ENGINEERING TRIPOS PART IIA

SF3: MACHINE LEARNING

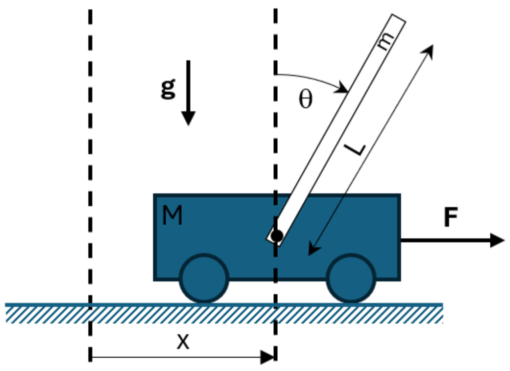
Final Report

Hana Iza Kim

Pembroke College, June 2025

Abstract

This report studies the inverted pendulum problem ('cartpole') using both linear and non-linear modelling approaches to analyse and control the system's behaviour. We begin by studying the system dynamics in the absence of external forces, and show that a non-linear model with Gaussian kernel offers greater accuracy than a simple linear model. Both models are then extended to account for a driving input force, enabling the design of a linear control policy aimed at stabilising the system around a target state. The development of the models and the control policy requires parameter tuning to optimise their performance. This is accomplished by defining suitable *objective* functions, and performing gradient-based optimisation. Finally, the impact of noise on system stability and control performance is evaluated.



1 Introduction

The CartPole system is a classic problem in control theory and reinforcement learning, and here we report the findings of our study using both linear and non-linear approaches to model and control the system's behaviour. The system is described with four state variables: the cart position x, the cart velocity x', the pole angle θ and the pole angular velocity θ'. Initial models were expanded to account for an external driving force and linear control policies were implemented to control the pole in its unstable equilibrium position.

To improve the accuracy and temporal resolution of the computational model of the cart-pole system (*CartPole.py*), the time step of the Euler integrator (*self.delta_time*) was reduced from 0.1 to 0.01. The maximum force limit (*self.max_force*) was also adjusted for parts of the study, but any additional temporary changes made to the file (e.g. changing the mass ratios and removing friction) to gain a better understanding of the system, were reverted, and results presented here are obtained with the default values in the original file.

Task 1.1 - Dynamical Simulation

In this task, we use the provided *performAction* function with zero external force, i.e. free dynamics, to simulate the motion of the cart and pole for different initial conditions.

Figure 1 shows the dynamics when the system starts stationary at the origin ($x_o = x'_o = \theta'_o = 0$) with the pole at various positions ($0 < \theta_o < \pi$). Symmetric dynamics are obtained when $-\pi < \theta_o < 0$ (data not shown). When $\theta_o = \pi$ and $\theta_o = 0$, the system remains in its stable/unstable equilibrium position. For any other initial angle, dynamics are observed. For presentation purposes, angles in this section are remapped to [0-2π], to avoid discontinuities around the equilibrium position $\theta = \pi$.

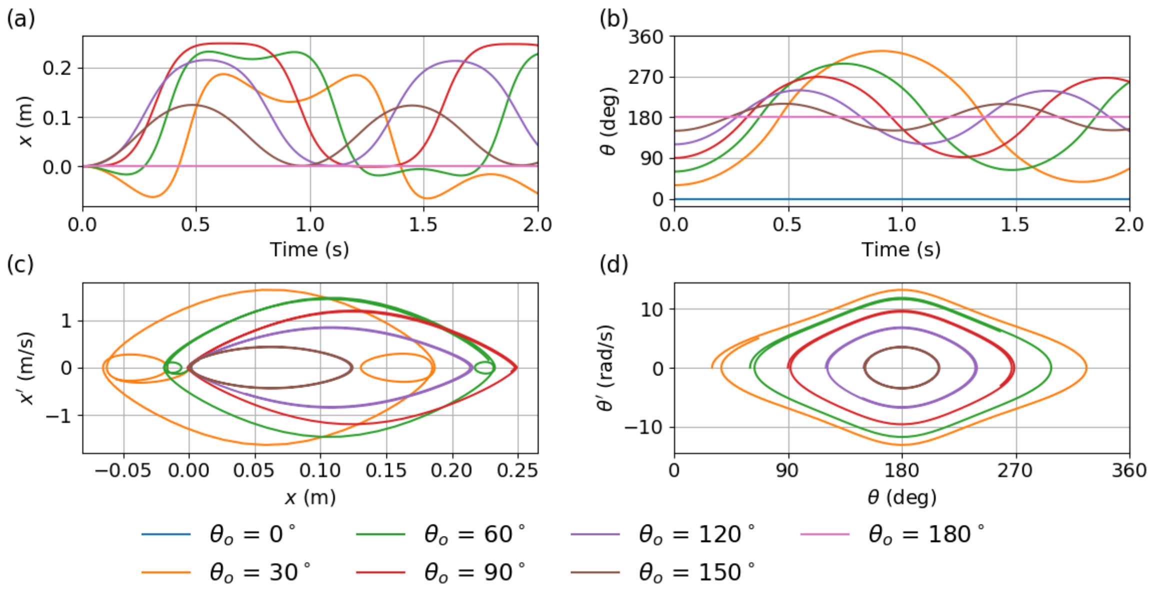


Figure 1: System dynamics for different initial positions of the pole: (a) Cart position x. (b) Pole position θ. (c) Cart phase plot (x' vs x). (d) Pole phase plot (θ' vs θ).

Since there are no external forces acting on the system, if there were no frictional forces, the total momentum would be conserved and the centre of mass of the system would remain stationary. As the pole swings, its centre of mass moves to the left and therefore the cart needs to move to the right; and vice-versa when the pole swings back to the right (see the $\theta_o = 150°$ lines in Figure 1). In other words, the cart oscillates around an equilibrium position as the pole swings back and forth. As expected, the maximum oscillation of the cart takes place when the pole starts at $\theta_o = \pi/2$ (see Figure 1a). When the initial position of the pole is above the horizontal ($\theta_o < \pi/2$), its centre of mass moves to the right before it starts moving to the left as it swings towards the equilibrium position. This causes an additional oscillation in the cart motion, which can be recognised in the phase plot as small loops at the extreme values of x (see Figure 1c). If there were no energy losses in the system, the dynamics would be perfectly periodic, i.e. the trajectories in the phase plots would close themselves. However, it can be seen that due to the frictional losses (and to the finite time step and rounding errors), the trajectories do not close perfectly (see Figure 1c,d). Instead, they slowly drift towards the equilibrium position ($\theta = \pi, \theta' = 0$) as energy is dissipated in the system.

Task 1.2 - Changes of State: True Model

The state of the system ($X = [x, x', \theta, \theta']$) after one time step depends on the previous state of the system, i.e. $X_{t+1} = f(X_t)$. In principle, this function f is complex, but we can study the change in the state variables from a given state, i.e. $\Delta X = X_{t+1} - X_t$, based on observations of the system dynamics without any need for its governing equations.

