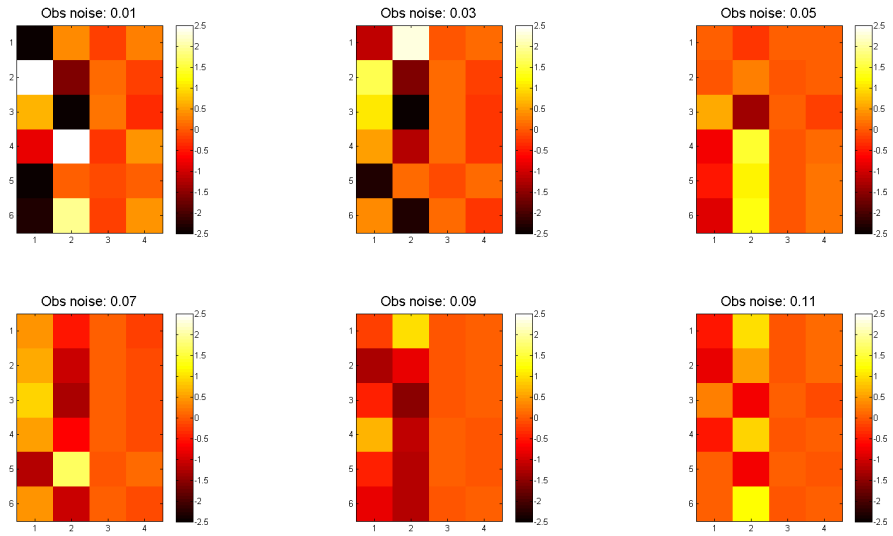
However, it is interesting to note that the dynamics model is still not good enough to successfully implement entirely open-loop control in the highest noise case. Figures 13a and 13b show that the policy for high noise has a very high reliance on feedforward weights, and relatively little dependence on feedback contributions. But Figure 12b shows the hand reaches the target, and then fails to maintain its position there. This fits with what was predicted earlier, as the movement is approximately as desired overall, but due to uncertainties in the forward dynamics model, the feedforward weights have not quite been able to implement fine level control far into the future. The feedback is too noisy to correct those small deviations in this case and the hand slides away from the target instead.

One anomaly in Figure 13a is the set of trained feedforward weights for the lowest level of observation noise. Clearly, the feedforward weights do not fit into the general trend of being increasingly used as observation noise increases. One feasible explanation for this is that the controller has become trapped in a local optimum of the cost function. Having achieved relatively good performance once with a certain set of weights, it is not possible to significantly change them without incurring a higher cost. Thus, instead of exploring the problem space fully, the optimisation process simply makes minor updates to the first 'reasonable' solution. Figure 14 shows the evolution of the feedforward weights over trials, from the first trial on which the task is almost achieved. Inspecting each line individually, it can be seen that the overall shape of the plots do not change very significantly with each optimisation. For example, there are distinct positive and negative peaks present for muscles 2 and 3 (red and green) in every trial from 3 onwards.

The conclusion must then be that the optimisation process has found a local minimum in the feedforward controls, which is not surprising given the complex and highly redundant nature of the controls. Once a solution has been found which does a reasonable job of minimising the cost function, the feedforward weights are only tuned slightly, rather than changed significantly, as any large change would result in a large increase in the cost function before it falls again. If this analysis is correct, then performing a random restart of the trial should result in very different feedforward weights, corresponding to a different local minimum, as shown in Figure 15.



(a) Feedforward weights for each muscle over the control horizon. Applied irrespective of current system state.



(b) Feedback weights from each state variable to each muscle for the linear feedback controller. Rows represent each muscle in order, from shoulder flexor to biarticular extensor. Columns represent the state variables in order, q_s, q_e, $\dot{q_s}$, $\dot{q_e}$. Colour scale is identical across all graphs.

Figure 13: Contributions from feedforward and feedback are added and then saturated to ± 1 before being scaled and shifted into the range [0,40].

This prediction is confirmed in Figure 15, where on a random restart of the same control problem, the trained weights are clearly very different to before. Additionally, we again see that the weights do not change very significantly between the first trial in which the task is almost achieved, and the last.

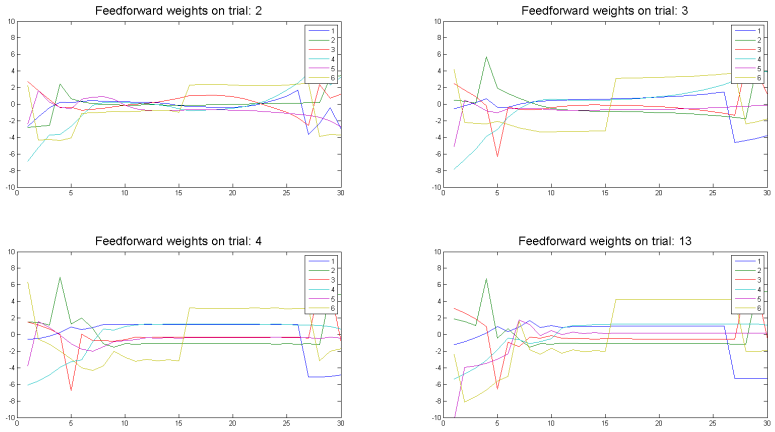


Figure 14: Feedforward weights for each of the 6 muscles at varying points in the training process. Process was terminated after 13 trials as task was achieved and little change between trials.

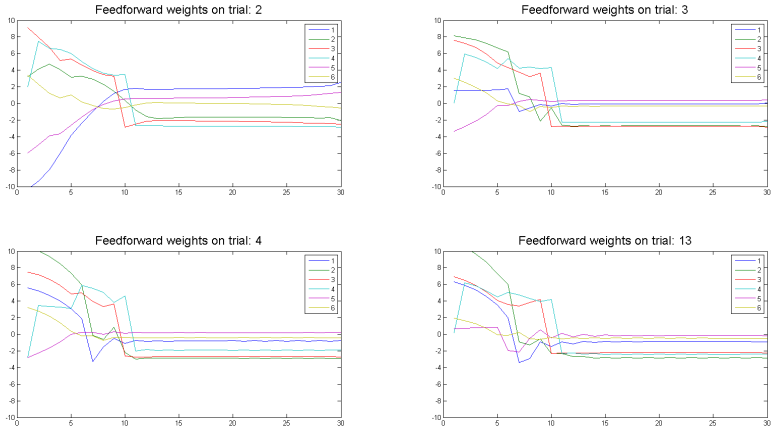


Figure 15: Scenario from Figure 14 repeated with different random seed.

4.3.2 Increasing use of feedforward as forward dynamics model improves

As discussed earlier, using feedforward control requires a high enough level of certainty in the forward dynamics model, such that the controller can predict with sufficient certainty that a particular control will be useful at a specific point in the future.

One good piece of evidence for this theory, is that in most of the feedforward weights shown so far (specifically for the lower noise cases where the task is achieved), the feedforward weights tend to be used less at later points in the control horizon. The later states in the control horizon are always the states about which the controller is least certain, and correspondingly, we see that the feedforward controls are used less in this region as predicted.

