SF3: Machine Learning Final Report

Zhen Yuen Chong
Cambridge University Engineering Tripos Part IIA
St. Edmund's College
zyc24@cam.ac.uk

***Abstract*—The project aims to control a cart-pole (or cart-pendulum) system via machine-learning techniques. Initially, the various behaviours exhibited by the system are explored by running the simulator under various initial conditions. Then, a linear model is trained and evaluated against the simulated data. Subsequently, non-linear regression models, which may include linear regression with non-linear basis functions, are trained and evaluated. Once accurate models have been established, they are utilized to develop controllers that can maintain the pole in an upright position. The proposed controllers and models must undergo rigorous testing in different scenarios to assess their resilience.**

I. Introduction

In this report, the process of exploring and collecting the data on the cart-pole (or cart-pendulum) system via running simulations under various initial conditions is documented, in addition to fitting and evaluating a linear and non-linear model to the system to predict changes in state trajectories. Then, we design linear a policy capable of maintaining the pole upright and analyse the effect of introducing observation and disturbance noise into the system. Finally, we attempt to design a controller using a non-linear policy that can swing the pole upright from its stable equilibrium and maintain it upright.

II. Preliminaries

A. Theory

In this report, the state vector is denoted as $\mathbf{x} = [x, \dot{x}, \theta, \dot{\theta}]$ such that x is the cart position, $\dot{x}$ is the cart velocity, θ is the pole angle and $\dot{\theta}$ is the pole velocity for conciseness. Each call to `perform_action()` causes the simulation to run for intervals of 0.2 seconds carried out in 50 Euler integration steps and returns the resulting state. This results in jumps/discontinuities in the observed state-space trajectories. Rather than decreasing the number of integration steps or interval size for each function call to obtain smoother plots, the decision was made to keep the simulation configuration unchanged. This is justified to replicate the data collection process in practice and real-world scenarios whereby measurements cannot be collected continuously but rather sampled at some fixed rate.

The equations of motion for the physical system are given in [1] as:

$$\begin{aligned} 3\ddot{x}\cos\theta + 2L\ddot{\theta} &= 3g\sin\theta - \frac{6\mu_\theta\dot{\theta}}{mL} \\ (m+M)\ddot{x} + \frac{1}{2}mL\ddot{\theta}\cos\theta - \frac{1}{2}mL\dot{\theta}^2\sin\theta &= F - \mu_x\dot{x} \end{aligned} \tag{1}$$

The equations of motion can be linearised around its stable equilibrium, $\mathbf{x} = [0, 0, \pi, 0]$, to the following:

$$\begin{aligned} 3\ddot{x} + 2L\ddot{\theta} &= 3g\theta - \frac{6\mu_\theta\dot{\theta}}{mL} \\ (m+M)\ddot{x} + \frac{1}{2}mL\ddot{\theta} &= F - \mu_x\dot{x} \end{aligned} \tag{2}$$

For a stationary cart and small angular velocities, if μ_θ/mL is small, we have:

$$\ddot{\theta} \approx \frac{3g}{2L}\theta \tag{3}$$

This implies that the restoring torque is linearly proportional to the angular displacement from the stable equilibrium. Note that the theoretical results above are used to sanity-check the data collected from the simulation. In practice, however, the equations of motion are usually much more complex or not available, thus more caution should be exercised during data collection.

B. Accelerating numerical computation

In later sections of this project, a significant amount of computation is required to fit the model and controller. To speed up the numerical computations required, I modified the original remapping function to use modulo operations instead of while loops. In addition, functions involving matrix multiplications with large dimensions such as computing the kernels for the non-linear model are compiled with the just-in-time (JIT) compiler from Numba. This allows each function call to be cached and avoids redundant computations by returning the desired result for a previously seen input.

III. Task 1.1

The behaviour of the cart-pole system is investigated by varying the initial value for a single state variable and running the simulation while keeping other state variables constant. By varying $x \in [-10, 10]$, the behaviour of the system remains unchanged as seen in Fig. 1, implying the system is translation invariant. This is consistent with the equations of motion having no dependence on x. Hence, in subsequent experiments, we fix the initial cart position, $x_0 = 0$. Unless varied or specified, the initial states of the system are set and fixed to the stable equilibrium $\mathbf{x}_0 = [x, \dot{x}, \theta, \dot{\theta}] = [0, 0, \pi, 0]$ respectively.

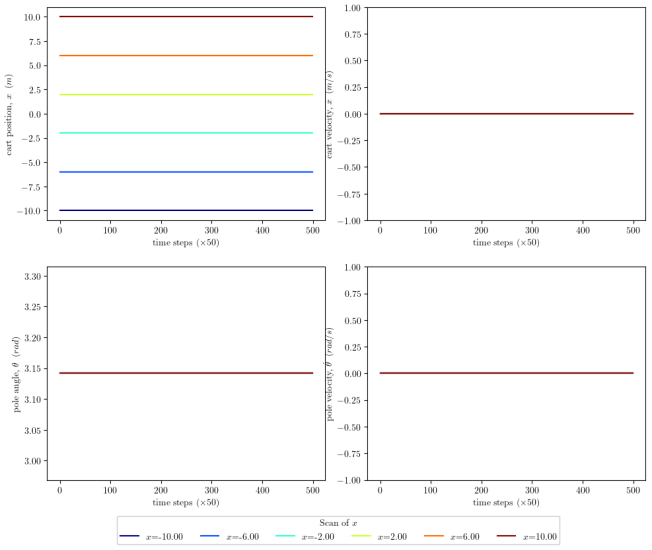

Fig. 1. Time response to cart position.

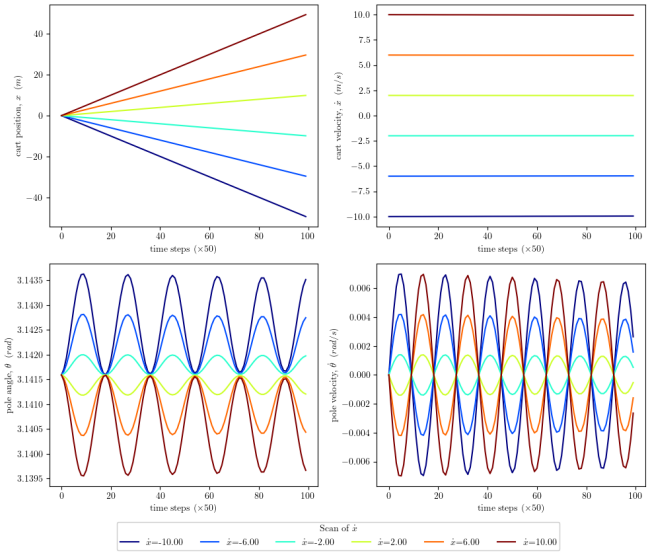

Fig. 2. Time response to cart velocity.

A. Simple oscillations

Fig. 2 shows the time response as the initial cart velocity is varied, $\dot{x}_0 \in [-10, 10]$ (values shown in legend). The θ always oscillates about the stable equilibrium, with the amplitude of oscillations being proportional to $\dot{x}_0$. When $|\dot{x}_0| = 15$, $|\theta|$ still does not exceed 2π. Hence, complete oscillations cannot be achieved via varying the $\dot{x}_0$ alone. Fig. 3 shows that when the system is perturbed, it eventually returns towards the stable equilibrium, $[\dot{\theta}, \theta] = [0, \pi]$, across time (purple to yellow) due to energy dissipation. In addition, the state-space trajectory of $\dot{\theta}$ against $\dot{x}$ is approximately linear which satisfies the small-angle approximation in (2) as $\ddot{x} \propto \dot{\theta}$ (assuming $3g\theta \ll \frac{6\mu_\theta \dot{\theta}}{mL}$ for small angular displacements and $\ddot{\theta} \approx 0$) and $\ddot{x} \propto \dot{x}$ (since $F = 0$), which implies $\dot{\theta} \propto \dot{x}$.

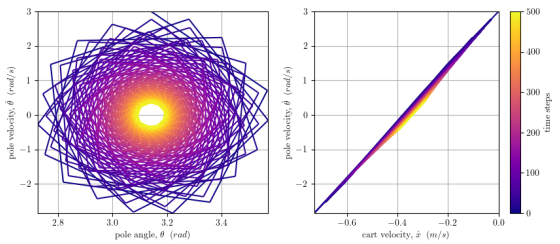

Fig. 3. State space trajectories of pole velocity against pole angle (left) and cart velocity (right).

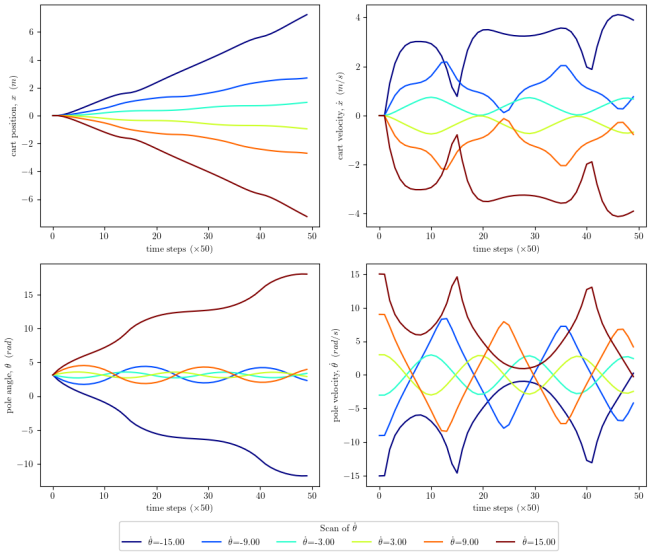

Fig. 4. time response to pole velocity.

B. Complete oscillations

Fig. 4 shows the time response as the initial pole velocity is varied, $\dot{\theta}_0 \in [-15, 15]$ (values shown in legend). The θ always oscillates about the stable equilibrium, with the amplitude of oscillations being proportional to $\dot{\theta}_0$. The small-angle approximation theory no longer applies as evident in the non-linear plots. We observe that the system performs complete oscillations only under sufficiently large initial angular velocities. The exact threshold is found to be $\dot{\theta} \approx \pm 14$ (the exact value is unknown but should be identifiable by repeating the same experiment in smaller increments/decrements of $\dot{\theta}$). Fig. 5. shows a plot of the state-space trajectories of the system initialized from $\mathbf{x} = [0, 0, \pi, 14]$ to meet complete oscillation conditions with angle remapping enabled. The plot shows that the pole undergoes complete oscillations but eventually returns towards the stable equilibrium, $[\dot{\theta}, \theta] = [0, \pi]$ across time (purple to yellow) due to energy dissipation. Hence, the small-angle approximation only holds once the oscillation has decayed sufficiently.

IV. Task 1.2

The system is initialized with a random value for all state variables, $\mathbf{x}_0 = [0.96, -5.85, 2.06, -10.52]$ and the

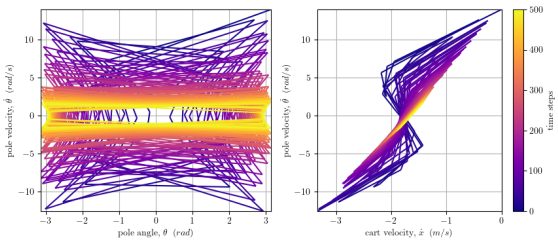

Fig. 5. State space trajectories of pole velocity against pole angle (left) and cart velocity (right).

final state $\mathbf{y}$ is recorded after a single time step (one call to `perform_action()`). From Fig. 6, the relationship between $\mathbf{x}$ and $\mathbf{y}$ is approximately linear as expected because the change in a single time step is small. By accounting for this and modelling the change in the states instead, we define a new target for the modelling as $\mathbf{y} = \mathbf{x}_T - \mathbf{x}_0$, where $\mathbf{x}_T$ represents the time evolution of the states under the dynamics, and T corresponds to a single run of the simulation. Note that the longer the time shift used (by increasing T), the model becomes increasingly complex and the linear relationship no longer holds. The relationship between $\mathbf{x}_0$ and $\mathbf{y}$ is shown in Fig. 7. We observe that x_0 does not affect $\mathbf{y}$. Furthermore, $\dot{x}_0$ does not affect $\mathbf{y}$, except for $y_0 = \Delta x$. This is because $\theta, \dot{\theta}$ is independent of the inertial frame of reference chosen (in this case, the xy-coordinate reference frame is chosen, but the moving cart can also be chosen as the reference frame).

When released from the stable equilibrium, θ_0 does not appear to have a significant effect (relative to varying the $\dot{\theta}_0$) on $\mathbf{y}$. It is observed that varying $\dot{\theta}_0$ leads to the largest $\mathbf{y}$, in addition to the system performing complete oscillations when $|\dot{\theta}_0| > 8$ (approx). This is expected as we found the complete oscillation threshold to be about $|\dot{\theta}_0| = 14$ when starting from the equilibrium point, $\theta = \pi$, so the threshold should be lower in this scenario as $\theta \neq \pi$ (the initial state is "further" from the stable equilibrium).