Figure 2 shows the change in state variables as a function of the initial pole position θ_o and angular velocity θ'_o, assuming the cart is initially at rest ($x_o = x'_o = 0$). Although not shown explicitly, ΔX does not depend on the cart position x. While the change in pole angle $\Delta\theta$ is approximately independent of the pole angle θ and proportional to the pole velocity θ', i.e. $\Delta\theta \sim \theta' dt$ (Figure 2c), the changes in cart position, cart velocity and pole velocity depend non-linearly on the initial angle and velocity of the pole (Figure 2a,b,d). It is noted that the contour lines in Figure 2c wobble due to the finite mass of the cart, i.e. the pole does not pivot around a stationary point. Qualitatively, the same dynamics are obtained when $x'_o \neq 0$.

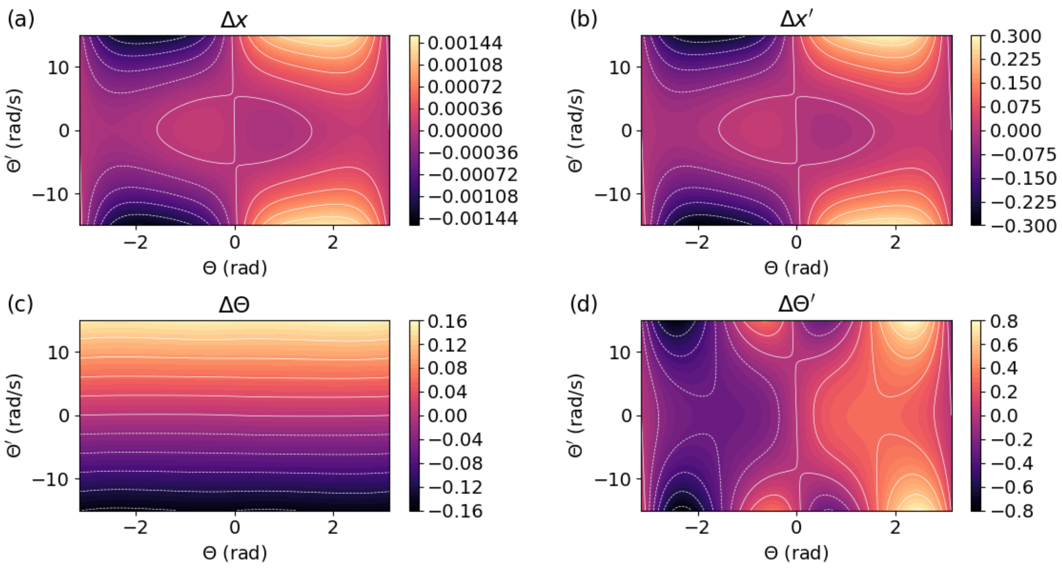


Figure 2: Change of state variables for $x = x' = 0$ and different θ and θ' initial conditions. (a) Δx, (b) $\Delta x'$, (c) $\Delta\theta$ and (d) $\Delta\theta'$.

Task 1.3 - Linear Model

The simplest model of the system based on observations of its dynamics is a linear one in which the change in state variables is proportional to the state of the system:

$$Y = \Delta X = XC^T \tag{1}$$

where Y is the change in state X after one time step. To build the model, we observe ΔX for a number of random initial states and use that information to determine the 4×4 matrix C using standard linear regression. Figure 3 compares the 'true' change in state variables (as predicted by the numerical integration of the equations of motion) after one time step with those predicted by the linear model. In addition, a subset of initial states in which the pole is at a small angle ($|\theta| < 15°$) and angular velocity ($|\theta'| < 3$ rad/s) is also shown (orange points).

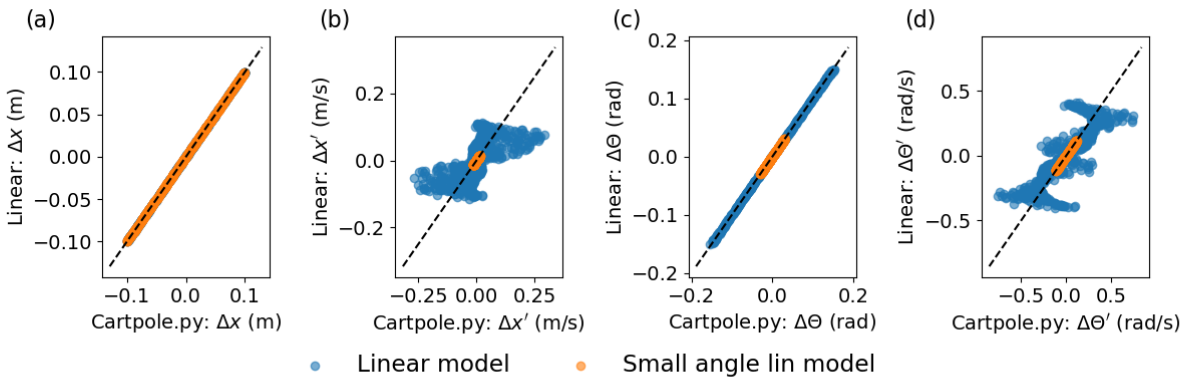


Figure 3: Comparison of state variable changes for 500 randomly selected initial states. $|x| < 10$, $|x'| < 10$, $|\theta| < \pi$, $|\theta'| < 15$. Small angle: $|\theta| < \pi/12$, $|\theta'| < 3$.

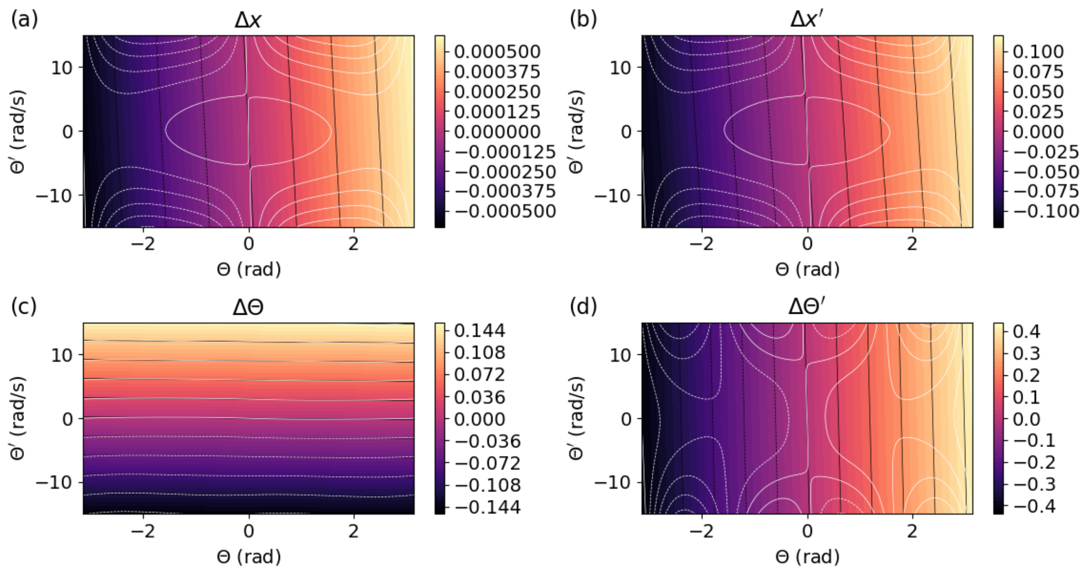


Figure 4: Change of state variables for $x = x' = 0$ and different θ and θ' initial conditions calculated using the linear model. (a) Δx, (b) $\Delta x'$, (c) $\Delta\theta$ and (d) $\Delta\theta'$. White overlaid lines correspond to the 'true' changes copied from Figure 2.