4.4 Noisy actuation

As discussed in section 2.1, noisy activation of muscles is an important property of the human control system. However, this concept is generally absent in a machine learning environment in which motors are actuated. For tasks such as controlling the inverted pendulum, any errors between the actual and desired torque are very small compared to the magnitude, and thus it can safely be assumed that there is no noise in the actuation without significantly degrading performance.

In order to investigate the effect of noisy actuation, it was desired to implement it in the machine learning setting and observe how it changed the way the task was achieved.

The controller in its existing state learns a control policy, $\pi(x)$, which is a distribution with a mean and variance. In order to implement a signal dependent actuation noise, the following modification to the policy was proposed, in which each muscle activation is corrupted by independent noise:

$$u = \pi(x) + \pi(x)\varepsilon \tag{2}$$

$$\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon) \tag{3}$$

Here, u is the muscle force for an individual muscle, but as the noise is independent for each muscle, the effect on the expectation and variance of u can be considered as six separate and identical 1D cases. Thus:

$$\mathbb{E}[u] = \mathbb{E}[\pi(x)] + \mathbb{E}[\pi(x)\varepsilon] \tag{4}$$

$$= \mathbb{E}[\pi(x)] \tag{5}$$

$$\mathbb{V}[u] = \mathbb{V}[\pi(x)] + \mathbb{V}[\pi(x)\varepsilon] + 2\mathbb{C}[\pi(x), \pi(x)\varepsilon] \tag{6}$$

And now:

$$\mathbb{V}[\pi(x)\varepsilon] = \mathbb{E}[\varepsilon]^2\mathbb{V}[\pi(x)] + \mathbb{E}[\pi(x)]^2\mathbb{V}[\varepsilon] + \mathbb{V}[\pi(x)]\mathbb{V}[\varepsilon] \tag{7}$$

$$\mathbb{C}[\pi(x), \pi(x)\varepsilon] = \mathbb{E}[\varepsilon]\mathbb{C}[\pi(x), \pi(x)] + \mathbb{E}[\pi(x)]\mathbb{C}[\pi(x), \varepsilon] \tag{8}$$

But as ε is zero-mean, and independent of $\pi(x)$, we know:

$$\mathbb{E}[\varepsilon] = 0, \mathbb{C}[\pi(x), \varepsilon] = 0 \tag{9}$$

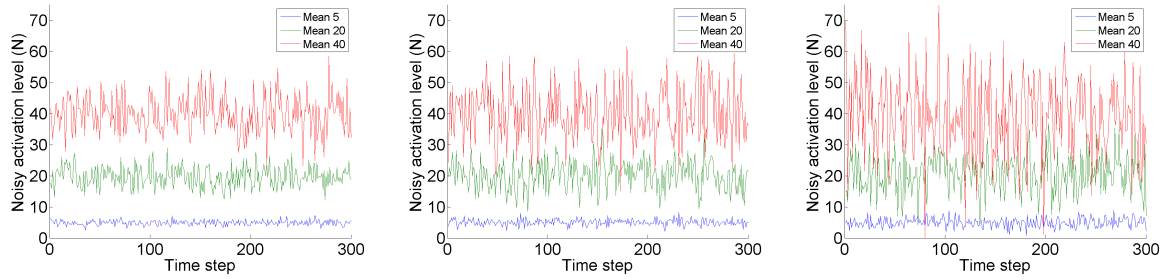
And so we have:

$$\mathbb{V}[u] = \mathbb{V}[\pi(x)] + \mathbb{E}[\pi(x)]^2\mathbb{V}[\varepsilon] + \mathbb{V}[\pi(x)]\mathbb{V}[\varepsilon] \tag{10}$$

Therefore, in order to implement noisy actuation as defined in equation 2, we need only

modify the variance of the control policy distribution. This was not a feature that already existed in the PILCO environment, and was implemented by modifying the policy variance in the function that performs rollouts of the controller. In the initial simulations, σ_ε was set to be uniform across muscles, i.e. $\sigma_\varepsilon = \alpha$, where α is a scalar controlling the level of noise.

An approximation of the actuation noise is illustrated in Figure 16.



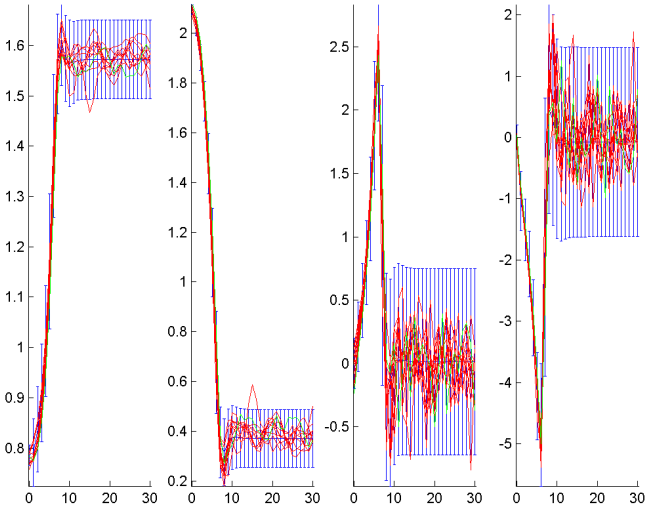
(a) $\alpha = 0.025$ (b) $\alpha = 0.05$ (c) $\alpha = 0.1$

Figure 16: The noisy actuation is illustrated above for zero variance in the policy ($\mathbb{V}[\pi(x)] = 0$), and constant means of 5, 20 and 40. This is only approximate, as in reality the signal is saturated and scaled to be strictly in the range [0, 40]. Thus, activations outside this range would be smoothly curtailed to fit within it.

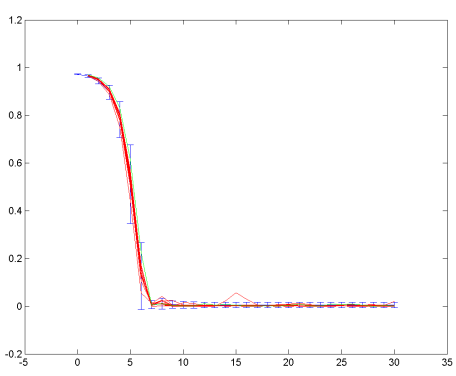
One obvious effect is that this should make the hand-paths intrinsically less smooth. More interesting, however, is how the controller strategy might change in response to this higher uncertainty. Redundancy in the arm means there are a huge number of ways to complete the task, and while previously the only way to differentiate between these different ways was to evaluate the expected cost, there is now an additional factor. By performing actions with smaller activations, the uncertainty in predictions can be reduced. Thus, one feasible outcome from increasing actuation noise would be to use smaller activations, and therefore perform a slower movement, in order to increase accuracy and decrease uncertainty.

Figure 17 shows the results of running the same simulation as in section 4.2 with the same level of observation noise, but with added actuation noise for $\alpha = 0.025$. Performance looks much the same as before as well, with a similar level of uncertainty in the dynamics model shown in Figure 17a, and similarly good execution of the task illustrated in each sub-figure. One subtle effect is that the speed of the movement is indeed slower, as predicted. This is illustrated most readily in the slightly slower drop to near-zero of the cost function.