Contour plots shown in Fig. 8, 9, 10 are used to visualize the effect of varying initial values of two state variables and the resulting change in the other remaining state variable. Given that the system is translation invariant, we omit the effect of varying cart position in our contour plots. Consistent with our observations above, Fig. 8 and 9 shows that $\mathbf{y}$ (with the exception of $y_0 = \Delta x$) is independent of $\dot{x}$, while it is evident From Fig. 9 that $\mathbf{y}$ is some non-linear function of θ_0 and $\dot{\theta}_0$. This tells us that a linear model should perform poorly when fitted to the data. Upon closer inspection, $\Delta\theta$ appears to be approximately linear with respect to the $\dot{\theta}_0$, with a some dependence on θ_0 due to gravitational forces acting on the pole.

V. Task 1.3

In this section, we attempt to fit a linear model on the system. 500 data points are generated by running the simulator

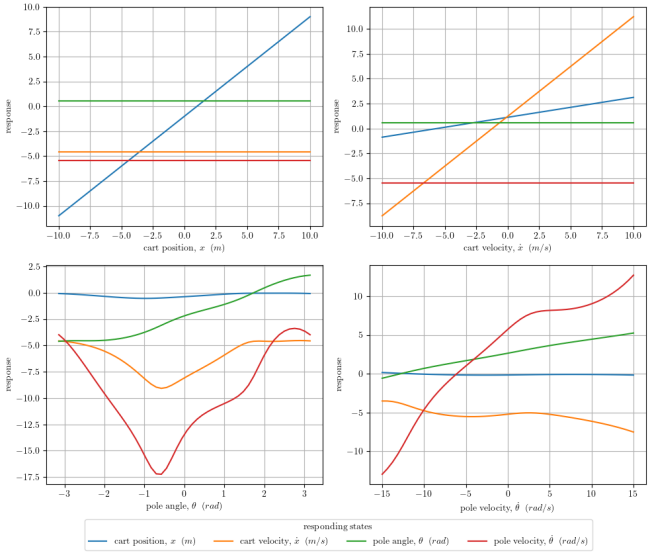

Fig. 6. Resulting states against cart position (top-left), cart velocity (top-right), pole angle (bottom-left) and pole velocity (bottom-right) after one-time step.

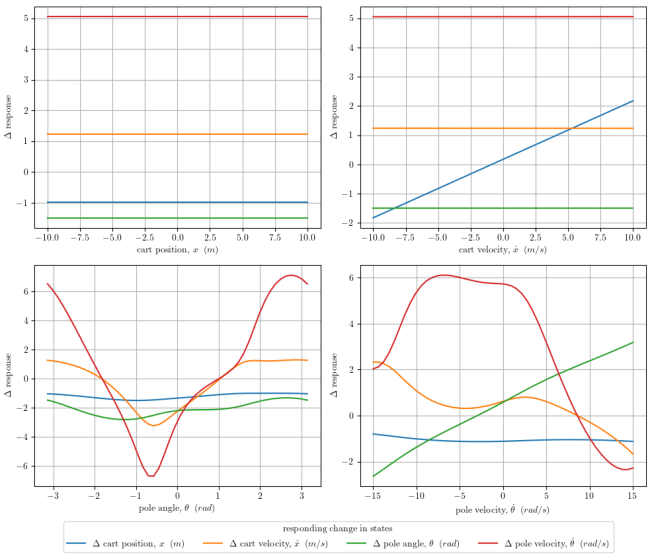

Fig. 7. Change in states against cart position (top-left), cart velocity (top-right), pole angle (bottom-left) and pole velocity (bottom-right) after one-time step.

with random initial states within suitable ranges as defined in (4),

$$\mathbf{X}_n = \{(x, \dot{x}, \theta, \dot{\theta}) : |x| \leq 10, |\dot{x}| \leq 10, |\theta| \leq \pi, |\dot{\theta}| \leq 15\} \tag{4}$$

and ran for a single time step (with zero force) to get the change in the state vector, $\mathbf{Y} = \mathbf{X}_{n+1} - \mathbf{X}_n$ respectively. The paired data, $(\mathbf{X}, \mathbf{Y})$, were then used to fit a linear regression model. The optimal coefficient matrix (least squares fit) can be obtained by (5).

$$\begin{aligned} \mathbf{Y} &= \mathbf{X}\mathbf{W} \\ \mathbf{W} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \end{aligned} \tag{5}$$

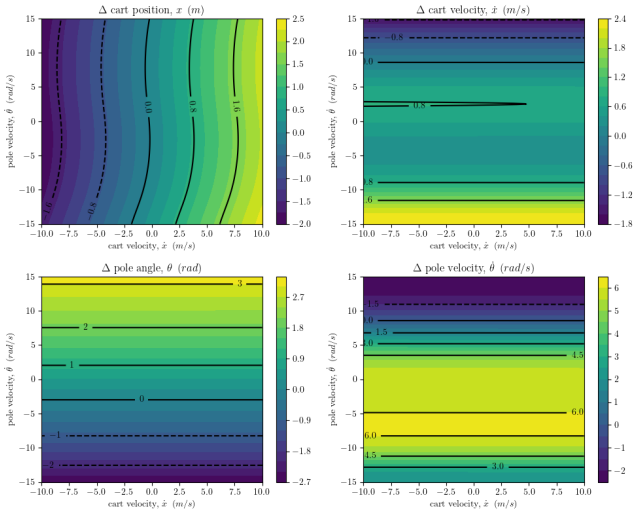

Fig. 8. Change in states against pole velocity (y-axis) and cart velocity (x-axis).

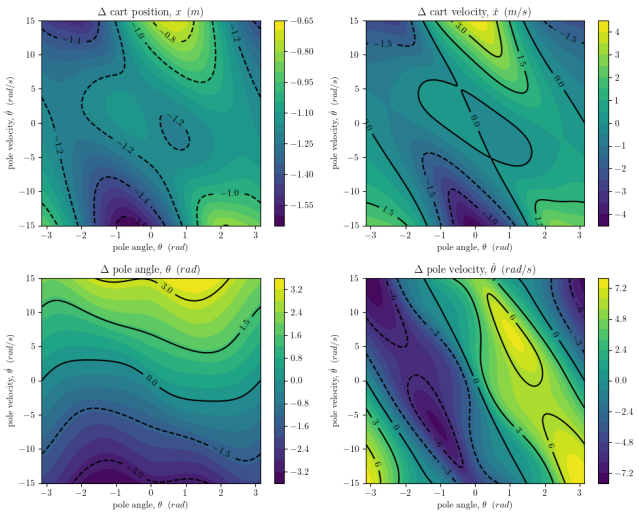

Fig. 9. Change in states against pole velocity (y-axis) and pole angle (x-axis).

With the fitted model, the predicted and actual final states after a single time step (with zero force), denoted as $\hat{\mathbf{X}}_{\mathbf{n+1}}$ and $\mathbf{X}_{\mathbf{n+1}}$ respectively, were plotted against the initial states $\mathbf{X}_{\mathbf{n}}$ as shown in Fig. 11. The predicted values for x and θ are relatively close to their actual values, but the first-order derivatives $\dot{x}$ and $\dot{\theta}$ deviate significantly from their actual values. The root-mean-squared error is calculated by (6) to be $[0.7295, 2.4916, 1.3234, 4.5511]$.

$$\sqrt{\frac{(\mathbf{Y}-\mathbf{X}\mathbf{W})^T(\mathbf{Y}-\mathbf{X}\mathbf{W})}{N}} \quad \text{as} \quad N = 500 \tag{6}$$

Fig. 12 shows a plot of the predicted values, $\hat{\mathbf{X}}_{\mathbf{n+1}}$ against the actual final state values, $\mathbf{X}_{\mathbf{n+1}}$, with black dotted lines representing a perfect linear fit. As expected, an approximately linear curve is observed for x and θ but not for $\dot{x}$ and $\dot{\theta}$. Contour plots corresponding to the predicted change in states are

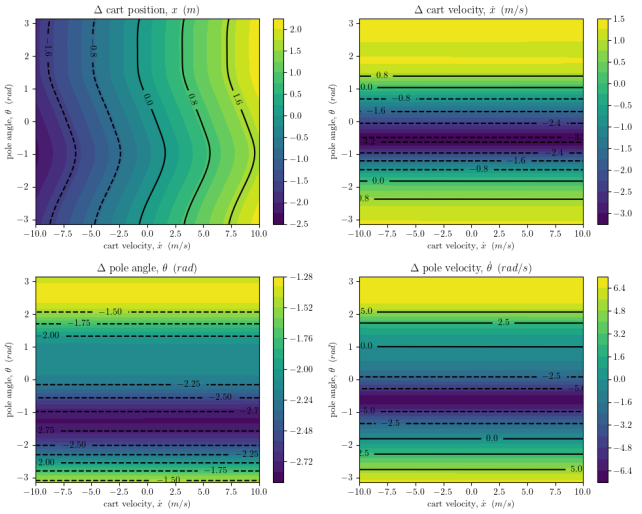

Fig. 10. Change in states against pole angle (y-axis) and cart velocity (x-axis).

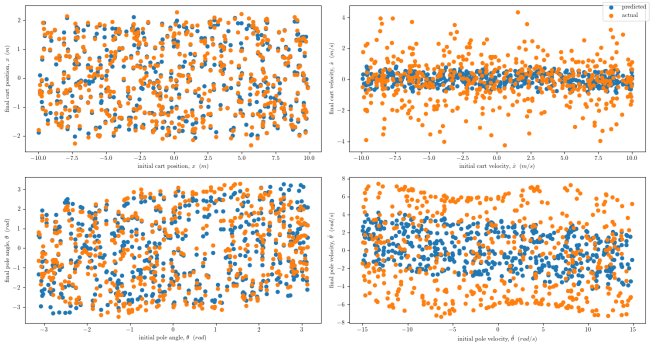

Fig. 11. Predicted and actual final states after a single time step (with zero force) against the initial states.

shown in Fig. 13, 14, 15 in which the white contour lines indicate the original contours of the actual system, while the black contour lines correspond to the linear decision boundaries of the model. Note that, if small oscillations about the stable equilibrium are only being considered, a linear model performs well as shown in Fig. 16. The root-mean-squared error, in this case, is calculated to be $[0.0270, 0.0755, 0.0762, 0.2162]$, which is much smaller. In theory, it is possible to model the system with an ensemble of linear models fitted around various points with small increments from one point to another but is not practical. The predicted and actual time response of the system is shown in Fig. 13.

VI. Task 1.4

In the previous analysis, the model fits well for x and θ for a single time step with zero force. However, when the linear model is used to predict the time evolution of the physical system, the predicted behaviour deviates from the actual behaviour for both simple and complete oscillation conditions given a sufficiently long time horizon as shown in Fig. 18 and 20 respectively. In the those plots, the state-space trajectories are coloured from purple to yellow (the colour

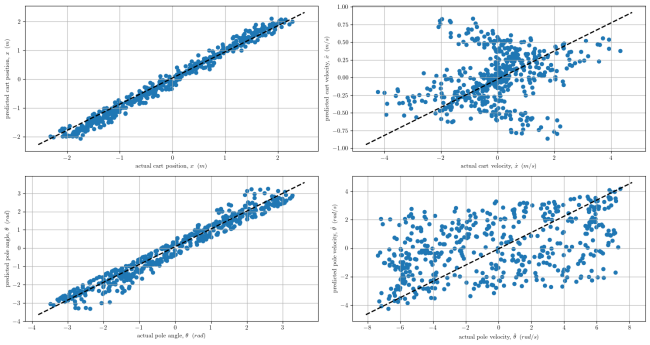

Fig. 12. Predicted states against actual states after a single time step (with zero force).

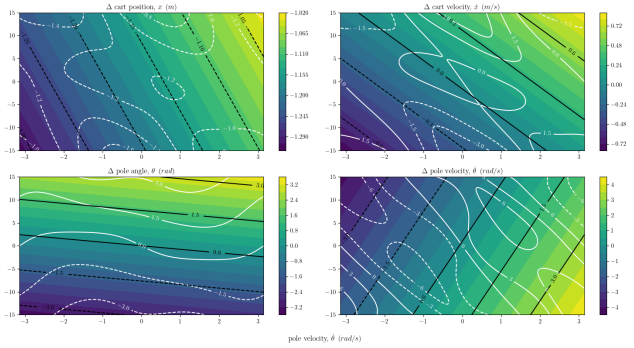

Fig. 13. Change in states against pole velocity (y-axis) and cart velocity (x-axis).

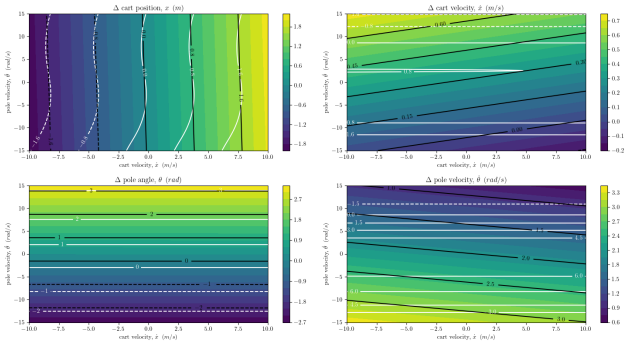

Fig. 14. Change in states against pole velocity (y-axis) and pole angle (x-axis).