Points lying close to the black dashed lines in Figure 3 represent state changes where the linear model agrees with the 'true' simulation, and therefore gives accurate predictions. As shown in Figure 3a,c, the change in cart position and pole angle are well predicted by the linear model. However, the changes in cart and pole velocities are not (see Figure 3b,d). Despite the limitations of the linear model, it performs well for small-angles around $\theta = 0$ (orange points).

Figure 4 presents the predicted change in state variables using the linear model, equivalent to the 'true' changes in state variables shown in Figure 2. A few contour lines from Figure 2 have been overlaid in white on Figure 4 to aid the comparison. The prediction for $\Delta\theta$ closely matches the 'true' behaviour.

Task 1.4 - Linear Model: Rollout

We can use the linear model to simulate the system's time evolution and assess its suitability by comparing the simulation results with those obtained from direct integration of the equations of motion (*CartPole.py* model). Figure 5 shows the system dynamics when the cart and pole are initially stationary ($x_o = x'_o = \theta'_o = 0$) and the pole is let go at different angles. Although the state evolutions in both models are similar for the first few time steps, they quickly start to diverge (see further discussion in the interim report). This is the case even for a starting angle of $\theta_o = \pi$, i.e. the system's stable equilibrium position. This may seem surprising at first, but it is to be expected given the poor fit of the linear model at large angles $|\theta| >> 0$ (Figure 4).

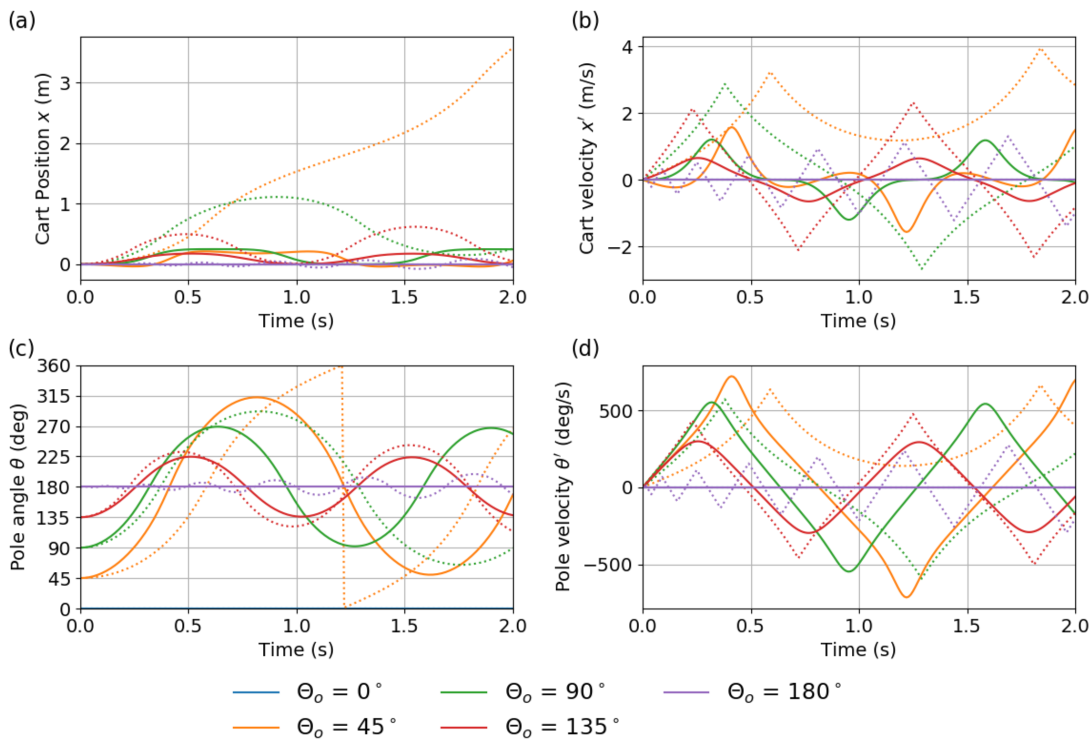


Figure 5: System dynamics predicted by the linear model (dotted lines) and the 'true' dynamics obtained by integrating the equations of motion (solid lines) for various initial positions of the pole: (a) Cart position x. (b) Cart velocity x' (c) Pole angle θ. (d) Pole velocity θ'.

Task 2.1 - Non-linear model

The linear model fails to capture accurately the non-linear dynamics of the cart-pole system. Therefore, instead of predicting the evolution of the system using the linear relationship shown in Equation (1), we construct an alternative non-linear model that uses a linear combination of M non-linear basis functions $K(X, X_i)$. We use a Gaussian kernel with basis centred at X_i:

$$Y = \Delta X = \sum_{i=1}^{M} \alpha_i \, K(X, X_i) = \sum_{i=1}^{M} \alpha_i \exp\left(- \sum_{j=1} \frac{\left(x^{(j)} - x_i^{(j)}\right)^2}{2\sigma_j^2} \right) \tag{2}$$

In Equation (2), the sum over j corresponds to different components of the state vector (X) and the parameters α_i are to be found using linear regression over training data. We substitute the angular difference term $(\theta - \theta_i)^2$ for $\sin^2((\theta - \theta_i)/2)$ to capture the 2π periodicity of the state variable θ.

Equation (2) can be written in matrix form as $K_{NN}\ \alpha_N = Y_N$, where the i, j entry of the K_{NN} matrix is the Gaussian kernel. To improve the stability of the model, we introduce *Tikhonov regularisation*, i.e. $(K_{NN} - \lambda I)\alpha_N = Y_N$, where the smaller the regularisation parameter λ is, the closer the fit is to the original data, but the more unstable the model becomes.

We limit the number of basis to $M < N$, by selecting M random samples from the training set. As shown in Figure 6, increasing the training set size N, increasing the number of basis M and reducing the regulation parameter λ, all improve the accuracy of the model, i.e. the mean squared error between the non-linear model's prediction and the actual state changes decreases. This is expected, as more training data better represents the underlying system, and a larger number of basis enables the model to better capture its features. However, we deliberately choose $M < N$ to avoid overfitting, reduce computational cost, and improve generalisation.

The M coefficients of α required for the model can be obtained by linear regression to a training set of N data points using equation (3):

$$\alpha_M = \left(K_{MN}K_{NM} + \lambda K_{MM}\right)^{-1} K_{MN}Y_N \tag{3}$$

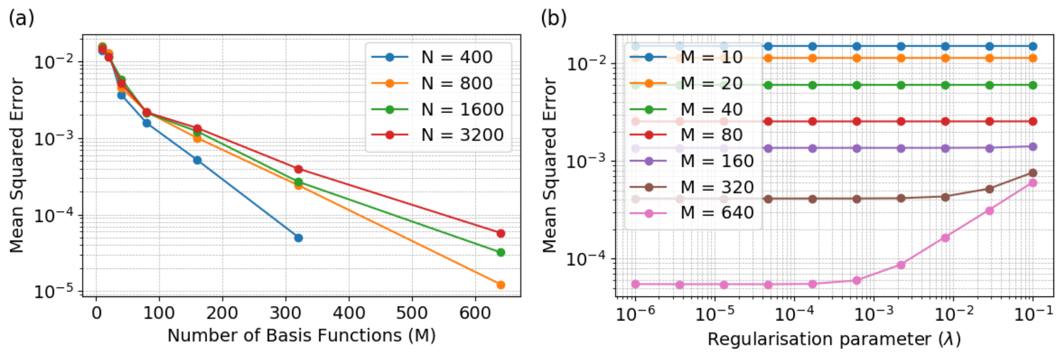


Figure 6: (a) Mean squared error (MSE) evaluated over different values of N (number of training samples) and M (number of basis functions). (b) MSE evaluated for a fixed data set $N = 3200$, scanned over varying values of M and regularisation parameter λ.