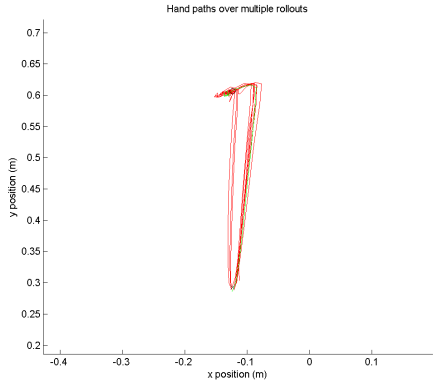
The full results for the two further levels of actuation noise are excluded, as the graphs show very similar results. The cost functions for three different levels of actuation noise



(a) Progression of 4 state variables over the control horizon.

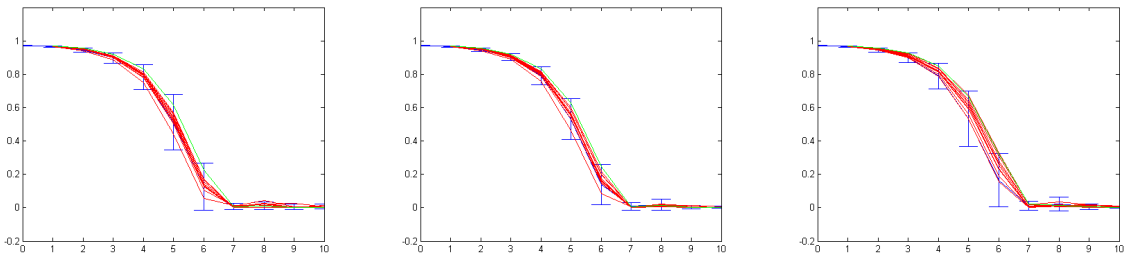


(b) Cost function.



(c) Hand-paths.

Figure 17: Performance on 20th trial for $\alpha = 0.025$. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.



(a) $\alpha = 0.025$. Total cost: 48.2 (b) $\alpha = 0.05$. Total cost: 48.6 (c) $\alpha = 0.1$. Total cost: 50.6

Figure 18: Zoomed in comparison of cost functions across 3 different levels of actuation noise for first 10 time steps. As noise increases, cost function increases mildly due to slower movement. Total cost quoted is across all trials.

can be compared directly in Figure 18 in order to illustrate the slightly slower movements made (causing a slightly slower drop in the cost function) under higher actuation noise.

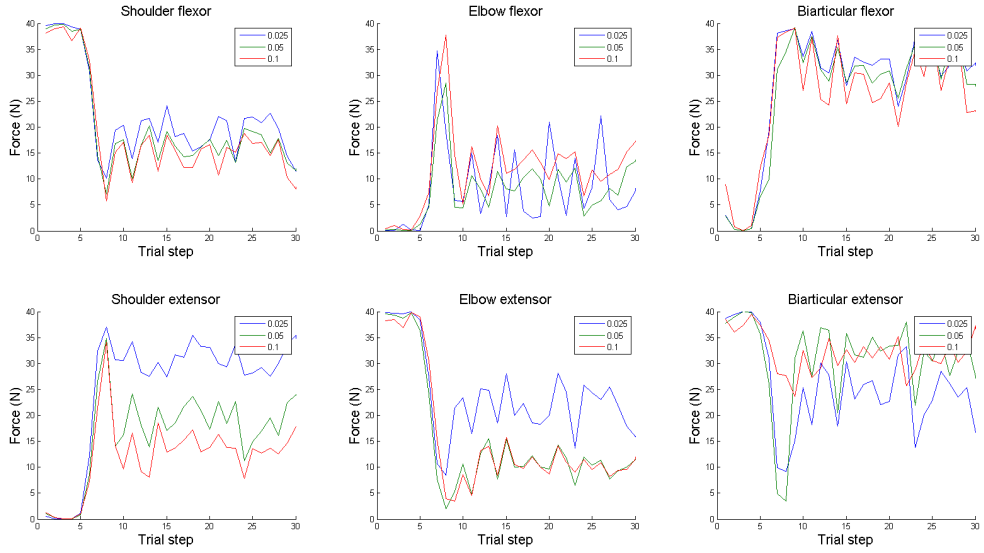


Figure 19: Muscle activations over the control horizon for 3 different levels of actuation noise.

A much clearer indication of the change in control strategy is shown in Figure 19. For 4 out of 6 muscles, shoulder flexor, shoulder extensor, elbow extensor and biarticular flexor, it is clear to see the predicted behaviour. The overall shape of the activations over time are similar, but for higher noise levels, the level of activation decreases. In addition, it is interesting to see that for all of these 4 except the biarticular flexor, the higher the activation level is for $\alpha = 0.025$, the larger the drop in activation is at higher noise levels. For example, the low noise case in the shoulder extensor uses almost the maximum muscle activation for much of the trial, but this drops significantly for the two higher noise cases. This is a pleasing result to see, as the noise added is proportional to the activation level, and so the controller does seem to be minimising its use of large activations in response to this.

The change in behaviour between different noise levels for the elbow flexor and biarticular extensor is less clear to interpret. There is much more overlap of the plots across the different simulations, and thus the change in noise level does not seem to be affecting the control of these particular muscles as much. In particular, the biarticular extensor activations seem to be lower for the lower noise case. This doesn't necessarily indicate a wholesale change in the use of this muscle, and could simply be a result of a different local optimum being found in this trial.

4.5 Delays in state feedback

Delays are a key feature in the human learning environment, as neurons take a significant amount of time to propagate signals over a distance. This means control decisions are made based on past states, and so the addition of delays to a system can lead to instability [2] if the stabilising control input has changed during the delay. In response to this then, the gain on the feedback controller must be relatively small so as not to overshoot and add to the instability of the system.

In contrast, delays are not generally a consideration for learning in robotic tasks. Electronic feedback is effectively instantaneous for any time scale of movement considered in this project. This means errors can be corrected with high feedback gain controllers, as any resulting overshoot will be observed quickly and can then be corrected.

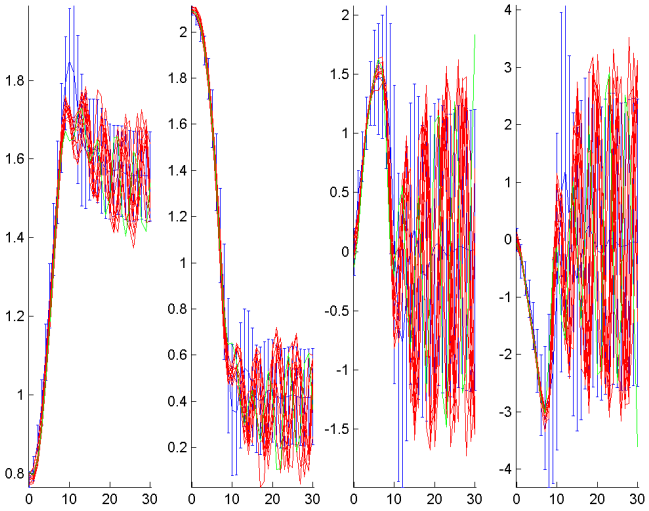
To look at the effects of delays in the machine learning environment then, artificial delays had to be added to the state observations on the order of 100ms. This was already implemented within the PILCO environment, and simply required specification of the length of feedback delay in the plant. With a 60ms delay, and otherwise identical settings to those analysed in section 4.2, a linear controller was still able to broadly achieve the task, as shown in Figure 20.