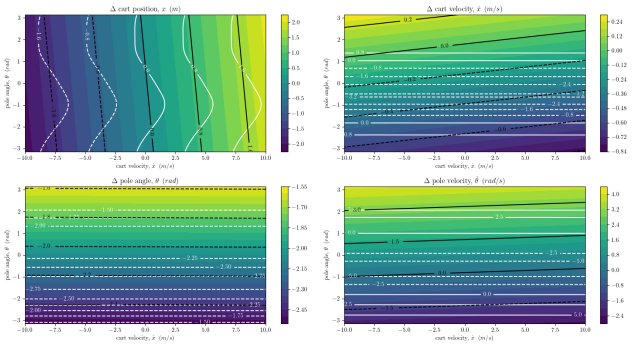

Fig. 15. Change in states against pole angle (y-axis) and cart velocity (x-axis).

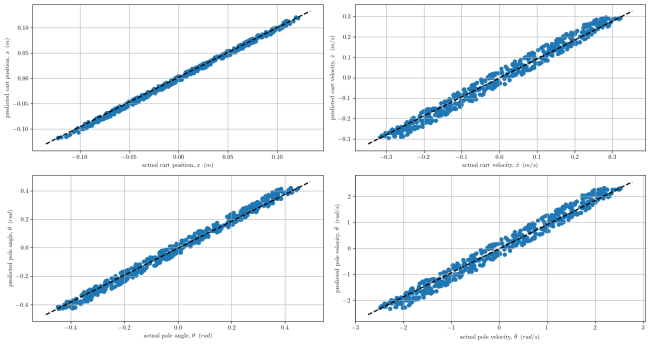

Fig. 16. Predicted states against actual states after a single time step (with zero force).

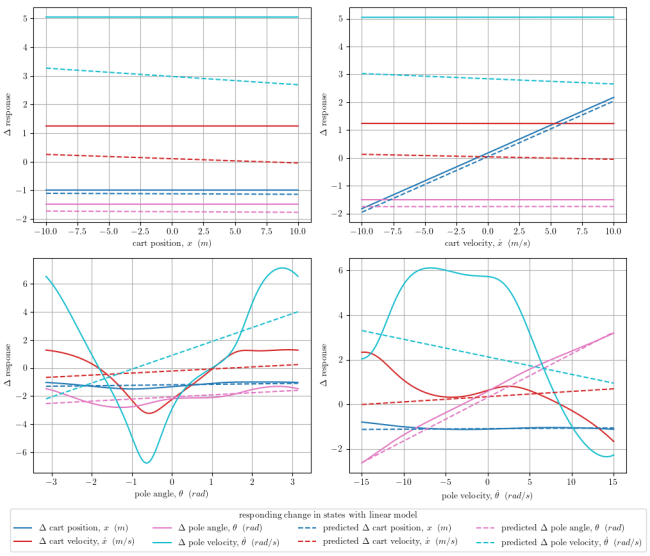

Fig. 17. Predicted and actual change in states against cart position (top-left), cart velocity (top-right), pole angle (bottom-left) and pole velocity (bottom-right) after one-time step.

gradient represents the time evolution of the system) for their predicted values and blue to yellow for their actual values. In addition, the experiments below are repeated with multiple different initial states which resulted in different state space trajectories, implying that the linear model is highly sensitive to initial conditions.

Note that, the angle has to be remapped such that $\theta \in [-\pi, \pi]$ to ensure the iterated model does not diverge. This is because (7) implies that the predicted pole angle diverges since $(\mathbf{W} + \mathbf{I})$ has a spectral radius greater than 1, which in turn causes the other predicted states to diverge too.

$$\mathbf{X}_{n+k} = (\mathbf{W} + \mathbf{I})^k \mathbf{X}_n \to \infty \quad \text{as} \quad k \to \infty \tag{7}$$

A. *Simple oscillations*

From Fig. 18, the state space trajectory of $\dot{\theta}$ against θ (with remapping) shows that the predicted oscillation amplitude is larger than that of the actual system and appears to oscillate about the stable equilibrium as with the actual system. In

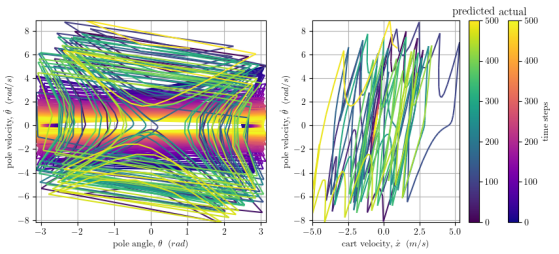

Fig. 18. Predicted and actual state trajectories for pole velocity against pole angle (left) and cart velocity (right) for simple oscillations.

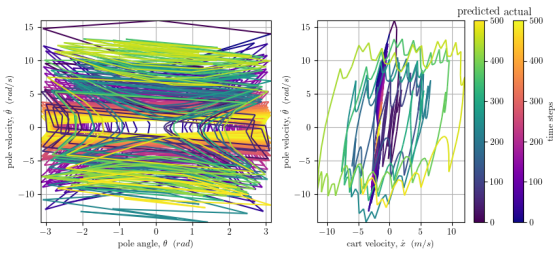

Fig. 20. Predicted and actual state trajectories for pole velocity against pole angle (left) and cart velocity (right) for complete oscillations.

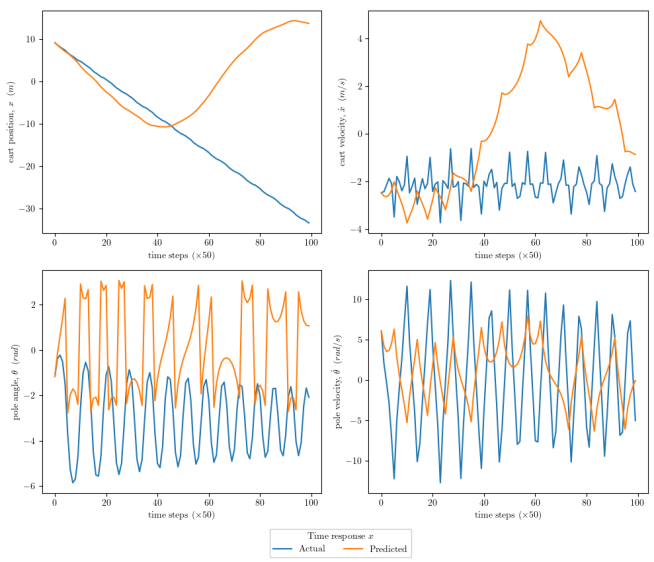

Fig. 19. Time response for complete oscillations.

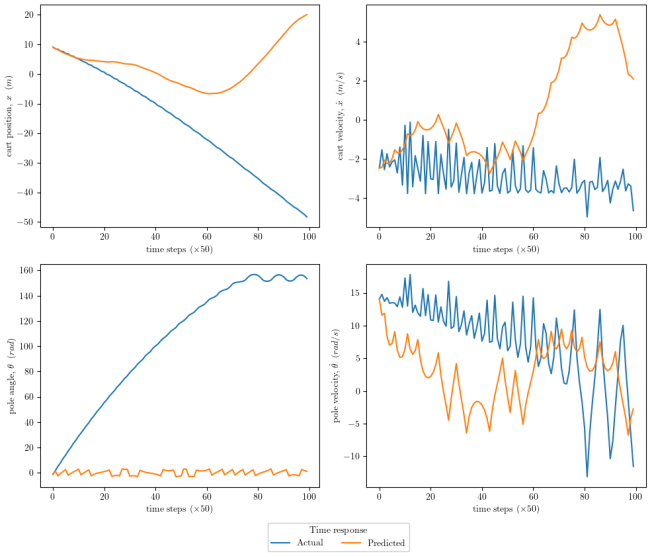

Fig. 21. Time response for complete oscillations.

addition, we see that the predicted state trajectory of $\dot{\theta}$ against $\dot{x}$ moves randomly around the origin.

B. Complete oscillations

From Fig. 20, the state space trajectory of $\dot{\theta}$ against θ (with remapping) shows that the system converges and bounces between two equilibrium points, both of which are different from the stable equilibrium point in which the actual system converges as $k \to \infty$. In addition, the plot of $\dot{\theta}$ against $\dot{x}$ shows that the predicted state trajectory diverges (spirals out) from the initial state $(\dot{x}, \dot{\theta}) = (0, 0)$. This implies that the linear model is unstable and is unable to obey the energy conservation law.

VII. Task 2.1

In this section, the non-linear model is used to predict the change in actual dynamics for the cart pole system. The non-linear model is the weighted sum of Gaussian radial basis functions (RBF) as defined in (8). To account for the periodicity in the pole angle, the difference in pole angle is computed by $\sin^2((\theta - \theta')/2)$ in place of $(\theta - \theta')^2$ in the exponent of the kernel function. σ_i is the length scale of the Gaussian RBF kernels.

$$f(X) = \sum_i \alpha_i K(X, X_i)$$
$$K(X, X') = e^{-\sum_{j=0,1,3} \frac{\left(X^{(j)} - X'^{(j)}\right)^2}{2\sigma_j^2} + \frac{\sin^2((\theta - \theta')/2)}{2\sigma_\theta^2}} \tag{8}$$

A. Sobol sequences and random sampling

The two choices for generating training data are either random sampling or using Sobol sequences. To investigate the differences between both methods, we plotted the cart velocity and pole velocity in a scatter plot shown in Figure 22. The random sampling method results in a state space with sparse sub-regions while the Sobol sequence has better coverage over entire the state space. Our hypothesis is verified by plotting a histogram showing the distribution of states in Figure 23. As evident from the histogram, the data points from the Sobol sequence are more evenly distributed. As a result, there are fewer chances for the model fitted with the Sobol sequence to encounter sparse sub-regions and perform poorly in those regions. Hence, the Sobol sequence is chosen over random sampling for subsequent experiments. The location of basis points (or equivalently, the centres for the RBFs) corresponds

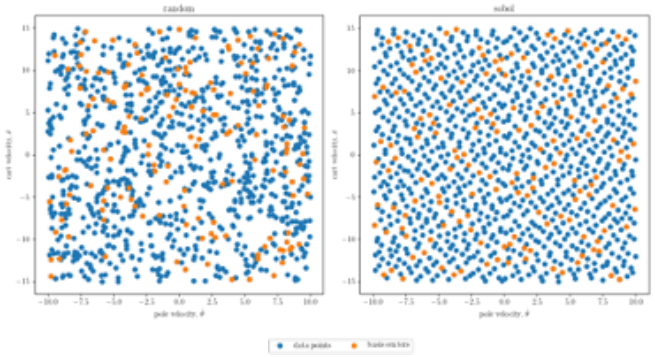

Fig. 22. Scatter plot showing locations of data points (blue) and basis points (orange) in the 2D plane ($\dot{x}$, $\dot{\theta}$) for random sampling and Sobol sequence.

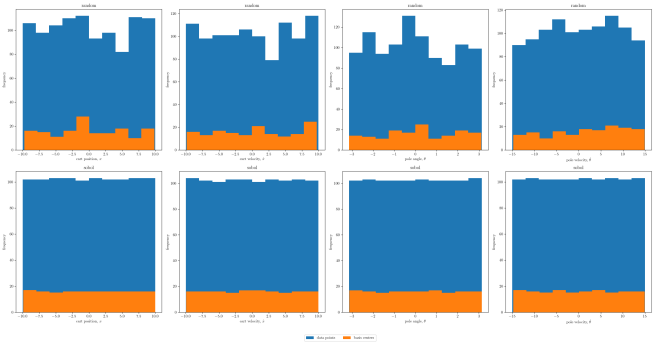

Fig. 23. Histogram showing the distribution of states for random sampling (top) and Sobol sequence (bottom).

to a subset of the generated data points highlighted in orange in Figure 22.

B. Tuning hyper-parameters

The root-mean-squared error (RMSE) of the model with respect to the number of data points used was investigated while fixing the number of basis points to 320. The RMSE appears to decrease linearly with the number of data points used. This is expected as the model can learn more information about the state space with more data points. However, rather than using all the data points for fitting, we experimented with applying an 80:20 train test split, i.e., using only 80% of generated data points to fit the model and the remaining 20% to test the fitted model. In doing so, the RMSE on the testing data shows diminishing returns as the number of data points is increased as shown in Figure 24.

Next, the RMSE of the model with respect to the number of basis points used was investigated while fixing the number of data points to 8192. Likewise, the RMSE appears to decrease linearly with the number of basis points and shows diminishing returns when the 80:20 train test split is applied. Hence, we chose 16384 data points and 5120 basis points to balance the trade-off between model accuracy and the computational resources required by the model in subsequent experiments, unless stated explicitly. Note that if all data points are selected as basis points, the model is equivalent to computing the

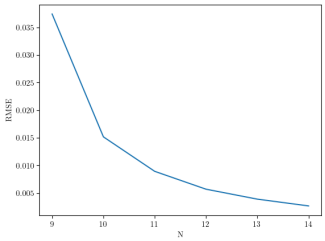

Fig. 24. Model RMSE against the number of data points.