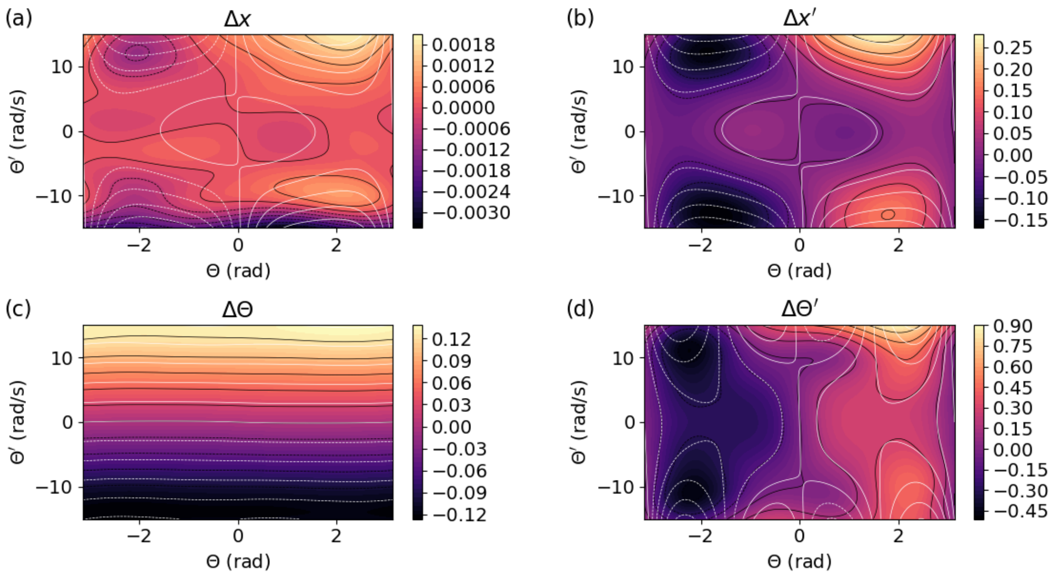


Figure 7: Colour map and black contours lines showing the change in state variables as predicted by the non-linear model with initial position and velocity $x = x' = 0$. Superimposed white lines show the contours of the 'true' change as copied from Figure 2.

Based on data from Figure 6, we select the following default parameters: number of training samples $N = 3200$, number of basis functions $M = 500$, regularisation parameter $\lambda = 10^{-3}$ and kernel bandwidth σ_j equal to the standard deviation of each state variable X_j. Figure 7 shows how the Gaussian-kernel based model captures non-linear behaviour across different regions of the state space. Without tuning the hyperparameters (σ_j and λ), the non-linear model appears to perform significantly better than the linear model (compare Figure 7 and Figure 4).

Figure 8 shows non-linear model rollouts from rest with varying initial pole angles. Predictions match the 'true' dynamics for 1–2 seconds before accumulating errors lead to growing deviations. Predictions are more accurate for angles closer to the stable equilibrium position.

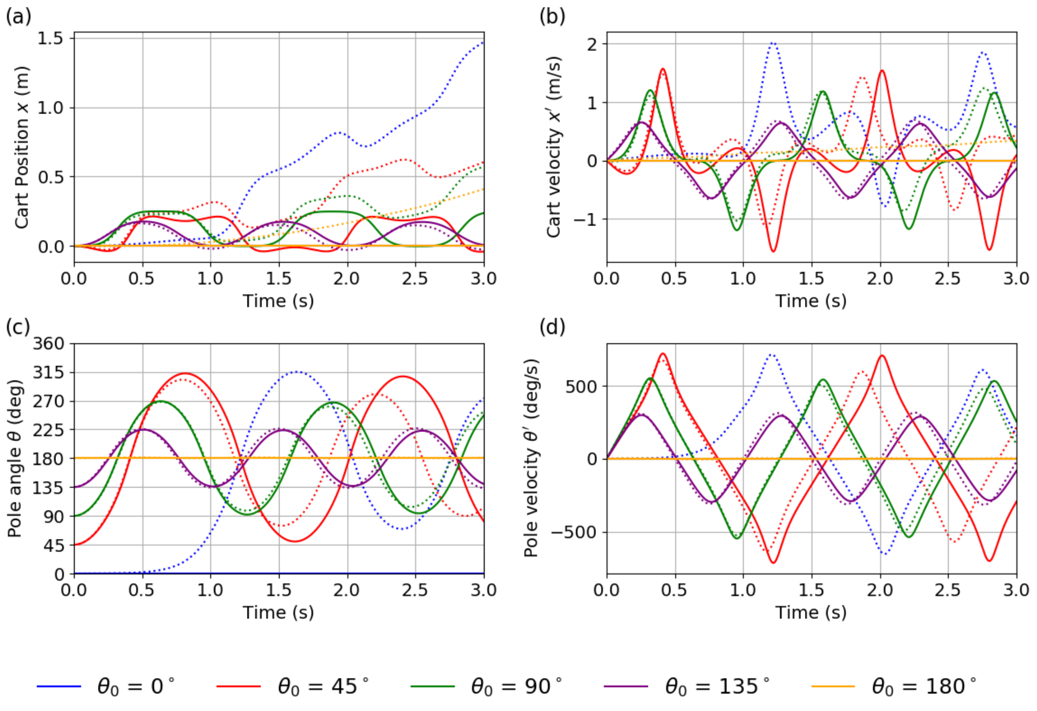


Figure 8: System dynamics predicted by the non-linear model (dotted lines) and the 'true' dynamics obtained by integrating the equations of motion (solid lines) for various initial positions of the pole: (a) Cart position x. (b) Cart velocity x' (c) Pole angle θ. (d) Pole velocity θ'.

Task 2.2 - Gradient based hyper-parameter tuning

The non-linear model represents an improvement from the linear one (compare Figures 5 and 8). However, the model could be further improved by optimising the kernel hyperparameters (σ_j and λ). Here we perform gradient-based optimisation of the hyperparameters to minimise the prediction error of the non-linear model. Specifically, we use the mean squared error (MSE) between the true and predicted change in state variables as the objective function we want to minimise. Gradients of the objective function with respect to the hyperparameters are then computed using JAX, which enable efficient and automatic differentiation, and we use the L-BFGS-B optimisation algorithm to carry out the optimisation step (*scipy.optimize.minimize*).