Figure 21 shows the trained feedback weights from section 4.2 and then the same scenario with progressively longer delays in state measurement. As predicted, the increased instability has forced the controller to lower the feedback gains in order to achieve the task stably. The last trial was unsuccessful (see Table 2) and so comparisons with the other trials has limited validity, but the feedback weights do tend to have less extreme values, as illustrated by a lack of any particularly dark or light weights.

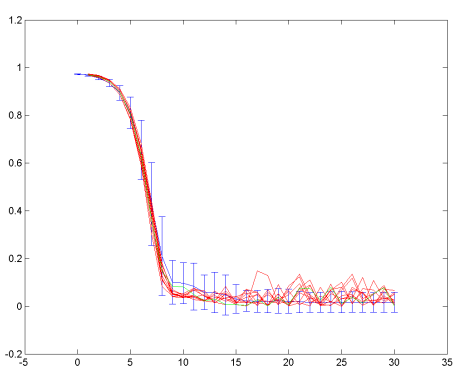
Delay (ms)	0	30	60	95
Total cost	60.1	52.0	72.7	108.6

Table 2: Total cost incurred over 11 rollouts of controller at the end of training with varying levels of state feedback delay. The task is considered failed for 95ms delay due to its total cost being well above the costs in the previous cases. Theoretically, total cost may range from 0 to 330, but typical performances range from 40 (excellent) to 260 (almost random).

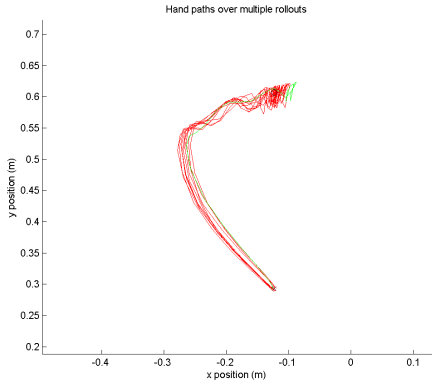
Delays are discussed later in this report as an important method to add instability to the system for investigations into stiffness and damping.



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.



(c) Hand-paths.

Figure 20: Performance after training with state observation delay of 60ms. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

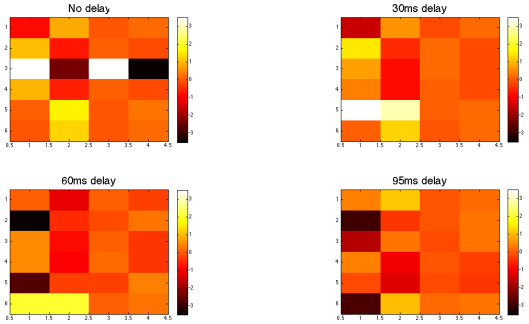
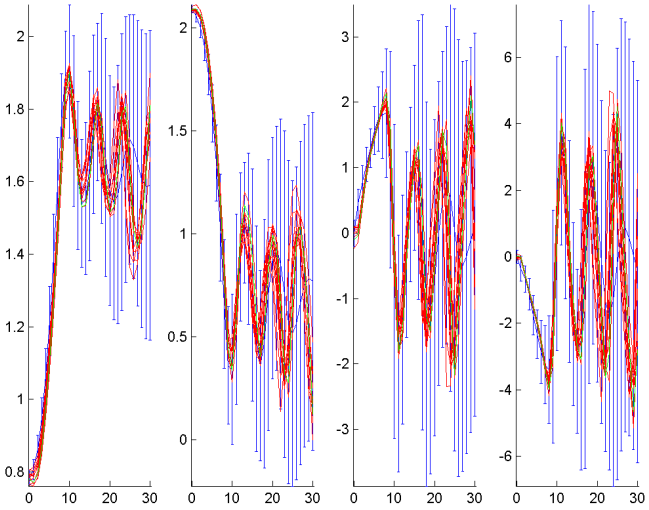
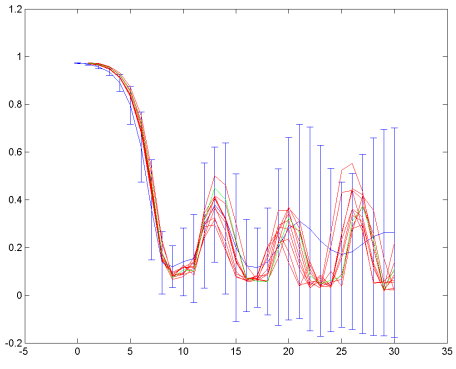


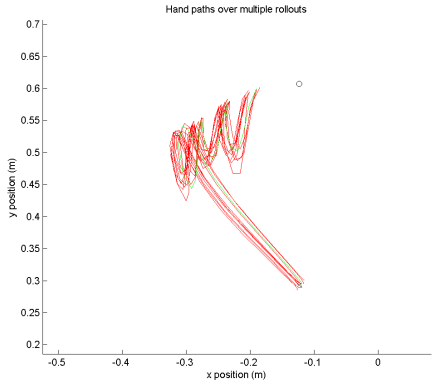
Figure 21: Heatmaps of feedback weights for the controller trained with varying lengths of feedback delay. Rows correspond to the 6 muscles in order, from shoulder flexor to biarticular extensor. Columns correspond to the 4 state variables: q_s, q_e, $\dot{q}_s$ and $\dot{q}_e$.



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.



(c) Hand-paths.

Figure 22: Performance after training with state observation delay of 95ms. Classified as failing using Table 2. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

4.6 Stiffness and Damping in the dynamics model

Stiffness and damping are very important properties for control of the human arm, arising from the passive mechanical properties of muscles [2][12]. They are passively resistive properties and both vary greatly with posture and muscle activation. They are particularly important in unstable dynamics, where deviations from a neutral path cannot be corrected by feedback control (assuming the feedback and actuation is not instantaneous).

In all of the literature found during this project, stiffness is implemented by considering deviation from a 'desired' location [9][13][14][15][16]. By treating the muscles as springs, the force generated by the stiffness of the muscles can be calculated with equations of the form $F = kx$. Here, k is the stiffness coefficient and depends on the activation level of

the muscle. The change in length of the muscle, x, is calculated as the distance between the current position and the 'desired' position.

A number of techniques are used to calculate a desired position. In some simulations, an inverse dynamics model of the system is used to compute a planned path achieved by a set of planned control inputs. Elsewhere, the current control inputs are used to predict the position of the hand at the next time step, and this position is used as the reference desired position. Some experiments also look at a regulation task, rather than a movement, in which case the desired position is the position at which the hand is being regulated.