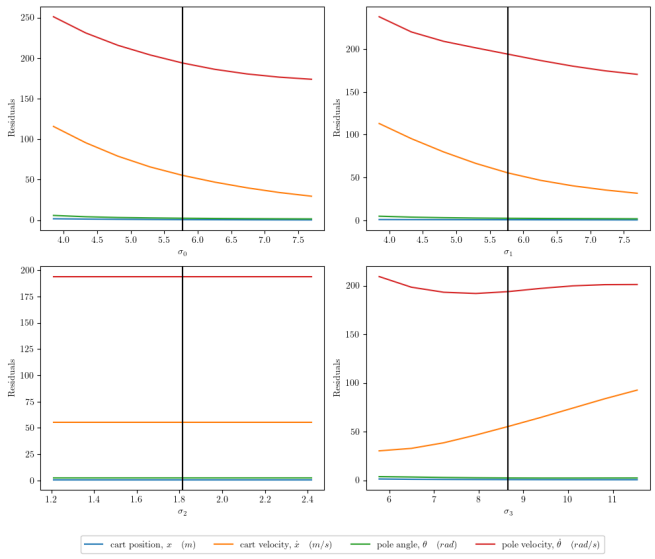

Fig. 25. Model residuals against σ_i.

mean of a Gaussian process with $K(X, X_i)$ representing the covariance matrix.

Lastly, the effect of varying the length scale σ_i on the model accuracy was investigated by considering the resulting model residuals over 1D scans for each σ_i as shown in Figure 25. The black vertical lines represent the initial estimates that are set to the standard deviation of states in the training data. Clearly, the values of σ_i used have a noticeable difference in the model's residuals for the predicted change in cart velocity and pole velocity. We find that σ_2 has no effect on the resulting residuals. Furthermore, the model appears to improve if σ_0, σ_1 are decreased and σ_3 is increased. Intuitively, the smaller the σ_i, the greater the influence of that state on the model output and vice-versa. Although the initial estimates of σ_i are not optimal, subsequent experiments would be carried out using them unless stated explicitly due to time constraints.

C. Evaluating the non-linear model

From the scatter plot of predicted against actual change in states in Figure 26, the non-linear model shows significant improvement over the linear model for the cart and pole velocities. The time response of the actual and predicted states for various initial states near the upright pole position is presented in Figure 27. It can be observed that the non-linear

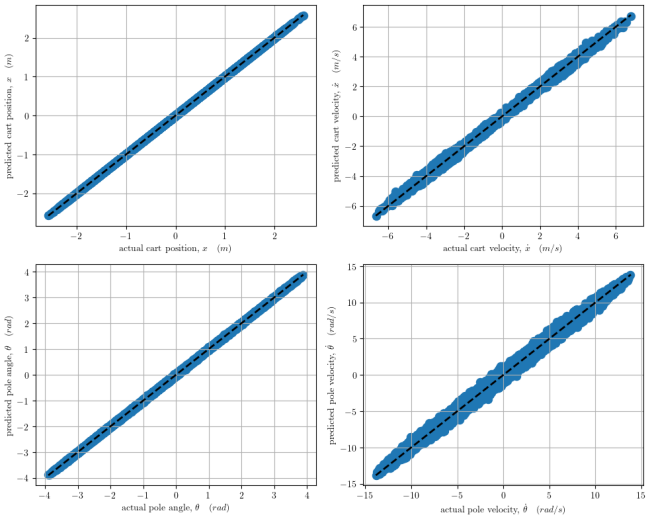

Fig. 26. Scatter plot of predicted against actual change in states.

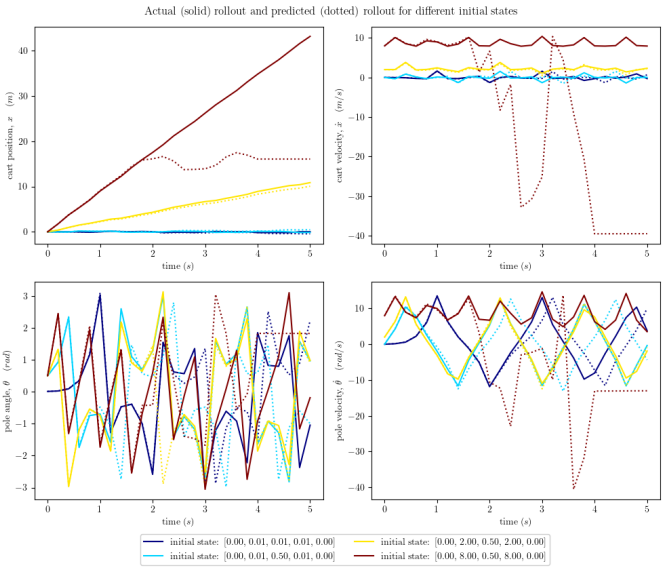

Fig. 27. Time response of system from various initial states

model tracks the actual dynamics well for smaller cart and pole velocities. When considering the stable equilibrium point as the starting state, similar observations can be obtained – the greater the velocities, the faster the model diverges. In addition, once the model diverges, it never converges back to the actual dynamics which is a consequence of the errors accumulating in the model. For most states, once a single state variable diverges from its actual trajectory, the remaining states would diverge immediately too. Overall, the model tracks the actual dynamics well for up to 2.5 seconds depending on the initial state. When released from the stable equilibrium, $(0, 0, 0.01, 0)$, the model tracks the pole angle well up to 2.4 seconds or about 2 complete oscillations.

As a side note, the remapping function was modified to include a small tolerance, i.e., $\varepsilon = 0.02$, so that the pole angle is remapped only when $|\theta| > pi + \varepsilon$. This is because small perturbations or noise about the stable equilibrium cause the pole angle to oscillate between $-pi$ and pi in the original definition. Although this remapping does not affect the actual dynamics, the abrupt change may cause difficulties when designing linear controllers using model predictive control in later sections.

One noticeable artefact of the model from the time response is the tendency of the model to diverge if the states exceed the state space defined in (4). Intuitively, the basis points act as points of reference for the model that are evenly distributed in the aforementioned state space. If the number density of basis points within a given region is high, the model should be able to track the state trajectories well in that region. This is consistent with the previous finding that the RMSE decreases with the number of basis points used. Hence, the model begins to diverge from the actual dynamics if it strays too far from the basis points. Once a state reaches a sufficiently large magnitude, that state saturates at some maximum value indefinitely. This is because the exponent of the kernel function becomes increasingly negative and tends towards zero, resulting in the predicted change of that state going to zero.

We proceeded to perform 2D scans of the states to see how the modelled dynamics differ from the actual dynamics across a time period $T = 1, 2, 3$ seconds respectively. For $T = 3$, the model already diverges by quite a significant margin from the actual system. For $T = 2$, the state trajectories remain largely similar, with the greatest differences being obtained when varying the pole velocity and cart velocity as shown in Figure 28. From the contour plots, we identified regions in which the model tracks the actual dynamics well after $T = 2$. This may help in reducing the feasible search space when optimizing the linear policy using model predictive control. These ranges are $|x| < 2, |\dot{x}| < 2.5, |\theta| < \pi, |\dot{\theta}| < 5$.

D. Further thoughts on the non-linear model

From (1), we know the system is translation invariant. Hence, the cart position should not influence the predicted change in state for an optimal model. However, this requires the exponent of the kernel function corresponding to the cart position, $\left(x^{(0)} - x^{'(0)}\right)^2 / 2\sigma_0^2$ to evaluate to zero. This is clearly not possible unless only one basis point is used, and the cart position always coincides with that of the basis centre. Hence, errors in cart position are unavoidable regardless of the model size used and compounds when simulating the model over long time horizons, usually resulting in the predicted cart position exploding. From a mathematical perspective, this should be avoidable by using some arbitrarily large value for the length scale σ_0. However, in doing so, the model may struggle to fit changes in the cart position. Hence, it may be advisable to train the model using default values of length scales from the standard deviations, then only changing σ_0 for inferencing. Alternatively, we can clip the cart position to some smaller range when predicting the state trajectories

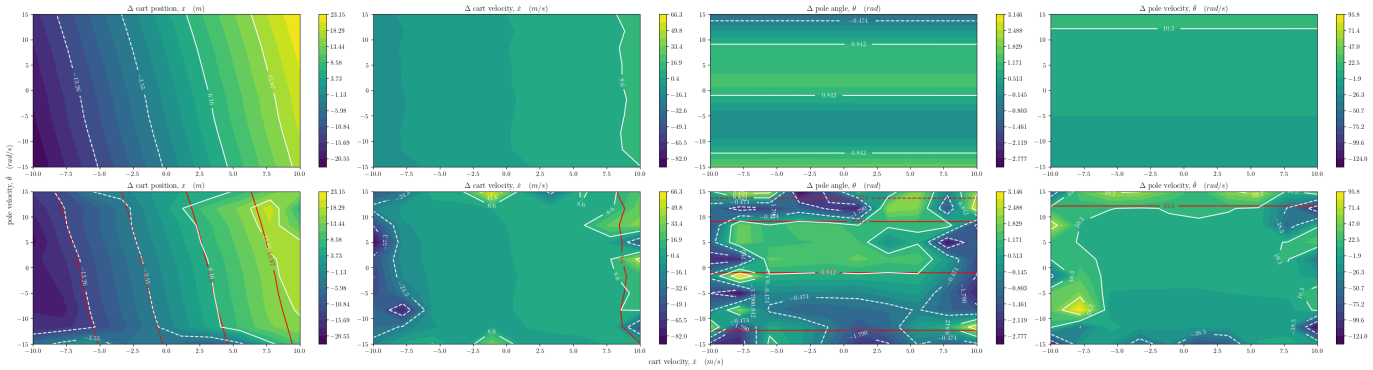

Fig. 28. Contour plots showing resulting states for actual (top) and predicted (bottom) dynamics for 2D scans of cart velocity and pole velocity after $T = 2$ seconds. Red lines correspond to contour lines from the actual dynamics.

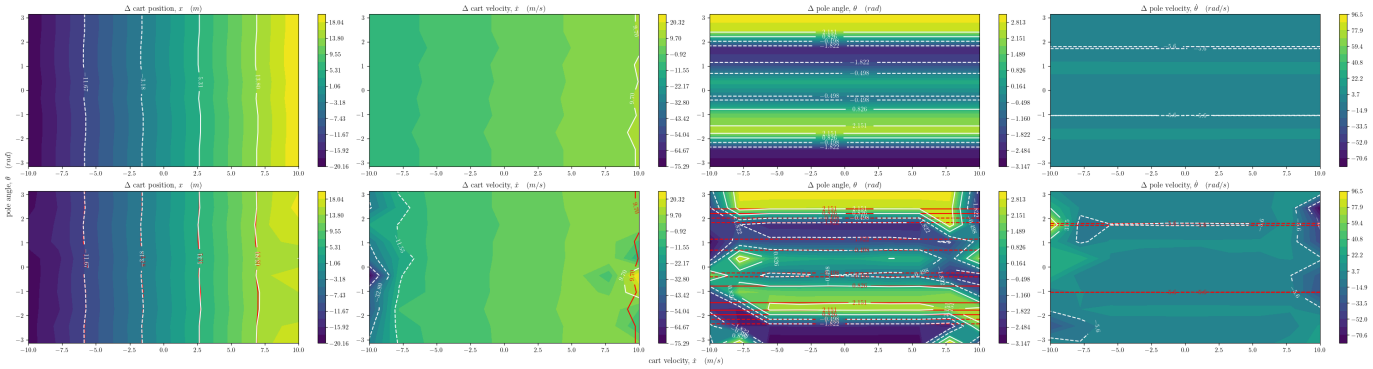

Fig. 29. Contour plots showing resulting states for actual (top) and predicted (bottom) dynamics for 2D scans of pole angle and cart velocity after $T = 2$ seconds. Red lines correspond to contour lines from the actual dynamics.

to prevent the cart position from exploding. In practice, however, identifying whether an unknown or complex system is translation invariant is likely impossible, and such changes cannot be easily done without hurting the predictive model. Hence we do not apply the aforementioned changes in our model.

VIII. Task 2.2

A. Addition of the action state into the non-linear model

The non-linear model was extended to include the force action F as an additional state. This allows the model the predict state changes under the influence of an applied force for $|F| < 20N$. Maintaining the same number of data points and basis points, the RMSE of the non-linear model increased from 0.00015 to 0.00072. This increase is expected as the number density of data points and basis points has decreased due to the state space becoming exponentially larger from the added dimension. Nevertheless, the model still fits the data relatively well. The time response of the actual system and model is presented in Figure 30 with an initial force input of 1N. The model appears to track better for starting states with non-zero velocities. For example, the model tracks the pole angle well up to 2.4 seconds or 2.5 oscillations when started from $(0, 2, 0.5, 2, 1)$. We found that the model performs worse in the absence of an input force and diverges from the actual dynamics even earlier. Interestingly, the model is noticeably more well-behaved when it diverges from the actual dynamics as the state magnitudes do not grow as fast as evident from the red line in the time response in constrast to that in Section VII. Although this may be due to coincidence, a possible explanation would be the model is now less over-fit due to the sparser training data density, allowing the model to better generalize to previously unseen trajectories.

When considering the 2D scans of the states across a time period $T = 1, 2, 3$ seconds, the state pair $(\dot{x}, F)$ shows the greatest variation in contours for F and $\dot{x}$ as shown in Figure 31. Intuitively, this makes sense when considering the actual dynamics. The direct application of some force on the cart causes it to accelerate, which is directly reflected in the increase in the cart and pole velocity. Hence, both these states would contribute a significant amount of errors in the model.