For the next comparison, and to better illustrate the impact of hyperparameter optimisation, the number of basis functions is reduced to $M = 50$. As before, the initial kernel hyperparameters σ_j are set to the standard deviation of each corresponding state variable, computed from N training observations. The model's performance with these initial hyperparameters are shown as blue points in Figure 9. The performance of the non-linear model with optimised hyperparameters are shown as red points, showing a significantly closer fit to the ideal black dashed line. To validate the model, Figure 9 also shows the performance of the optimised model for additional test cases (yellow points), which shows similar performance to the training set.

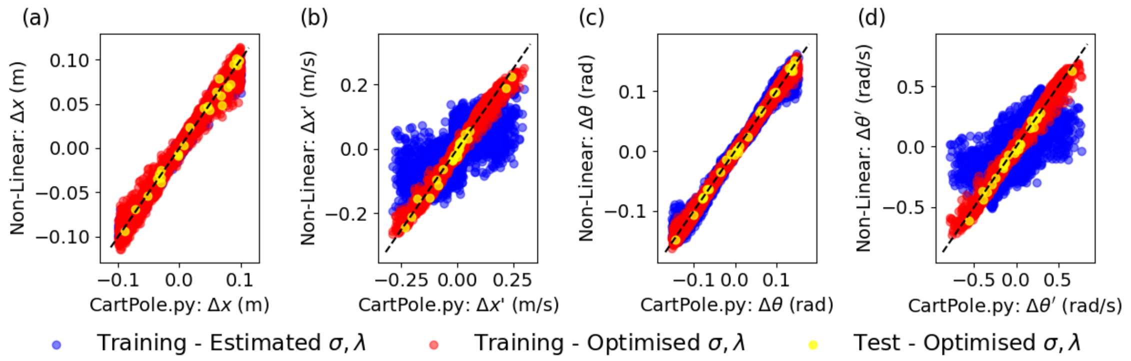


Figure 9: Comparison of state variable changes for $N = 3200$ randomly selected initial states (training set) for the non-linear model with estimated (blue) and optimised (red) hyperparameters. Performance of the optimised model on additional 25 test cases (yellow). $|x| < 10$, $|x'| < 10$, $|\theta| < \pi$, $|\theta'| < 15$.

Task 2.3 - Improved feature representation

In this task, rather than using the raw angle θ and modifying the kernel to account for periodicity, we redefine the input vector to include $\sin\theta$ and $\cos\theta$, eliminating the need for angular correction in the kernel. Figure 10 shows rollout results using the 5-state kernel model with optimised hyperparameters. Although the system still fails to keep the pendulum in its unstable position, the predicted trajectories match the 'true' dynamics longer than with the non-linear model without optimised hyperparameters (see Figure 8).While higher accuracy and longer rollouts are possible with larger N and M, this increases training and inference time.

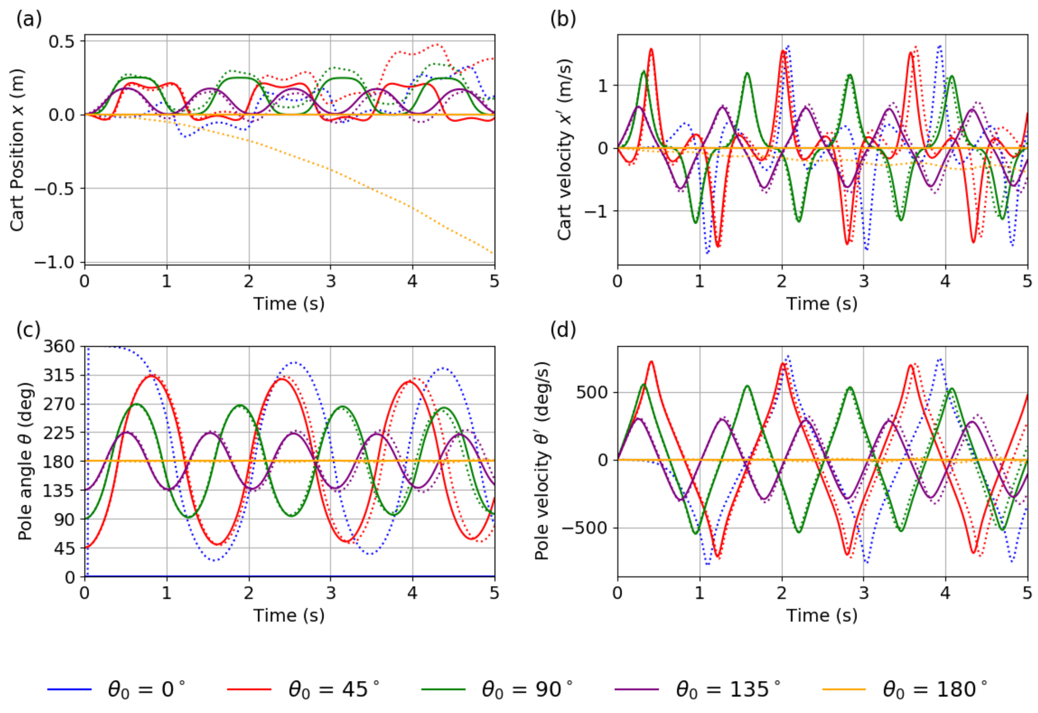


Figure 10: System dynamics predicted by the non-linear model with optimised hyperparameters (dotted lines) and the 'true' dynamicss (solid lines) for various initial positions of the pole: (a) Cart position x. (b) Cart velocity x' (c) Pole angle θ. (d) Pole velocity θ'.

Although using the standard deviation of each state variable as the initial hyperparameter σ_j leads generally to good optimised hyperparameters, this is not always the case. As shown in Figure 11, optimisation runs starting with different values result in different optimised hyperparameters, with different final MSE values. Not only that, but optimised parameters performing well on the training dataset may not do so on test datasets (Figure 11b). Therefore, when optimising hyperparameters, we consider different initial values and chose the optimised parameters that result in minimum error for both training and test sets, i.e. the hyperparameters that minimised $MSE^2_{training} + MSE^2_{test}$.

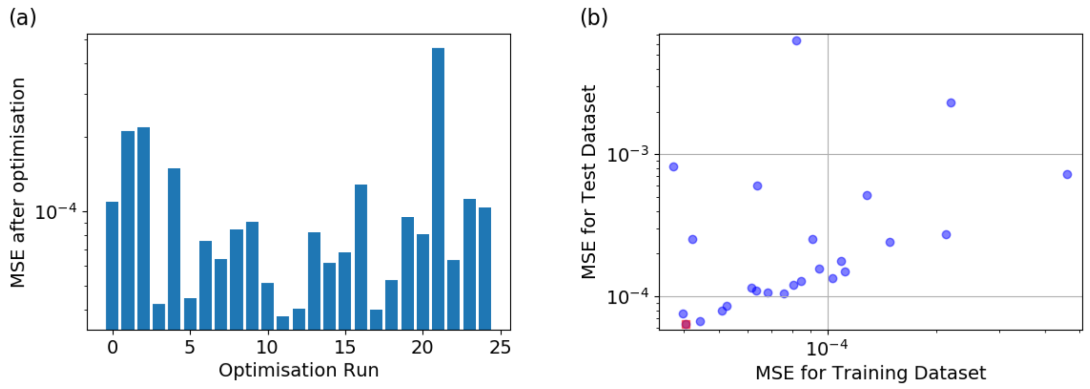


Figure 11: (a) MSE produced by the non-linear model after hyperparameter optimisation for different initial conditions. Each optimisation run leads to a set of optimised hyperparameters. (b) Performance comparison of optimised hyperparameters on training and test datasets. The red square indicates the selected set of optimum hyperparameters.