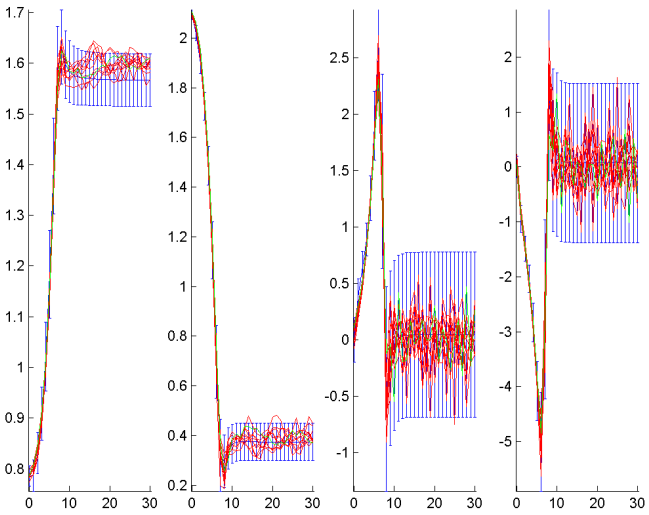
Unfortunately, however, specifying any desired position for the controller trained by PILCO would be invalid and incongruous with its operation. The algorithm is concerned with learning to predict and control the dynamics of a system with no prior knowledge of it. As such, the task is specified only by a cost function defining the most desirable state, and no constraints are made over the way the task is achieved. This means there is no pre-planned way to complete the task and the idea of a desired position does not make sense.

In order to implement a dynamics model with a material resistive property then, a different approach was required. The proposed implementation was to modify the muscle activations with a term proportional to the intended muscle activations and the velocity of the lengths of each muscle.

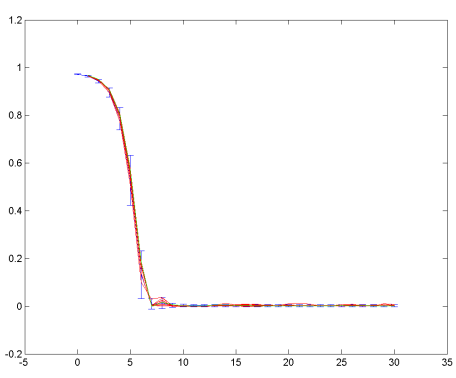
$$F_i = -a\, u_i |v_i| \tag{11}$$

Here, u_i is the muscle activation (which is always positive) of muscle i, $|v_i|$ is the absolute velocity of muscle length for muscle i and a is a positive tunable parameter to control the strength of this stiffness term. The F_i term is then added to the intended muscle activation for each muscle. The effect is to introduce a damping term which instantaneously resists the current velocity of the hand, and its strength is proportional to the level of muscle activation and the muscle velocity itself.

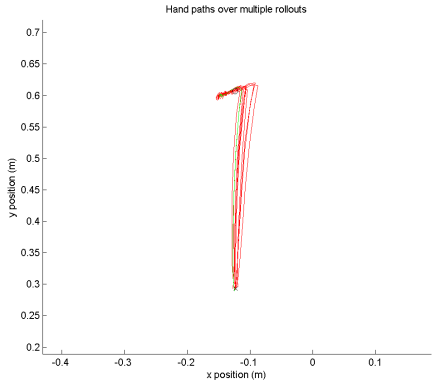
This additional term will also actually make controlling the arm more difficult, as velocities affected by the controller are resisted. However, the intention of adding this term is not to make the general control problem easier, but to introduce a property of the human arm that allows it to be controlled effectively in an unstable environment. The interesting result will then be whether PILCO optimises the controller to take advantage of this damping property in a similar way to humans. This is discussed further in section 4.7.1. Figures 23 and 24 show the results of training a controller on the same task as in section



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.



(c) Hand-paths.

Figure 23: Performance on 13th trial, with $a = 1$. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

4.2, but with the dynamics modified to include the damping term. Clearly, the controller is still able to achieve the task effectively, and the muscle activations are a similar shape, but with lower magnitudes. This is slightly surprising considering that the muscles are now slightly harder to actuate, but it seems likely that many of the large magnitude activations in the initial experiments were cancelling each other out.

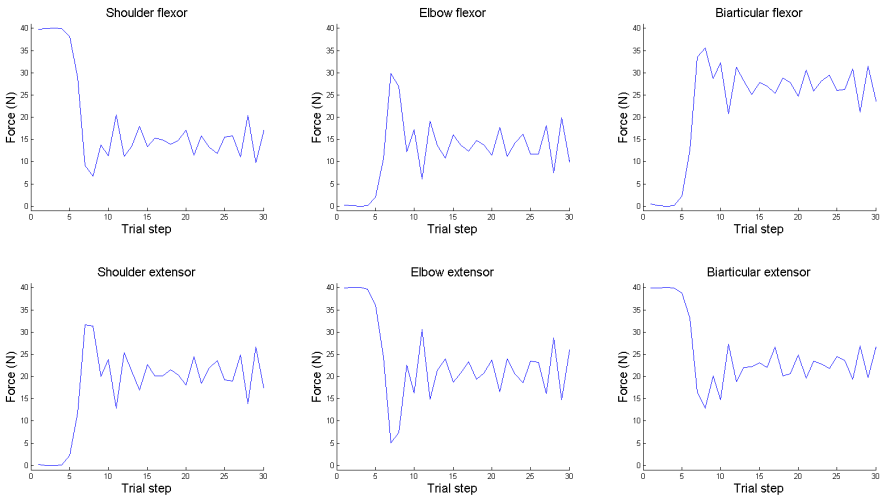


Figure 24: Muscle activations enacted to achieve the task as shown in Figure 23c.

4.7 Learning movements in a force field

4.7.1 Divergent force field

As discussed in section 4.6, and in the literature [7], stiffness and damping are particularly important properties when performing an unstable task. For example, when using a (flat head) screwdriver, any perturbations are very difficult to recover from as the task is performed in an unstable equilibrium. Co-contracting pairs of muscles changes the material properties of the muscles without producing any change in joint torque. The higher stiffness and damping results in an increased instantaneous resistance to perturbations, and thus increases the stability of the task.

This could be considered an alternative form of feedforward control, as the muscles are stiffened in anticipation of perturbations or deviations from the planned path. However, rather than specifying a particular action in the future, stiffness can be increased along the direction of instability in order to decrease the level of instability. Thus, increasing stiffness is not a prediction of the control inputs needed at a future state, but an indication that the controller expects perturbations but is uncertain of the exact controls that will be required.

As discussed in section 2.1, there are some interesting results in human trials which show that endpoint stiffness is directionally tuned to instability in the environment. At the beginning of the project, this was picked out as a key point of comparison between machine and human learning, as it would be interesting to see if the machine learning solution responds to instability in the same way. This is an imminently measurable phenomenon and therefore is a good point of comparison.

The unstable fields studied in [7] are divergent force fields (DFs), and this project looked at the 0° DF, such that the external forces applied to the hand, for hand-position (x, y), are:

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = K \begin{bmatrix} 1 \\ 0 \end{bmatrix} x \tag{12}$$

Here, K is a constant scalar which can be modified to change the strength of the field. This force field was an extension of the model of the arm, and therefore did not previously exist in the PILCO environment.

In order to test the implemented damping term properly, it was first necessary to find a task unstable enough such that a linear controller could not achieve it without damping. Given the strict limitation on muscle force being no greater than 40N, it would not be sensible to arbitrarily increase the strength of the field, as this could easily make the task almost impossible. Instead, K was set to 30, and to further destabilise the system, a delay of 95ms was added to the state feedback.