IX. Task 2.3

A. Implementing a linear policy

We implement a controller using a linear policy given in (9) that is capable of balancing the pole at the upright position or the unstable equilibrium.

$$p(X) = \mathbf{p} \cdot \mathbf{X} \tag{9}$$

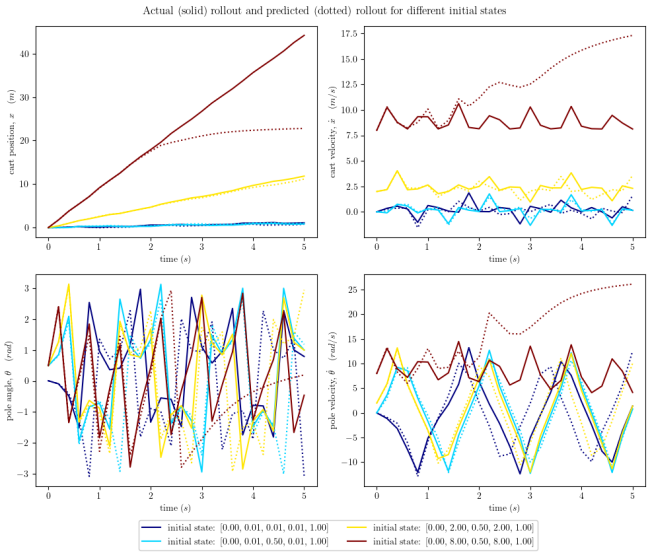

Fig. 30. Roll out

There are four weights p_i to be optimized for the linear policy. The loss function used to train the weights is given by (10). By default, the scaling factor σ used is 0.5 and can be fine-tuned further if necessary to aid the optimization process. For now, using $\sigma = 0.5$ is sufficient to find a set of optimal weights that achieves the desired objective. If the current state is far from the target state, $\mathbf{x}^* = \mathbf{0}$, the loss function saturates at 1 with small gradients as the exponential term tends towards zero. Hence, traditional machine learning techniques such as stochastic gradient descent (SGD) would be ineffective.

$$L = \sum_{i=1}^{N} l(X_i)$$
$$l(X) = 1 - e^{-|X - X_0|^2 / 2\sigma_l^2} \tag{10}$$

Given the relatively small dimensions and gradients, the Nelder-Mead optimizer from SciPy is used to fit the linear policy and minimize the aforementioned loss function. The chosen time period must be sufficiently long to ensure the states have decayed to zero indefinitely. Hence, we set $T = 3$ seconds, i.e., computing the total loss over 15 calls to `perform_action()`. Starting from an initial state of $(0, 0, 0.5, 0)$, we took 1D scans over the initial guess for each weight $p_i \in [0.8, 1.2]$. However, the optimal weights returned still fail to keep the pole upright. Hence, the optimization is performed further for regions nearby this set of weights. The range limits defining the next search space is computed using my own custom function, which perturbs the current weights by a small amount dependent on the hyper-parameter α. To further refine the search, α can be increased. We found that if $alpha$ is too large, i.e. 20, the Nelder-Mead effectively arrives at the same local minima. To escape a local minimum, α can be decreased instead. We found that using $\alpha = 5$ is effective in allowing Nelder-Mead to explore several nearby local minima.

Algorithm 1 Computing limits of next search space

Input: Optimal weights from previous search P_0, factor, α

```
δ = |P_0|/α
limits = Array()
for i, p in enumerate(P_0) do
    lim min = p - delta[i]
    lim max = p + delta[i]
    limits.append(lim min, lim max)
end for
```

Even by repeating the above process of exploring and refining the search, the resulting set of weight vectors could not balance the pole upright. Hence, we decided to identify regions containing local minima by hand, while using Nelder-Mead to further refine our search if required. We begin by plotting 2D scans of the weight vector around the region close to the last weight vectors found, $(1, 1, 13, 2)$. By inspecting the contour plots, we can identify regions of local minima, get the intersection of those regions across each plot and move our next initial guess there. As an example, the following contour plots were obtained on the first attempt as shown in Figure 32. Although p_1, p_2, p_3 has no ranges in common, we selected the region that appears to have the widest local minimum which can be found from the middle contour plot. Hence, the next initial guess is updated to $(1, 1.2, 13, 2)$. By repeating this process, we arrive at the weight vector $(1, 1.2, 16, 2.75)$ which gives the following contour plot shown in Figure 33. The wide dark blue regions imply a feasible local minimum has been found. Performing the Nelder-Mead optimization once more, the optimal weight vector obtained is $(1.83, 2.20, 19.37, 3.02)$.

The time responses of the actual system starting from the state (0, 0, 0.5, 0) without feedback action and with the optimal linear policy are shown in Figure 34. The linear policy is successful in maintaining the pole in an upright position for an indefinite amount of time. To get the range of states in which the linear policy is capable of maintaining the pole upright, a 1D scan is performed over each state variable and the points at which the loss diverges were identified as shown in Figure 35. Hence, the operating ranges are approximately $|x| < 1.4, |\dot{x}| < 1.1, |\theta| < 0.55, |\dot{\theta}| < 2.1$. Note that the true range is smaller than that defined previously as the 1D scans are performed with other variables fixed at 0. If other states are non-zero, the operating range for some state variables decreases. This can be visualized by setting all states to 0.2 and performing a 2D scan across the aforementioned operating ranges and visualising the loss with contour plots. From Figure 36, the total loss clearly diverges at certain regions within the specified ranges.

X. Task 2.4

A. Model predictive control

Given that the model tracks the actual dynamics relatively well for 2 seconds and the optimal linear policy found previously stabilizes the system after approximately 2 seconds, we expect the optimal linear policy found using model predictive

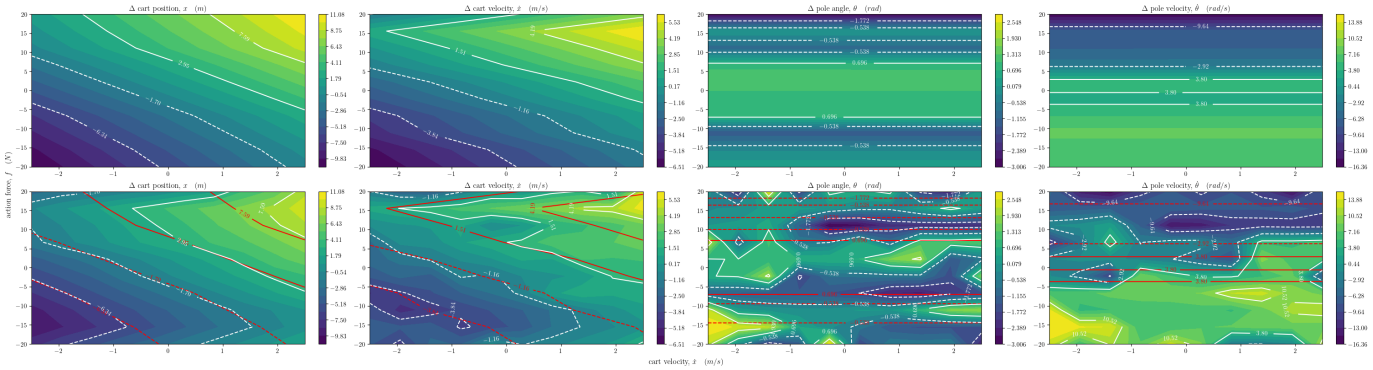

Fig. 31. Contour plots showing resulting states for actual (top) and predicted (bottom) dynamics for 2D scans of force and cart velocity after $T = 2$ seconds. Red lines correspond to contour lines from the actual dynamics.

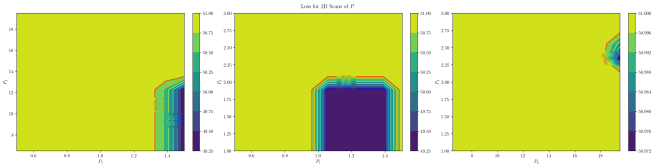

Fig. 32. Contour plots showing 2D scans of p_1, p_2, p_3 and the resulting loss in our initial search

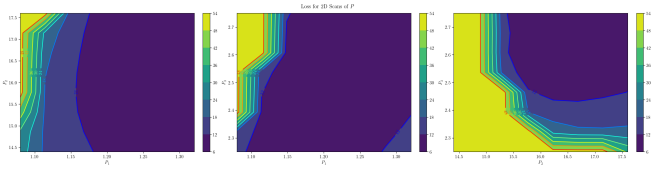

Fig. 33. Contour plots showing 2D scans of p_1, p_2, p_3 and the resulting loss in our final search.

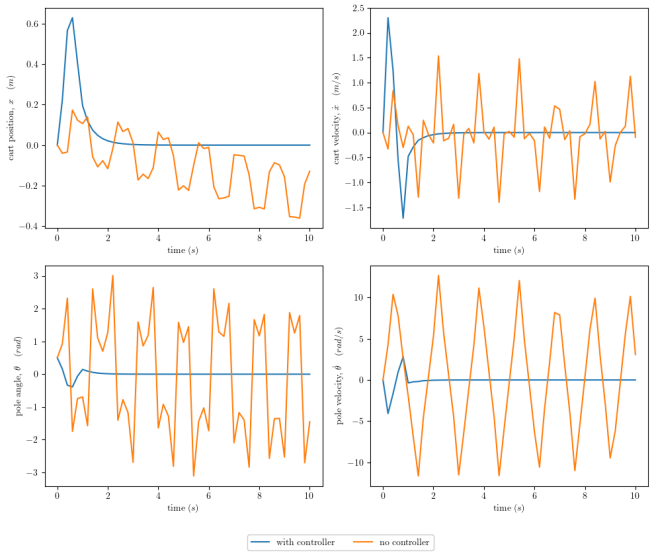

Fig. 34. Actual time response with (blue) and without (orange) the optimal linear policy.

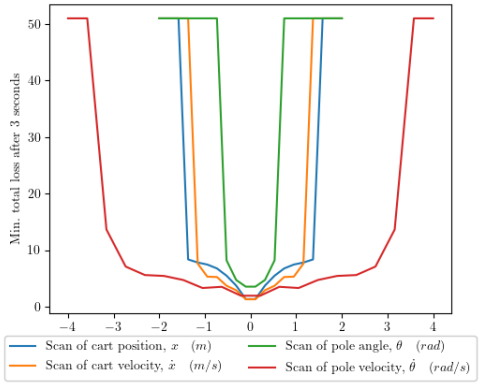

Fig. 35. Total loss against 1D scans of state variables.

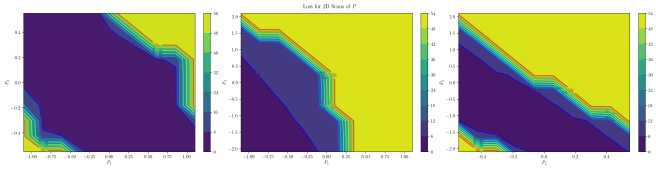

Fig. 36. Total loss against 2D scans of state variables.

control to be relatively similar to that found using the actual system, with small deviations due to errors in the model. The optimal policy found previously meets the target objective when applied to the predicted dynamics but with non-zero steady-state errors. Applying the Nelder-Mead optimization with the non-linear model and the previous optimal policy as our initial guess, we obtain a different but similar weight vector as before. When a time horizon of $T = 3$ seconds is used, the updated weights are unable to stabilize both the model and the actual system. However, a new but slightly worse optimal linear policy that is able to stabilize both the actual and modelled system as shown in Figure 38 can be found with a time horizon of $T = 10$ seconds. However, predicted dynamics still show a non-zero steady-state error. As expected, the operating ranges obtained for this linear policy, $|x| < 1.4, |\dot{x}| < 1.6, |\theta| < 0.35, |\dot{\theta}| < 3.5$ are smaller than

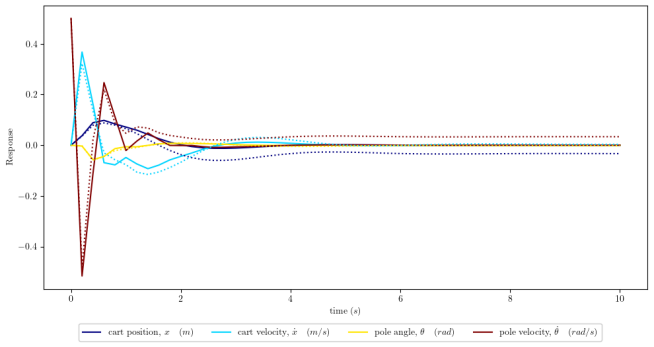

Fig. 37. Actual (solid) and predicted (dotted) time response with model predictive control.

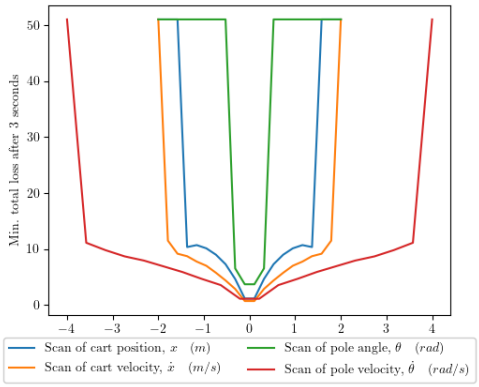

Fig. 38. Operating range of optimal linear policy model predictive control.

that of the previous optimal linear policy with the exception of $\dot{\theta}$. An interesting thought would be introducing some form of integral action into the system to reduce the steady-state errors to zero.