Task 3.1 - Control

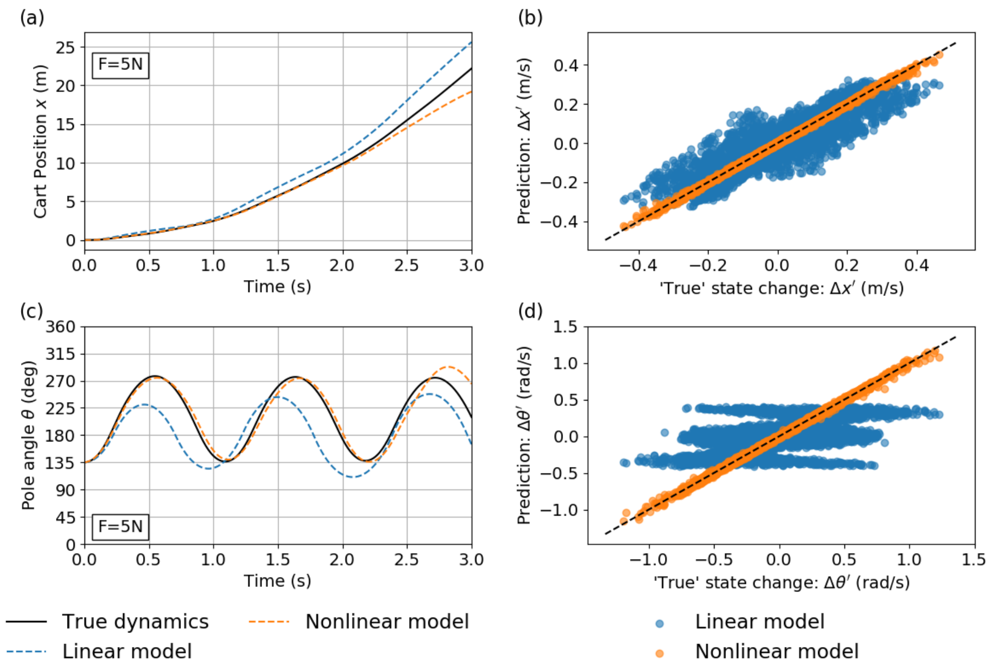


Figure 12: System dynamics predicted by the linear and nonlinear models (dotted lines) compared to the true dynamics (solid lines): (a) cart position x, (c) pole position θ for the system starting at rest and $F = 5N$. Comparison of state variable changes for the linear and non-linear models: (b) $\Delta x'$, (d) $\Delta\theta'$ for initial states in the interval: $|x| < 10$, $|x'| < 10$, $|\theta| < \pi$, $|\theta'| < 15$ and $|F| < 200$.

We now consider the effect of an external force acting on the system. The maximum force that can be modelled by *cartpole.py* depends on the time step of integration dt, which we set to 0.01s. Although not shown explicitly, for $dt = 0.01$, reasonable resolution and stability is maintained for forces in excess of 200N, although the models prove to be less accurate for large forces.

To model the effect of an external force, both linear and non-linear models were extended to include the current state and applied force as inputs. As shown in Figure 12, both models capture the system's response to a constant 5N force, with the non-linear model providing more accurate predictions of the dynamics.

However, predictions suffer as the applied force increases. Figure 13 shows how as the force increases to 10N and then 20N, the models capture the dynamics for shorter and shorter times.

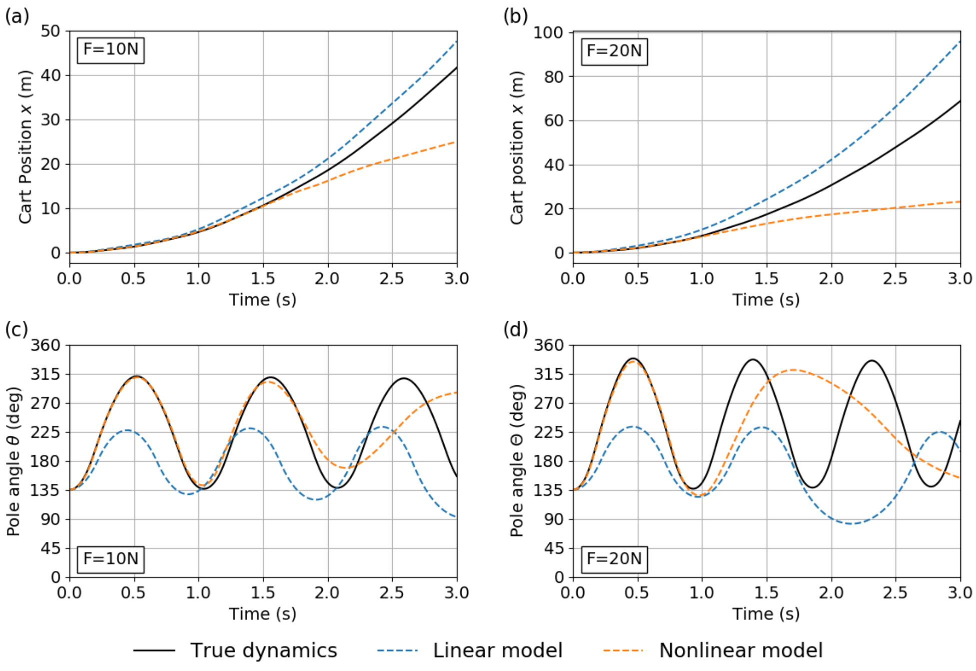


Figure 13: System dynamics predicted by the linear and non-linear models (dotted lines) and the 'true' dynamics obtained by integrating the equations of motion (solid lines): (a,b) cart position x, (c,d) pole position θ for the system starting at rest with a force applied of (a,c) $F = 10N$ and (b,d) $F = 20N$.

Task 3.2 - Policies

Having modelled the cart-pole system's response to external forces, we now focus on implementing a control strategy to balance the pole upright at its unstable equilibrium. We use a linear control policy, where the applied force $F(X)$ is a linear function of the current state: $F(X) = p \cdot X$. The policy coefficients p are chosen by optimising against a loss function that drives the system to the desired state. Here we consider the following loss function L:

$$L = \sum_{i=1}^{N} \left[1 - \exp\left(-\frac{|X - X_o|^2}{2\sigma^2} \right) \right] \tag{4}$$

where $X_0 = [0, 0, 0, 0]$ (or $[0, 0, 0, 1, 0]$ when using $\sin\theta$ and $\cos\theta$) is the target state, i.e. the cart is centred and stationary with the pole upright. The loss ranges from 0 to 1, and sensitivity to each state variable is controlled by the vector σ. Tuning σ is essential for effective control. The pole angle θ is the most critical variable, as even small deviations from upright must be quickly corrected o prevent the system from becoming unrecoverable. To ensure this, the σ associated with θ must be small, so the loss function strongly penalises angular deviations. In contrast, if σ for angular velocity θ' is too small, the controller becomes overly cautious, limiting its ability to apply strong corrective actions. The same applies to cart velocity x': over-penalising it restricts movement and hinders balance recovery, so a larger σ is preferred. Cart position x, though less critical for stability, still helps guide the system back to the target $x = 0$. Based on this, we selected $\sigma = [2, 100, 0.5, 100]$, which gives a sensible loss of 0.5 when $\theta = 2\sigma^2 \ln 2 \approx 20°$. Hyperparameter optimisation was performed using the L-BFGS-B algorithm as before.