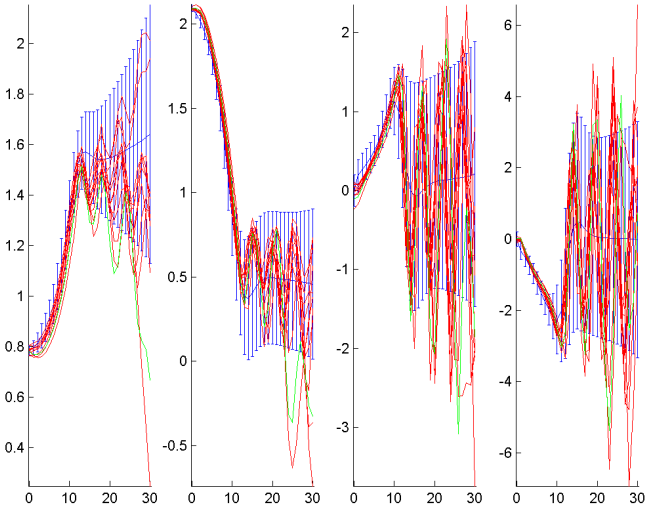
The resulting controller is shown in Figure 25. Although it is performing well on some of the rollouts, it is still getting dragged away by the DF field in a significant proportion of the rollouts.

Figure 26 shows the performance with the damping term included in the dynamics. Clearly, the cost function is more reliably minimised, and although the hand-paths are not ideal, the hand is reaching the target in every trial. So the damping term is helping the controller to achieve the task (given a sufficiently difficult task). However, in order to compare this result with human trials, it is also important to analyse how the controller strategy has changed.

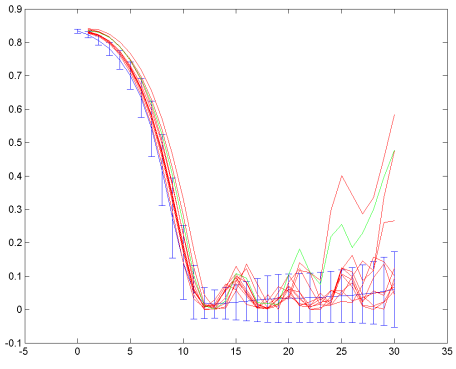
Figure 27 compares the muscle activations enacted by the controller, first without damping, and then with. The first observation is that in both cases, within each plot, the muscle activations generally increase after learning. This is to be expected in all cases because the DF field has been made strong enough to make the movement challenging.

More importantly, the muscle activations also change across the two sub-figures, 27a and 27b, after training (in green). Without damping, in Figure 27a, the muscle activations look analogous to bang-bang control. This seems to imply that there is no penalty for having low muscle activations, which is as expected.

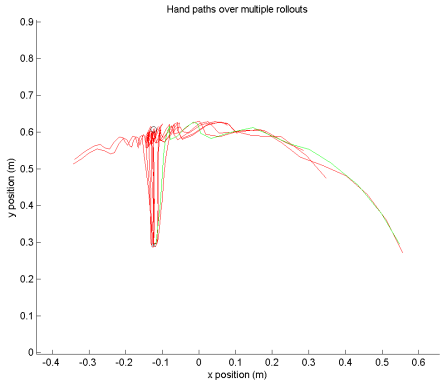
Conversely, Figure 27b shows that the muscle activations almost never go to zero when



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.

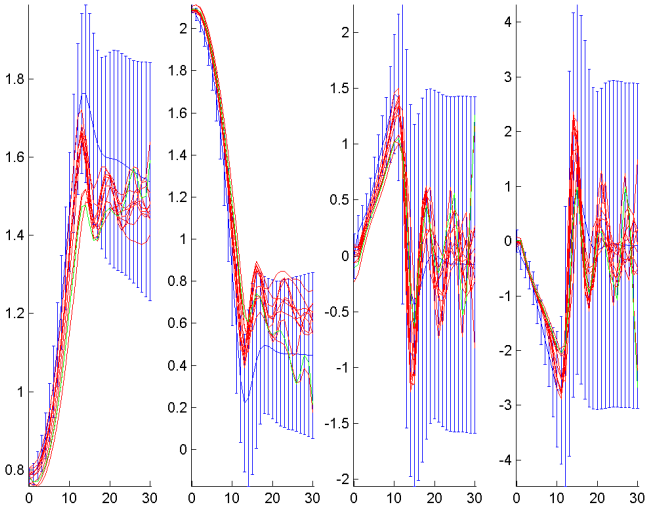


(c) Hand-paths.

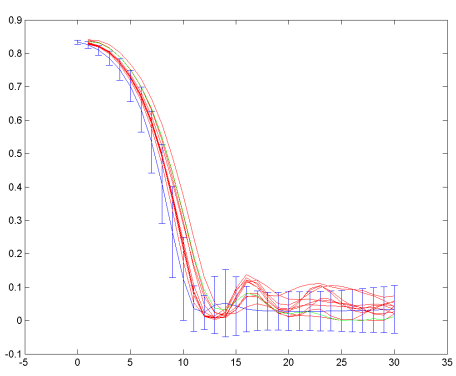
Figure 25: Performance on 20th trial without stiffness, in a strong DF field. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

damping is present. Instead, the muscles are modulated around a relatively high mean in order to control the arm while maintaining high damping coefficients throughout the control horizon. This shows that the controller has changed its strategy to take advantage of the increased stability resulting from highly activated muscles. This is a key result, and something that the project set out to analyse, but it is not completely analogous to the human learning trials.

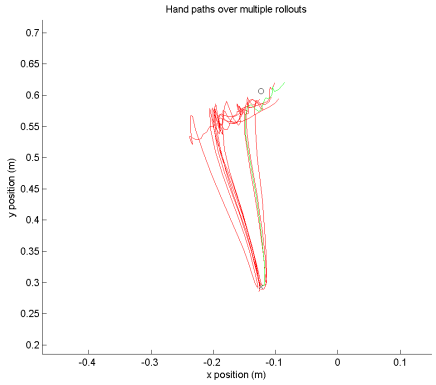
As well as increasing the effect of damping in general, it is also desired to see if the damping is specifically increased in the direction of the instability. This is somewhat analogous to [7]. Figure 29 shows the damping ellipses for the controller trained with and without damping in the dynamics model. Clearly, plotting the damping for a trial in which the damping term is not present is not technically accurate, but the muscle activations have



(a) Progression of 4 state variables over the control horizon.



(b) Cost function.



(c) Hand-paths.