XI. Task 3.1

A. Modelling the dynamics with added observation noise

In this section, we consider modelling the actual dynamics with added observation noise using both a linear and non-linear model. The noise samples are drawn from a multivariate Gaussian with zero mean and varying noise scales to model additive white Gaussian noise (AWGN) present in sensor measurements in real systems. To simplify experiments, we assume the same noise scale across all states which is unrealistic. The scatter plots showing the predicted against actual change in state values after a single call to `perform_action()` for both linear and non-linear models are shown in Figure 39 and Figure 40 respectively. The accuracy of both models deteriorates with increasing noise scales when contrasted with the no-noise models as reflected in their RMSE.

Upon plotting the RMSE as a function of noise scales as shown in Figure 41, we see that the quality of the non-linear model deteriorates rapidly and approaches that of the linear model at large noise scales. We reason that this is due to the non-linear model having learnt to fit the noise rather

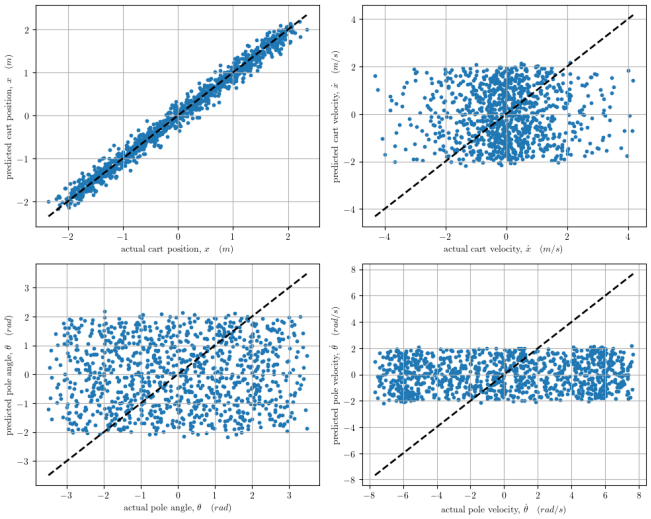

Fig. 39. Scatter plot of predicted change against actual change for a linear model fitted with noisy observations

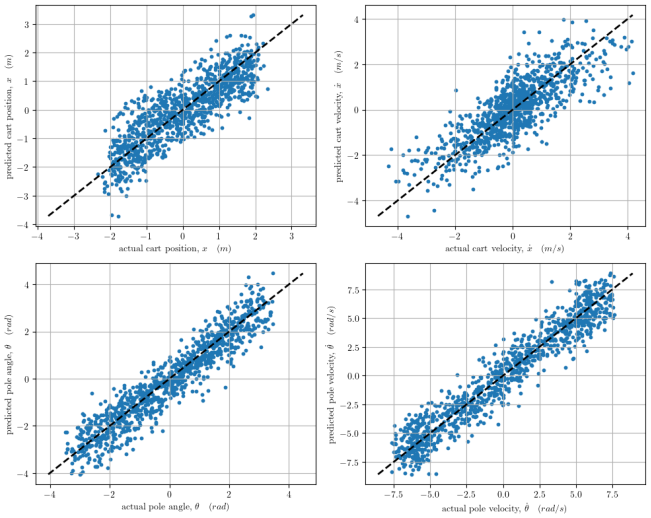

Fig. 40. Scatter plot of predicted change against actual change for a non-linear model fitted with noisy observations.

than the actual system dynamics. In contrast, the linear model fails to capture any information about the noise. The linear model effectively takes the linear combination of independent Gaussian-distributed noise which gives an expectation of zero when averaged over large samples as they cancel out. Our hypothesis is further verified with the use of a non-linear model with twice as many basis points (320 to 640). Consistent with our expectations, the larger non-linear model over-fits on the noise rather than the actual dynamics and starts performing worse than the linear model for noise scales larger than 6. Hence, we must be wary when using large non-linear models in the presence of observation noise. Furthermore, the linear model is more robust to the impact of increasing noise levels and is thus more appropriate for larger noise scales.

For a noise scale of 1, we can plot the time response

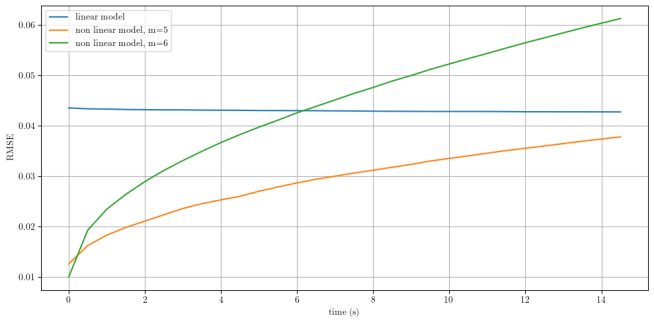

Fig. 41. Model RMSE against noise scale used.

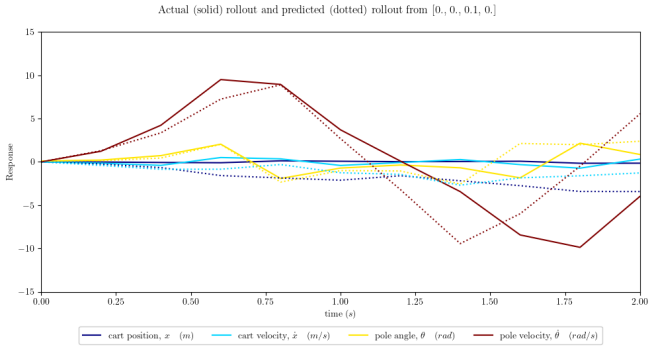

Fig. 42. Actual (solid) and predicted (dotted) time response fitted with observation noise.

to observe how the model tracks the actual dynamics and how long it takes before the predicted trajectories diverge when the pole is slightly displaced from its upright position, $\mathbf{X} = (0, 0, 0.1, 0)$. From Figure 42, it takes about 0.25 seconds before the model diverges, except for the pole angle which it tracks well up to 1 second. When longer time horizons are considered, the predicted pole velocity and cart velocity still oscillates around zero just as the actual dynamics do.

When optimizing the linear policy with the added observation noise, we apply the same Nelder-Mead optimizer and Sobol sequences as initial guesses. The weight vector returned is $(2.00, 9.50, -1.36, 7.36)$ which is significantly more different than that found using non-noisy observations. Applying this linear policy to the actual dynamics, the system response is shown in Figure 43 which fails to keep the pole upright but maintains it around 2.8 radians. In addition, the velocities oscillate around zero, and the cart position is prevented from growing exponentially.

If our initial guess is set to the optimal linear policy found in Section IX and optimized over $T = 10$ seconds, the optimal linear policy found is $(1.81, 2.22, 17.55, 3.23)$, which is relatively similar to the initial guess. This optimal policy to the actual system manages to achieve the desired target but requires a longer time ($\tilde{4}$ seconds) to decay the oscillations to zero. However, with model predictive control, the optimal linear policy fails to decay the pole dynamics to zero and the system oscillates about the origin as seen in Figure 44.

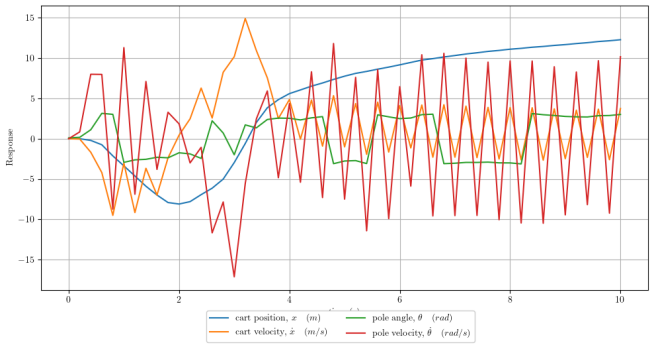

Fig. 43. Time response of the actual system with the optimal linear policy applied, $P = (2.00, 9.50, -1.36, 7.36)$.

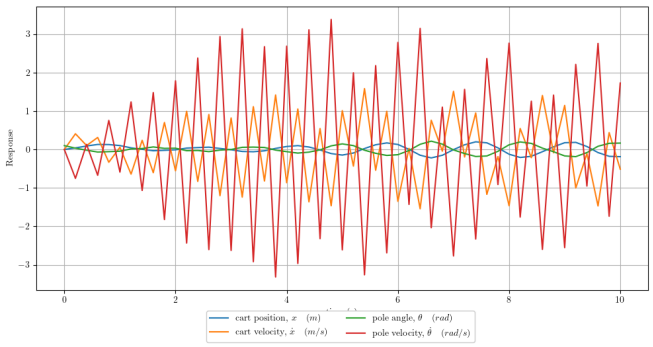

Fig. 44. Time response of the actual system with model predictive control applied, $P = (1.90, 2.30, 15.75, 3.35)$.

XII. Task 3.2

A. Modelling the dynamics with added observation and disturbance noise

We now consider modelling the actual dynamics with added observation noise using both a linear and non-linear model. To simplify our investigation, we begin with the same multivariate Gaussian with zero mean and unit scale for modelling both types of noise. The scatter plots showing the predicted against actual change in state values after a single call to `perform_action()` for both linear and non-linear models are shown in Figure 45 and Figure 46 respectively. Although the non-linear model still fits the actual dynamics better with the unit noise scale, the RMSE for the linear model remains largely similar, that is 0.045 (now) vs 0.043 (previously), demonstrating its robustness to noise. However, note that the errors in tracking cart position have amplified significantly as the addition of disturbance noise offsets the cart from the origin. The non-linear model suffers a huge increase in RMSE, that is 0.026 (now) vs 0.018(previously) which shows the non-linear model's greater sensitivity to noise.

Taking a noise scale of 1, we can plot the model time response to check how well the model tracks the actual dynamics and how long it takes before the predicted trajectories diverge from the pole slightly displaced from its upright position, $\mathbf{x} = (0, 0, 0.1, 0, 0)$. As a result of the added disturbance noise,

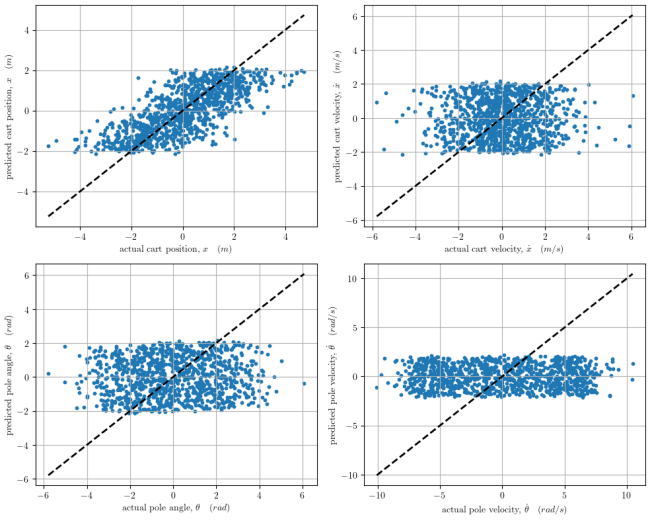

Fig. 45. Scatter plot of predicted change against actual change for the linear model fitted with observation and disturbance noise.

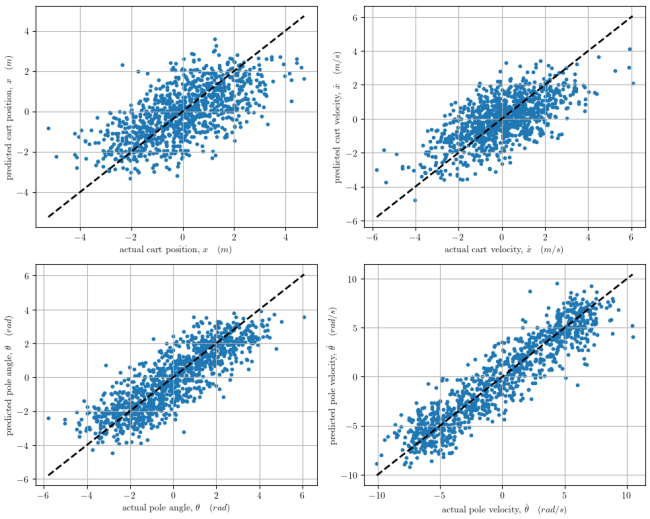

Fig. 46. Scatter plot of predicted change against actual change for the non-linear model fitted with observation and disturbance noise.

the actual dynamics now appear to oscillate around a moving trend due to drift from the disturbance noise. Furthermore, the period of oscillations appears to vary randomly and the non-linear model fails to track the oscillations accurately. The model still manages to capture the trend for the pole dynamics but fails to do so for the cart dynamics. The predicted dynamics diverge from the actual dynamics right from the start as shown in Figure 47. Hence, the effect of added disturbance noise is much more apparent in the cart dynamics compared to the pole dynamics. We reason that the trigonometric functions and periodicity in the pole dynamics somewhat dampen the effect of the added disturbance noise.