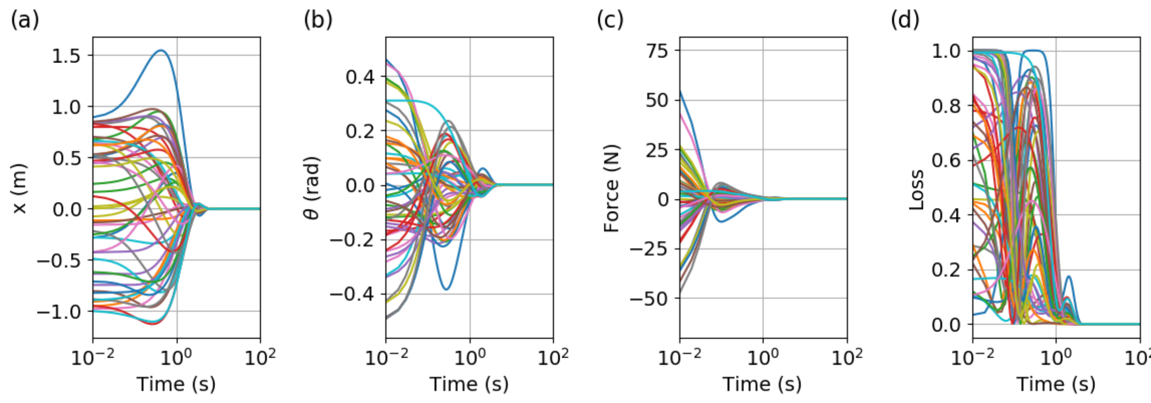


Figure 14: Time evolution of the (a) cart position, (b) pole angle, (c) applied force and (d) loss term for 50 different initial conditions of the system.

Figure 14 shows the system's behaviour for 50 different initial conditions. The controller successfully drives the system to equilibrium, with the loss function approaching zero within ~2 seconds. In some cases, the loss decreases and rises again due to the pole oscillating around the upright position before full stabilisation takes place. Because the controller prioritises correcting the pole angle over the cart's position, the cart may move away from the origin while the angle is being stabilised (e.g. the top blue trajectory in Figure 14a).

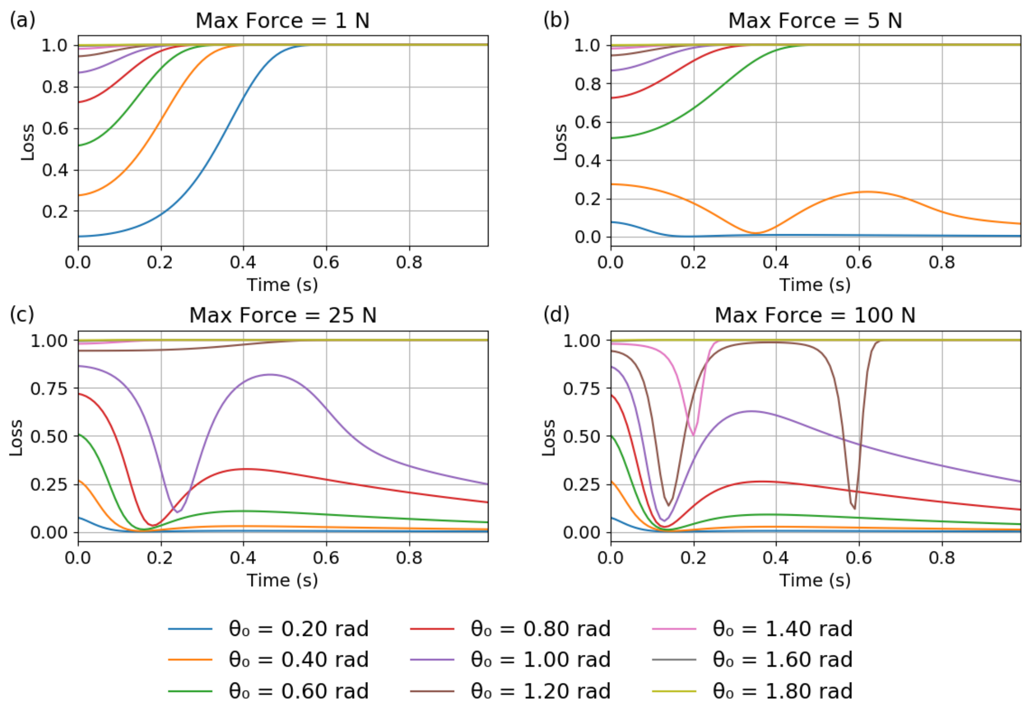


Figure 15: Effect of limiting the force acting on the system on the ability of the controller to control the system. Force limited to (a) 1N, (b) 5N, (c) 25N and (d) 100N. System started at rest with the pole at various initial angles. Same control policy applied to all cases.

As shown in Figure 14c, the controller initially applies large forces to stabilise the system. It is therefore expected that limiting the available force would impair control performance. To confirm this, the system was simulated from rest with varying initial pole angles using the same control policy but applying different force limits (see Figure 15). When the maximum force is limited to 1N, the controller fails even for small initial angles, i.e. the loss evolves rapidly to 1 for all conditions (Figure 15a). As the force limit increases, the controller successfully stabilises the system in more cases, and for a limit force of 100N, the system can balance the pole even when starting at $\theta_o \sim 1$ rad$\sim 60°$ (Figure 15d).

Task 3.3 - Model predictive control

Previously, we optimised the policy vector p by directly simulating the real system dynamics, resulting in a successful control strategy that stabilises the system relatively fast (Figure 14). In this task, we shift from using the true system dynamics to the non-linear model for policy optimisation.

First it was noted that the same policy vector found in the previous section is found to be able to bring the modelled system to the desired state. Developing a policy using the non-linear model, however, proved time-consuming as I could not get around to vectorising the objective function.

Task 4.1 - Noise in observed dynamics

Introducing noise into the cart-pole simulation helps evaluate the robustness of our models and controller under more realistic, imperfect conditions. In real-world systems, sensor measurements are rarely noise-free. To reflect this, we simulate sensor noise by applying multiplicative Gaussian noise to the input state vectors, i.e. $x \to x \cdot \left(1 + \mathcal{N}(0, \sigma^2)\right)$. This allows us to assess how the models generalise when trained and tested on noisy observations. Additive noise ($x \to x + a\mathcal{N}(0, \sigma^2)$) can be considered similarly, but that it is not explored here.