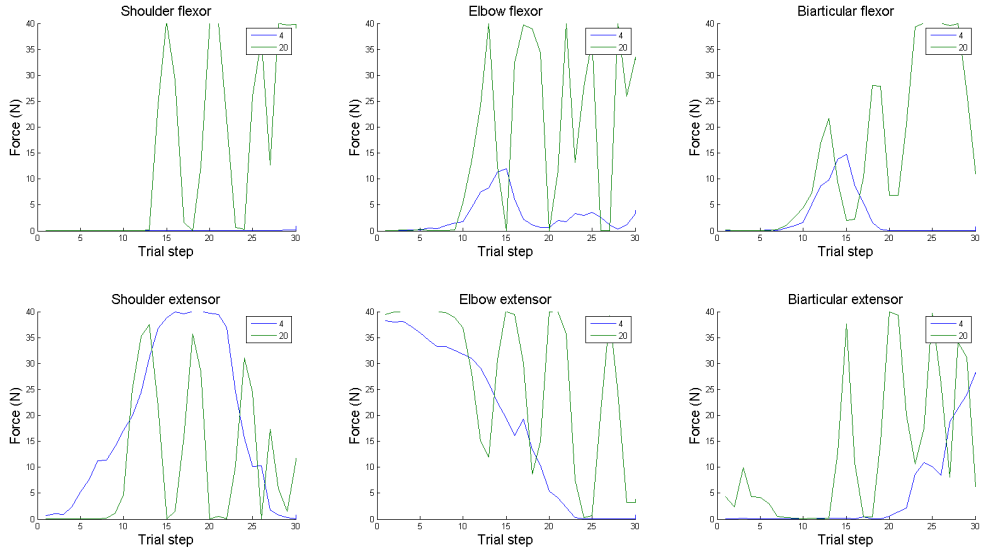
Figure 26: Performance on 20th trial with stiffness, in a strong DF field. Mean and error bars of predictions shown in blue. Controlled trial in green, and repeated (but unseen by controller) trials in red.

been taken as the damping coefficients as they would have if damping was added.

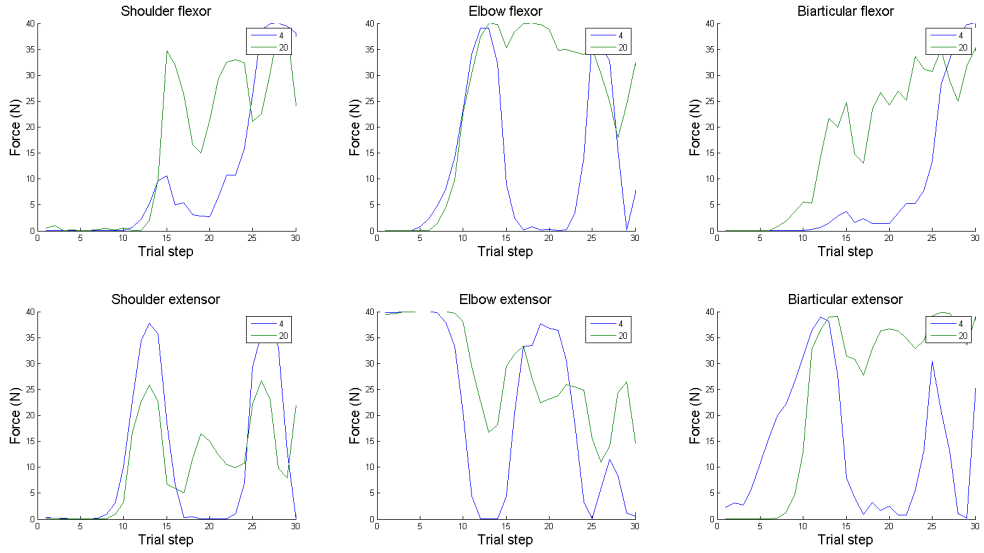
The calculation of damping was derived using [3] and [9], and is as follows. First the damping coefficients in muscle space are simply:

$$D_m = a \times \texttt{diag(musc)} \tag{13}$$

Here, `musc` is the 1×6 vector of muscle activations, and a is the tunable damping parameter, which is set to 1. The damping coefficients form a diagonal matrix as there are no co-dependencies between muscles. Next the damping coefficients are converted to joint



(a) Muscle activations without damping term. Strategy looks very similar to the initial simulations in section 4.2, which was the same system except without the state feedback delay or DF field.



(b) Muscle activations with damping term.

Figure 27: Muscle activations enacted at each time step in the first trial which comes close to achieving the task (in blue), and the final trial after learning (in green). Muscle activations shown for training both with and without damping term in the dynamics.

space, using the muscle to joint space Jacobian, J_m:

$$D_q = J_m^T D_m J_m \tag{14}$$

Then, in Cartesian coordinates, using the joint space to Cartesian space Jacobian, J:

$$D_x = J^{-T} D_q J^{-1} \tag{15}$$

The resulting ellipses in Figure 29 confirm that the increased muscle activations have indeed increased the amount of damping in order to stabilise the task. However, there doesn't seem to be any evidence that the damping has been tuned to increase only in the direction of instability. Instead, it has been increased in all directions indiscriminately.

Given that PILCO knows nothing about the dynamics of the system other than what it learns from experience, it would be very difficult to learn about the strategy of co-contraction. In human learning, agonist/antagonist pairs of muscles are co-contracted to change the stiffness and damping in the arm, without producing any net change in torque. However, this strategy is quite intricate, and it is not surprising that the controller has not 'discovered' it.

Furthermore, as this simulation was run without actuation noise, and the simulated arm cannot get tired or run out of energy, there is actually no incentive to reduce unnecessarily high muscle activations. Thus, as long there are no problems to do with muscle activations saturating at their maximum, the strategy to increase the damping in all directions is just as desirable as a strategy which directionally tunes damping. It would also seem likely that the former is easier to discover by random trials.

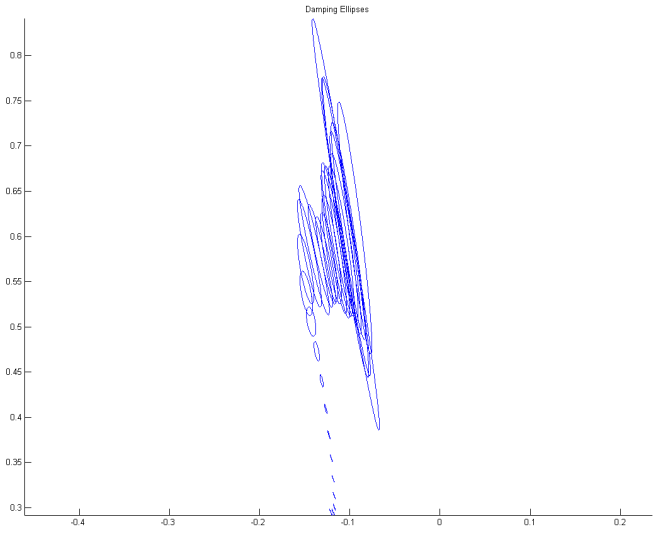


Figure 28: Damping ellipses at each point along the hand-path scaled down to illustrate the changing shape over the control horizon. There is surprisingly little damping during the bulk of the movement.