We attempted to optimize the linear policy with the actual model and through model predictive control. We experimented with different but shorter time horizons to account for the

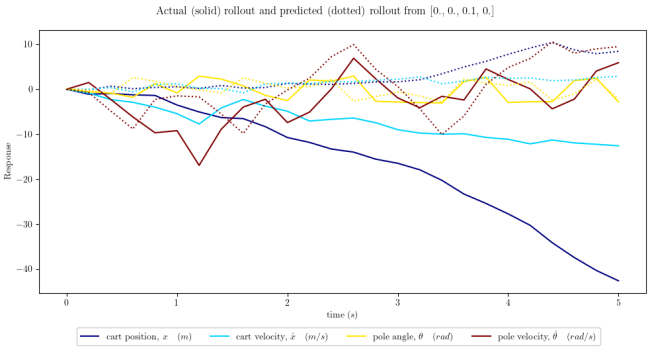

Fig. 47. Actual (solid) and predicted (dotted) time response fitted with observation noise.

Fig. 48. Time response of the actual system with linear policy trained on the actual model, $P = (-5.66, -8.37, 3.10, -8.88)$.

bad tracking and prevent errors from further accumulating for model predictive control. Applying the same Nelder-Mead optimizer and Sobol sequences as initial guesses, we found that none of the linear policies found was able to maintain the pole at a stable position, with the cart dynamics eventually exploding as seen in Figure 48. Using the optimal policy found in Section IX as initial guesses, the optimized linear policy returned by the optimizer still fails when tuned using the actual system and model as shown in Figure 52 and Figure 50 respectively.

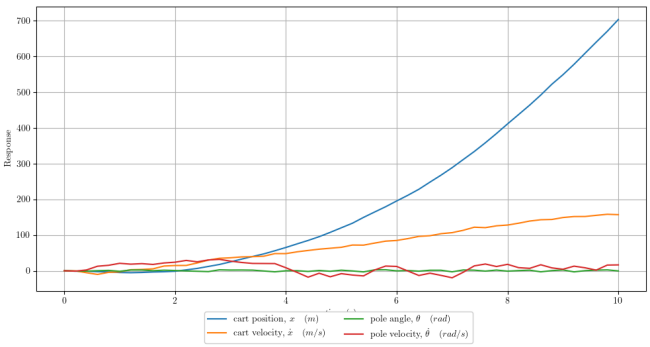

Fig. 49. Time response of the actual system with linear policy trained on the actual model, $P = (1.78, 2.24, 19.42, 3.11)$.

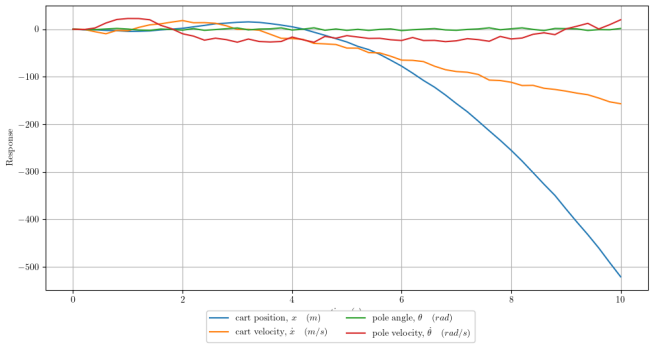

Fig. 50. Time response of the actual system with model predictive control applied, $P = (1.83, 2.24, 17.22, 3.09)$.

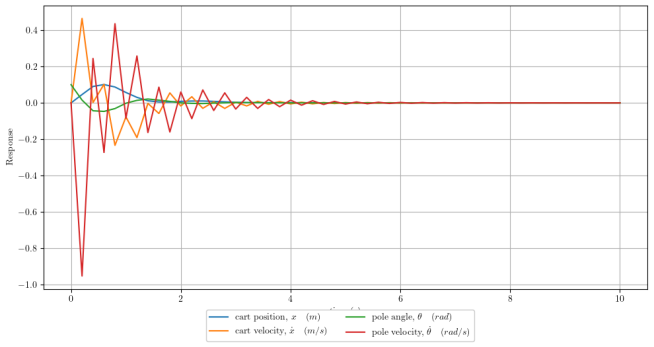

Fig. 51. Time response with model predictive control (obs. noise).

B. A review on the noise scales

Taking a step back, the same unit noise scale is applied across all states in the previous experiments which is unrealistic. Treating the observation noise as measurement noise from noisy sensors, a noise scale of 1 is extremely large for states such as the pole angle, which ranges between $-pi$ and pi. Hence, it is expected that the model and linear policy to perform poorly in this case. Hence, we now consider the use of noise samples drawn from a multivariate Gaussian distribution, but now with dynamic noise scales that are proportional to the current states so that the larger measurements result in greater measurement errors and vice-versa. In doing so, we managed to achieve improved model-tracking performance and model predictive control. An interesting observation was obtained - for the same initial guesses of the linear policy, the optimal policy obtained with added disturbance noise appears to perform better than that with added observation noise only with a faster response shown in Figures 51 and 52. A similar observation can be found with model predictive control whereby the model with added disturbance noise has a faster response decay. We reason that this is due to the added noise in the system dynamics contributing some form of damping that helps to "smooth" the observed dynamics, allowing for the linear policy to learn better.

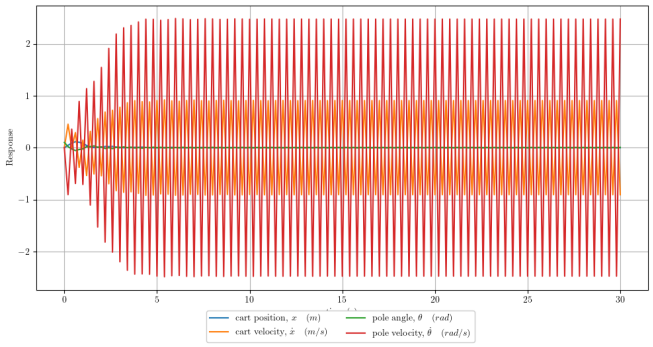

Fig. 52. Time response with model predictive control (obs. + disturb. noise).

C. A comparison between observation and disturbance noise

To isolate the effects of the disturbance noise on the system, we repeated the experiments with only disturbance noise added. In this case, the non-linear model manages to track the actual dynamics significantly better than when observation noise is added. Hence, the observation noise is mainly responsible for the large errors encountered in previous sections.

We considered the effects of improving our models with the use of a Wiener filter from SciPy to extract the actual dynamics from the noisy signal. Empirically, we found that a filter window of 2 works best. While longer filter windows are able to achieve better smoothing, they tend to over dampen the effects of oscillations in the pole dynamics. Figure 53 shows multiple time responses of the actual system and model under various noise conditions, in addition to the filtered noise models. As expected, the addition of the wiener filter hurts the performance of the model without no noise (blue). When observation noise is added (orange), the use of a wiener filter allows the model to track the pole dynamics better. However, the filtered model still fails to track the systems with added disturbance noise. In addition, we note that the use of a wiener filter allows the model somewhat capture the oscillation periods better in the presence of observation noise, and that it prevents the predicted state trajectories such as the cart position from exploding across long time horizons. Consequently, we hypothesize that using the filtered models model predictive control may yield better results. It would be interesting to experiment with the use of more advanced filters such as Kalman filters if time permits. We suspect they may be effective in mitigating the effects of disturbance noise due to their ability to correct drift errors in the model.

D. The addition of bias and other types of noises

The effect of bias in the noise is investigated by observing the time response of the system in the presence of a multivariate Gaussian noise centred on 0.5 shown in Figure 54. The added disturbance noise causes the cart dynamics to drift significantly, while the pole dynamics remain relatively similar as evidenced by the pole velocity. The model manages to track the system with added disturbance noise only (red)

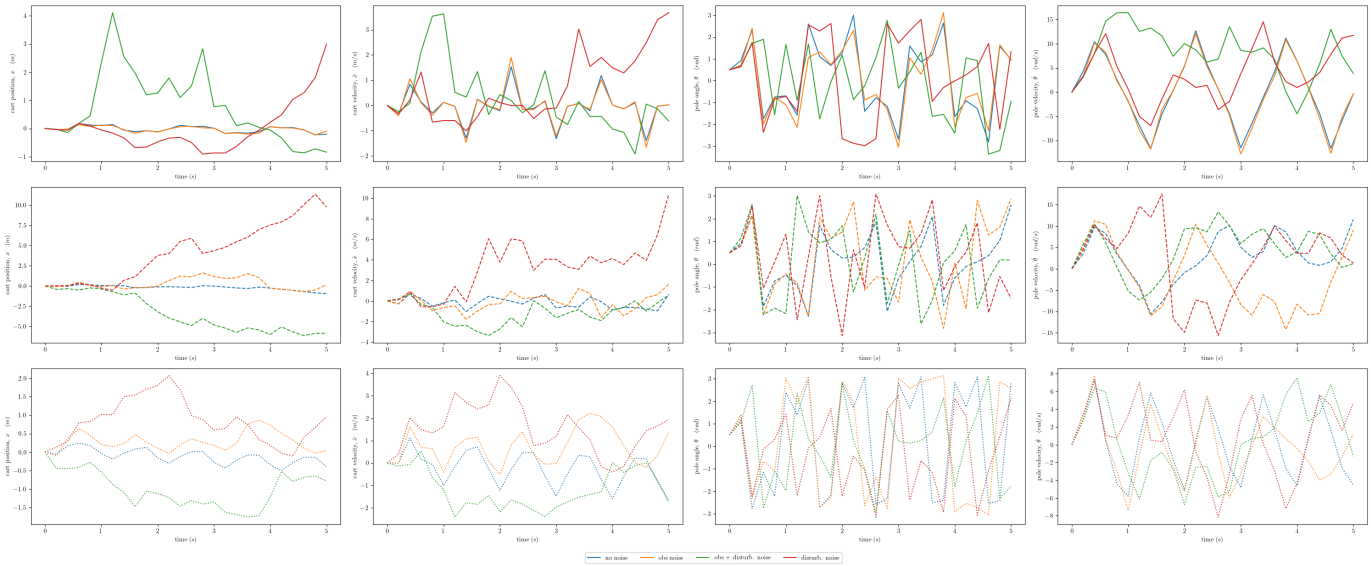

Fig. 53. Actual (top), model (middle) and filtered model (bottom) time response starting from $(0, 0, 0.5, 0, 0)$ under different noise conditions.

in addition to the resulting drift in the actual dynamics but suffers in predicting the pole dynamics. When observation noise is further added, the model completely breaks down as the offset errors in the training samples are compounded. Hence, we would expect any form of model predictive control developed directly using these models to fail. Similarly, the use of Kalman filters here may help address the offset errors.

If time permits, it would be interesting to investigate other disturbance noise distributions such as Poisson processes, which are common in modelling abrupt real-world events such as earthquakes. An example of such disturbance noise would apply some external force towards the system at time intervals that follow the negative exponential distribution. Maintaining the pole in its upright position under such conditions requires the controller to be able to react quickly to abrupt changes in system dynamics. In this case, it might be necessary to decrease the current Euler integration time delta in the cart pole system. This corresponds to increasing the sampling rate of the controller in practice and enables the controller to react faster to changes in the system dynamics.

XIII. Task 4

In this section, we design a non-linear policy that can swing the pole upright from its stable equilibrium position to an upright position and maintain that position. To simplify the problem, we start by training the non-linear policy using the noise-free version of the actual system dynamics. The non-linear policy is given by the following equation:

$$p(X) = \sum_i w_i e^{-0.5(X-X_i)^T W (X-X_i)} \tag{11}$$

which represents the weighted sum of kernels with basis centres x_i.

A. Brute force optimization

For a symmetric W, there are 10 entries to be selected (number of non-diagonal entries on one side + 4 diagonal entries). For n basis points, there are $4n$ entries to select. In addition to the length n weight vector $omega$, there are $5n+10$ variables to be optimized, which is significantly more than that of the linear policy. For our initial guesses, we used random sampling points for our basis centres and set the rest of the variables to 1. Packing these parameters into a vector and applying the Nelder-Mead optimization method as before, we found that the variables barely changed or stayed the same as our initial guess. Multiple attempts at trying different initial guesses were carried out to no avail. This is expected as the Nelder-Mead method is only effective for low dimensional problems without available gradients [1]. To remedy this, we experimented with other optimization methods provided by Scipy such as the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. Unlike Nelder-Mead, BFGS accepts constraints, which allowed us to reduce the search space. However, this did not work either.

Before considering other methods, we experimented with custom loss functions. From the original loss function given in (10), we reason that variables were unable to converge using the proposed optimization methods as they were stuck on a flat hyperplane – if the states are far away from the basis centres, $e^{-|X-X_0|^2/2\sigma_l^2}$ tend towards zero and the loss function saturates at a value of 1, resulting a constant total loss for a wide range of states. To address this, we decided to minimize the log loss instead. In addition, to prevent the Nelder-Mead method from exploring unfeasible solutions, a conditional statement was added which returns the maximum loss if the parameters become too large. Intuitively, this should also help reduce the training times, although the training procedure would likely be more unstable due to discontinuities

Fig. 54. Actual (top), model (bottom) time response starting from $(0, 0, 0.5, 0, 0)$ under different noise conditions with non-zero bias.

in the new 'gradient'. Lastly, we added the feedback action as a state parameter into the loss function. Intuitively, this penalizes the non-linear policy for using excessive force and preventing the weights from exploding, given that the force saturates at 20 N in the actual system. After trying various combinations of such modifications, none of the changes gave a noticeable improvement in the optimization process. We note that moving the basis functions nearer to the target objective yields solutions that allow the system to converge closer to the target objective. However, doing so makes the optimization harder due to the low density of basis points in other regions with the loss saturating at 1, causing the model to be trapped on a local maximum.