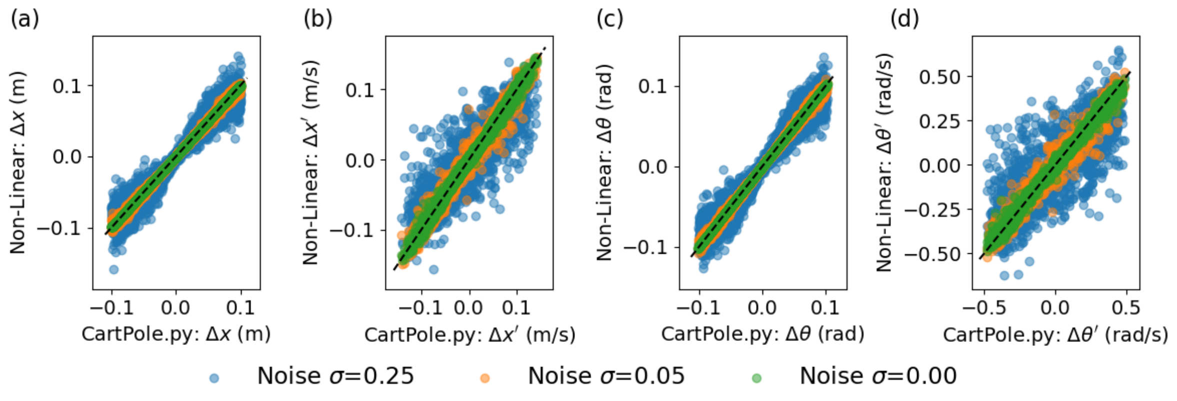


Figure 16: Effect of multiplicative Gaussian noise on state prediction in the non-linear model.

The presence of sensor noise makes the models less accurate. As shown in Figure 16 for the state change predictions of a non-linear model trained with noisy sensors, state change predictions move away from the ideal black dashed line as the noise increases. Despite this loss in accuracy, the model is able to predict to some extent the system dynamics (not shown explicitly) and the controller to balance the pole upright (Figure 17), although it does so by applying forces that change rapidly and may not be physically implementable (see Figure 17 and compare with the noise-free case in Figure 14). We did not explore the potential use of filtering to remove noise, although their use could introduce delays and affect performance.

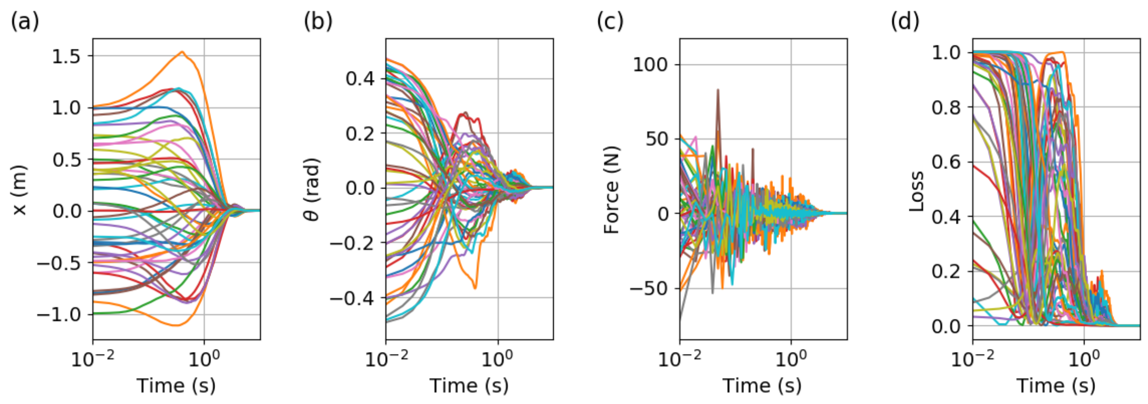


Figure 17: Rollout of the non-linear model for 50 initial conditions demonstrating the ability of the controller to balance the pole in the presence of sensor noise $\mathcal{N}(0, 0.25^2)$.

Task 4.2 - Noise in actual dynamics

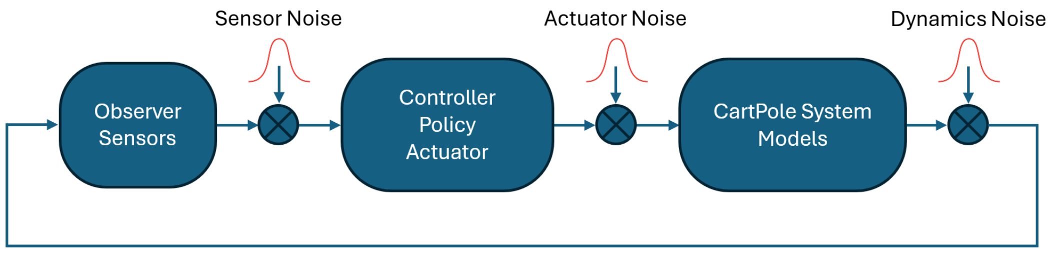


Figure 18: Sources of noise. 'Sensor noise' affecting the input to the controller, 'Actuator noise' the force applied to the CartPole, and 'Dynamics noise' the system's state.

In addition to sensor noise, other sources of noise can be affect the system (see Figure 18). These were considered individually, and it was found that for the same level of noise, i.e. same $\mathcal{N}(0, \sigma^2)$, the system was most sensitive to the 'dynamic noise', which accounts for random disturbances in the system state due to environmental factors such as wind, surface irregularities, etc. And the 'actuator noise' was found to be the one affecting the dynamics the least. Nevertheless, a linear control policy was able to balance the pole upright even in the presence of (small $\sigma \sim$0.05) noise in the sensors, the actuator and the dynamics of the system (Figure 19).

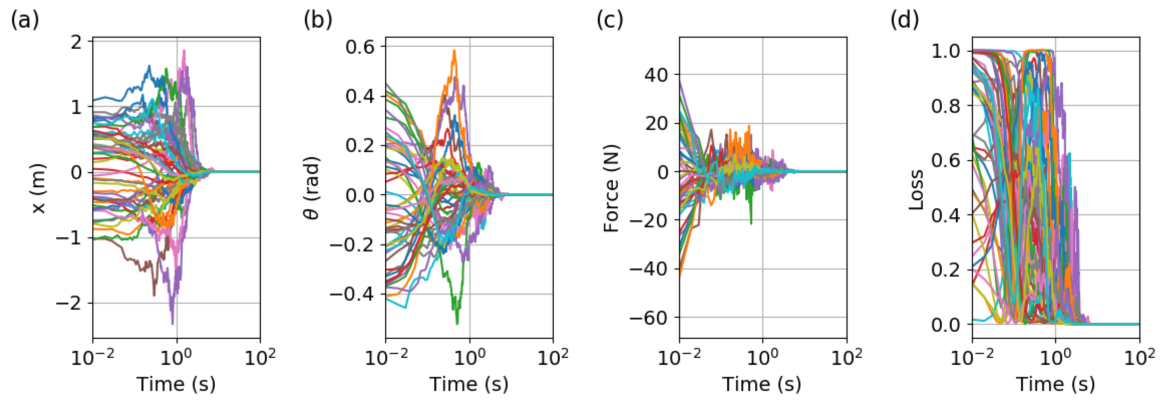


Figure 19: Rollout of the non-linear model for 50 initial conditions demonstrating the ability of the controller to balance the pole in the presence of sensor, actuator and dynamics noise: noise $\mathcal{N}(0, 0.05^2)$.