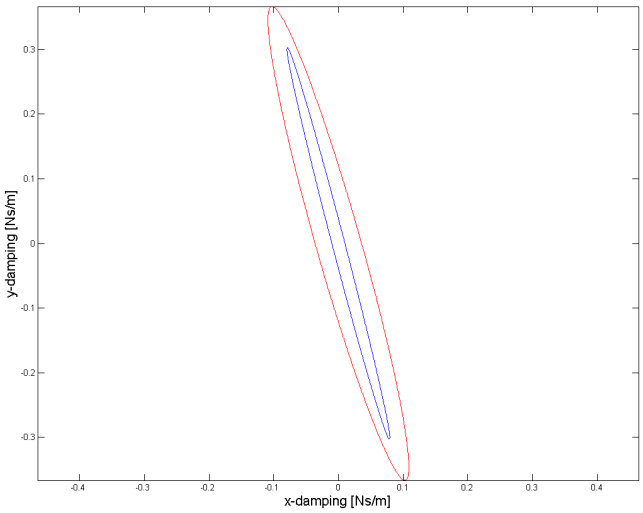


Figure 29: Damping ellipses without damping (blue ellipse) and with damping (red ellipse) term.

5 Future work

Several components of the human learning problem have been studied in a machine learning environment. In the interest of making the analysis simpler and easier to explain convincingly, these components have mostly been studied in isolation. Clearly one direction in which this project could be taken, would be to analyse the interplay between these components in more depth and detail. In particular, it would be interesting to look at the effect adding noisy actuation to the simulations from section 4.7 would have. The controller has to increase the muscle activations in order to stabilise the problem, but this is the area where actuation noise is most extreme.

There are also many parameters of the human arm model that have been set to a value that 'works', and then left unchanged in all the simulations. For example, the maximum muscle force was set to 40N, and the effect of changing this limit has not been investigated. This would certainly be important in order to make the model more comparable with human learning experiments.

Another area that could benefit from further investigations is implementing stiffness in the dynamics model and improving the model of damping. A satisfactory way to implement stiffness was never found, but it is an important property to simulate if convincing comparisons are to be made with human learning experiments. Additionally, the strength of the damping in the system was not investigated extensively, and it would be interesting to vary the level of damping and analyse whether the controller strategy changes.

6 Conclusions

1. PILCO is capable of learning the forward dynamics model and an effective controller for a simple model of the human arm with a very small amount of experience. Depending on the exact system and external forces, it will generally take between 30s and 60s of experience to perform the task successfully.

2. There is high amount of redundancy in the system, and as such, the muscle activations used to perform the task can be very different on random restarts.

3. Adding feedforward control to the policy increases the amount of redundancy and introduces a trade-off between feedforward and feedback control. A less accurate dynamics model will cause the controller to rely more on feedback, while greater observation noise will result in more active feedforward controls.

4. Adding signal-dependent noise to the actuation of the muscles introduces an uncertainty penalty on strong activations. As a result, the controller tends to reduce muscle activations and thus minimise the uncertainty.

5. Adding delays to the state measurements increases instability in the system. In response to this, the controller has to reduce the gain in the feedback weights, thus avoiding further destabilising controls.

6. Although stiffness was not implemented, damping was added to the model and had little effect in tasks which were not previously challenging.

7. Damping proved to be critical for successfully performing a reaching movement in a very unstable environment. In an unstable field, and with delayed state feedback, the controller utilised activation-dependent damping by using higher muscle activations throughout the control horizon. This stabilised the system enough for the task to be achieved.

8. However, the change in damping did not quite match up with the analogous change in stiffness seen in human trials for unstable fields. Instead of tuning the damping to only increase in the direction of instability, damping was increased in all directions.

A Code Listing

The full code listing for PILCO can be found at: https://mlg.eng.cam.ac.uk/svn/carl/ctrl
All code specific to this project may be found in the arm scenario subdirectory:
https://mlg.eng.cam.ac.uk/svn/carl/ctrl/scenarios/arm

Permission must be obtained from Carl Rasmussen in order to access this repository.

B Risk Assessment

Initially, the risk assessment stated that there was low risk except for any work with robotics, for which a full risk assessment would have to be completed. As all the work completed in this project was simulation based, the initial risk assessment was appropriate, and ergonomic considerations were the only risks concerned.

These considerations are as follows. The chair should be adjusted to ensure that the back is properly supported and knees are level with hips. The keyboard should be used with wrists and forearms straight, with 6 inches of desk space in front for support. Feet should be rested on the floor and the top of the screen should be at eye level. Finally, the screen should be kept free of glare to avoid eye strain.

References

[1] Marc P. Deisenroth and Carl E. Rasmussen (2011), "PILCO: A Model-Based and Data-Efficient Approach to Policy Search"

[2] David W. Franklin and Daniel M. Wolpert (2011), "Computational Mechanisms of Sensorimotor Control", *Neuron*, 72(3) pp. 425 - 442

[3] Etienne Burdet, David W. Franklin and Theodore E. Milner, (2013), *Human Robotics: Neuromechanics and Motor Control*, chapter 5, MIT Press, Cambridge, MA, USA

[4] Etienne Burdet, David W. Franklin and Theodore E. Milner, (2013), *Human Robotics: Neuromechanics and Motor Control*, chapter 8, MIT Press, Cambridge, MA, USA

[5] Harris, C.M., and Wolpert, D.M. (1998), "Signal-dependent noise determines motor planning", *Nature* 394, 780784

[6] Daniel M Wolpert and J Randall Flanagan, (2010), "Q&A: Robotics as a tool to understand the brain", *BMC Biology*, 8:92

[7] Franklin DW, Liaw G, Milner TE, Osu, R, Burdet E, and Kawato M (2007), "Endpoint stiffness of the arm is directionally tuned to instability in the environment", *Journal of Neuroscience*, 27, 7705-7716

[8] Burdet E, Tee KP, Mareels I, Milner TE, Chew CM, Franklin DW, Osu R, and Kawato M (2006), "Stability and motor adaptation in human arm movements", *Biological Cybernetics*, 94, 20-32

[9] Franklin, D. W., Burdet, E., Tee, K. P., Osu, R., Chew, C., Milner, T., Kawato, M. (2008), "CNS learns stable, accurate and efficient movements using a simple algorithm". *Journal of Neuroscience*, 28(44): 11165-11173

[10] Marc Peter Deisenroth (2010) "Efficient reinforcement learning using Gaussian processes", PhD thesis, Karlsruhe Institute of Technology, Karlsruhe, Germany

[11] Buchli, J., Stulp, F., Theodorou, E., Schaal, S. (2011) "Learning variable impedance control", *International Journal of Robotics Research*

[12] Burdet, E., Osu , R., Frankl in, D.W., Milner, T.E., and Kawato, M. (2001), "The central nervous system stabilizes unstable dynamics by learning optimal impedance", *Nature* 414, 446 - 449

[13] Mitrovic, D., Klanke, S., Osu, R., Kawato, M., Vijayakumar, S. (2010), "A computational model of limb impedance control based on principles of internal model uncertainty", *PloS One*, 5(10), e13601. doi:10.1371/journal.pone.0013601

[14] Buchli, J., Theodorou, E., Stulp, F., Schaal, S., (2011), "Learning variable impedance control", *The International Journal of Robotics Research June* vol. 30 no. 7 820-833

[15] Todorov, E., Li, W., Pan, X., (2005), "From task parameters to motor synergies: A hierarchical framework for approximately-optimal control of redundant manipulators", *Journal of Robotic Systems*, 22(11), 691710. doi:10.1002/rob.20093

[16] Todorov, E., (2003), "On the role of primary motor cortex in arm movement control", *Progress in Motor Control III*, chap 6, pp 125-166