It became clear that the number of parameters to be optimized had to be reduced. Firstly, we let $W = ww^T$ be the outer matrix product of a length 4 vector. This reduced the number of entries to be optimized from 10 down to 4. Intuitively, the exponent of the non-linear policy becomes $-0.5((X - X_i)^T w)(w^T(X - X_i))$, which is the weighted sum of the squared difference of the current state and the basis points.

B. Devising an optimization strategy

From the previous section, it is clear obtaining the optimal weights for the non-linear policy via that brute-force optimization is computationally infeasible with the currently available resources. Hence, we devised a multi-step optimization strategy based on an intuitive understanding of the system.

Starting from the stable equilibrium, a sufficiently large force must be applied to swing the pole up. By applying a 20 N initial force at the stable equilibrium, the system begins oscillating as shown in Figure 55. However, the amplitude of the oscillations is too small for the pole to approach the upright position. Furthermore, a constant application of 20 N force towards the system causes the pole angle to approach 1 radian after 1 second but proceeds to oscillate between -1 and -3 radians indefinitely. Hence, the applied force should have the same resonant frequency as the system to provide sufficient momentum to swing the pole upright.

Without referring to equations of motion, we can train another model to predict this frequency. However, we hypothesize that this is not necessary, and can instead be exploited as a heuristic when designing the appropriate loss function for training the non-linear policy. That is, the direction of the applied force F should follow the pole velocity $\dot{\theta}$. In addition, an over-application of force causes the cart position x and velocity $\dot{x}$ to increase exponentially, which is undesirable. Hence, we reason that the applied force should be proportional to the pole angle θ so that no force tends to zero once the upright position is reached. Coupled with our observation that having the applied force in the same direction causes x and $\dot{x}$ to grow exponentially quickly, we instead consider the change in θ to determine the direction of F as presented

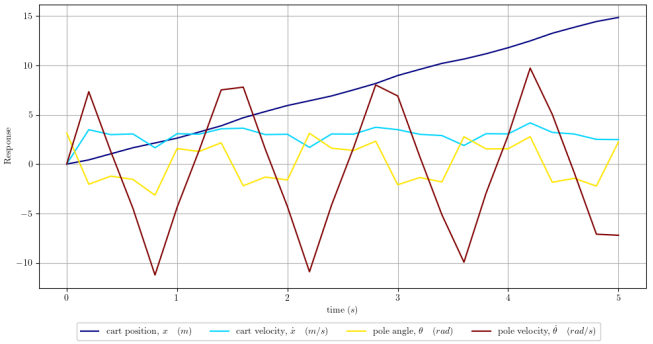

Fig. 55. Time response when a 20 N initial force is applied.

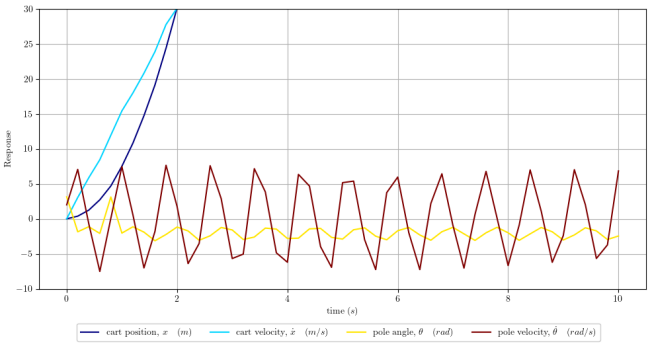

Fig. 56. Time response when a 20 N force is continuously applied.

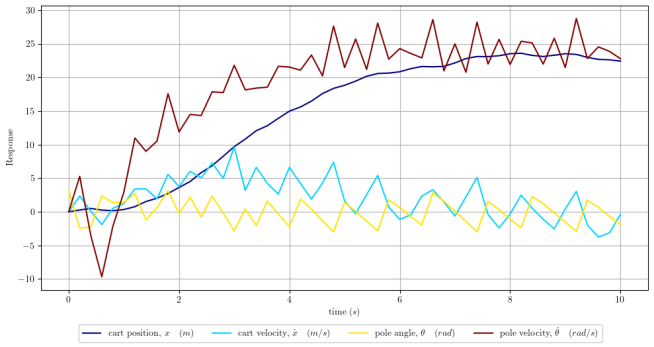

Fig. 57. Time response when the resonance controller is applied.

Fig. 58. Time response when the non-linear policy is applied.

in the pseudo-code below. In doing so, we found that this introduces some form of 'delay' in terms of knowing which direction the pole swings, leading to the periodic application of F in the wrong direction and thus providing sufficient damping effect as demonstrated in Figure 57. This controller (termed resonance controller for explanation purposes) is able to provide sufficient momentum to the system without x and $\dot{x}$ growing too quickly. Once the following conditions in (12) are met, the system has now sufficient energy to achieve the upright pole position without requiring further application of force at the resonant frequency.

$$|\dot{x}| > 1, |\dot{\theta}| > 1, |\theta| < pi/2 \tag{12}$$

Algorithm 2 Heuristic for computing the force to be applied

If $\Delta\theta >= 0$:
  If $|\theta| >= \pi$: sign = -1
  else: $sign = 1$
else:
  If $|\theta| >= \pi$: sign = 1
  else: $sign = -1$
$force = sign * \theta * 10/\pi$

In the next stage, we attempt to use the non-linear policy to decay the states to zero. Intuitively, this process should be executed quickly to prevent the pole angle from falling back to its stable equilibrium. Hence, we expect the application of force in this stage to be large and sudden. To pick the parameters of the non-linear policy to be optimized, we consider the following. (i) The parameter w acts as weights assigned to each state, similar to that found in the linear policy. Hence, it is reasonable to re-use the optimized linear weights found previously as initial guesses for w and fix them. (ii) At resonance, the state trajectories of the system are difficult to predict and cover a wide region. In Section VII, we found that the Sobol sequence provides relatively uniform coverage over the state space. Hence, we fix the basis points, X_i, using the Sobol sequence over the entire state space defined in (4). It may be better to constrain the state space further and increase the number density of basis points in the region of interest. For now, we use 8 basis points to reduce the computational resources required for optimization, leaving us with just ω to optimize over.

To train the non-linear policy, we use an initial state of $(0.34, 1.25, 1.43, 2.94)$ which satisfies the condition in (12) and achieved using the previous controller. With an initial guess of all ones, the optimized weights for ω are $(32.67, 71.91, 16.48, -2.99, 12.65, -33.94, 13.91, -8.85)$. We found that this set of weights can decay the states to zero for a small window of 0.4 seconds but cannot maintain them.

Since our linear policy from the previous section worked for small perturbations from the unstable equilibrium, we can use the same controller to maintain the states at zero. The condition required for the switch to a linear controller is when the absolute values of all states are smaller than 0.1. This satisfies the operating ranges of our linear controller found in Section IX. Hence, we proposed the ensemble controller consisting of the resonance controller, the non-linear policy and the linear policy, with internal states to switch between controllers accordingly depending on the input states.

C. Ensemble controller performance

The following time response is obtained with the ensemble controller, starting from the stable equilibrium position as shown in Figure 59. The controller manages to achieve the desired objective within 5 seconds. The response is consistent with our intuition – large oscillations at the beginning due to feedback from the resonance controller allowing energy build up in the system, followed by the rapid decay of states with the non-linear controller and finally maintaining the pole upright with a linear controller. As of now, the controller is extremely sensitive to disturbances in the initial states. For any perturbation in pole angles larger than 0.0001, the controller fails. Nevertheless, we hope that our experiment provided valuable insight into how such a controller interacts with the system. From there, more heuristics be designed/inferred and used to design a better controller.

D. Potential improvements and further works

The ensemble controller's sensitivity to the initial condition is a clear indication of over-fitting in addition to some luck for stumbling upon such a state trajectory. Given that the operating range of the linear controller is roughly known from Section

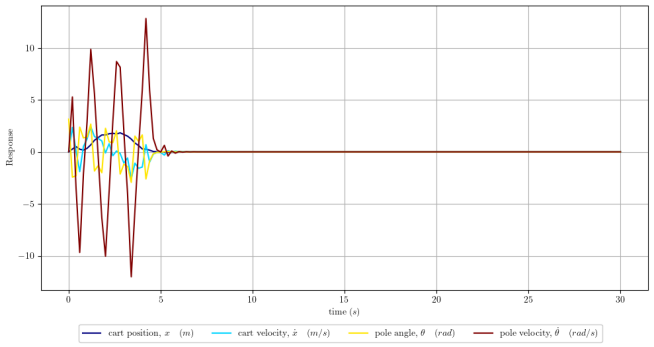

Fig. 59. System response with the ensemble controller.

IX, the failure modes of the ensemble controller should lie in the resonance controller and the non-linear controller.

Firstly, the resonance controller can be replaced with the non-linear policy and provide even better feedback control. The target objective of such a controller would be to drive the system to some state whereby the pole angle is close to 0, without incurring excessive cart position from the origin. The non-linear policy in the next stage then decays the cart and pole velocities to zero as fast as possible while preventing the pole from falling back down. To improve robustness, the non-linear policy should be optimized over a range of initial states with non-zero velocities. For optimization, we would focus on tuning the locations of the basis points. We found that the controller learns better when the input states are closer to the basis points. Hence, I would selectively concentrate these basis points in regions whereby the system is in "resonance" due to feedback action from the resonance controller. In terms of optimizing W, I found that the Nelder-mean optimization fails if W is too large as the exponential term tends to zero. If I let $W = ww^T$, decreasing w_i causes the loss function to penalize the input states far away from $mathbfx_i$ more. Hence, we may use w to "guide" the training trajectory.

Given the non-linear model's ability to generalize and learn non-linear state space trajectories, I believe a well-tuned controller with a non-linear policy should be able to drive and stabilize the pole upright for a variety of initial conditions with a fixed set of parameters. Building on our ensemble controller, we propose such a controller might be trained using a custom objective function which accepts a set of states across a pre-defined time period corresponding to the state trajectories achieved by the ensemble controller to stabilize the system. Rather than penalizing the current state for deviating from the target state, this objective function instead penalizes the controller for deviating from the state trajectory. Due to time constraints, we are unable to test out such a loss function, and instead leave a pseudo-code below for interested readers.

By fitting the model across a set of initial starting points, the resulting non-linear policy should be robust to changes in initial states. If successful, the problem can be further extended to include the addition of observation and disturbance noise, whereby the use of Wiener filters or Kalman filters might be useful to smooth out variations and drift errors due to noise.

Algorithm 3 Objective function for training non-linear policy

Input: Initial state x_0 **Input**: Desired state trajectories s **Input**: Time, T

total loss, $L = 0$
current state, x_i
for i = 1, ... , T **do**
    x_i = perform action from x_{i-1}
    L += compute loss of x_i and s_i
end for

XIV. Conclusion

In this project, data on the dynamics of a cart-pole system is collected by running a simulation under various initial conditions. We found that the system is translation invariant and that the system undergoes simple or large oscillations about its stable equilibrium depending on the initial pole angle or pole velocity. By fitting a linear model on the data using the least-squares error, the model predicts the next cart position and pole angle reasonably well for a single call to `perform_action()`, but quickly differs from the actual system dynamics when the number of time steps is increased. Furthermore, the predicted time response by the model is very sensitive to the initial states used. The pole angle response requires mapping to prevent the model from diverging (becoming unstable). Hence, a non-linear model in the form of the weighted sum of Gaussian RBFs was used to model the system and achieves a much better fit in predicting the state changes across a finite time horizon of approximately 1.5 to 2.5 seconds. With the improved model and model predictive control, an optimal linear policy can be found via numerical optimization and the appropriate loss function. This linear policy is able to keep the pole upright when slightly displaced from its unstable equilibrium. Next, observation noise and disturbance noise was introduced into the system. We found that the linear model is more robust to noise relative to the non-linear model when the noise scales are sufficiently large. The use of observation noise significantly reduces the performance of the predictive models. Furthermore, the model fails to provide useful predictions when biases are introduced into the noise samples. In the final section, we attempt to design some non-linear policy that brings and maintains the pole upright from the stable equilibrium. Performing numerical optimization over higher dimensions is increasingly difficult, so we devised a multi-stage strategy to achieve our target objective with an ensemble of controllers. Overall, the cart pole system is a well-known system that has been studied extensively from the perspective of control theory and in recent years machine learning. It can provide many useful insights due to the wide variety of experiments that can be conducted on the system with varying difficulty levels.

References

[1] Gábor Csányi, "IIA project SF3: Machine Learning", Deepnote, Cambridge University Engineering Department, 2023.