UNIVERSITY OF
CAMBRIDGE

SF3 Machine Learning Final Report

Mihai Varsandan

mv436
Girton College

7th June 2019

Abstract:

This lab investigates the implementation and performance of the dynamics of a cart which at its centre has an inverted pendulum attached. A *linear* and *non-linear* model of the system is implemented using the change in state to predict the dynamics of the real system. Having fitted a good model, a *linear policy* is used control the predicted system. The objective is to balance the pole upright from the equilibrium position. Throughout the report, comparisons between the real system and the predicted system are presented and explained.

1 Task 1 Dynamical Simulation

The first task of the project was to explore the dynamics of the inverted pendulum system("Cartpole"). A Python class in *CartPole.py* file was given where it described the system. The state of the system is given by an array $state = [x, v, \theta, \omega]$ where:

- x is the *position* of the cart
- v is the *velocity* of the cart
- θ is the *angle* of the cart
- ω is the *angular velocity* of the cart

1.1 Task 1.1 Time Evolution Dynamics

The first task of the project was to use the *CartPole.py* to build a function which would display the time evolution of the system given an initial condition(state). In the python class given, there was a function defined as *performAction()* which would perform the dynamics of the system for a $\Delta t = 0.1s$. Within this time step, the function uses the Euler method for 200 steps and it updates its state at each step. At the last step, the stored state would correspond to a system evolution after $0.1s$. Therefore to create a time evolution of the system it is required to call the function *performAction()* a certain number of times. For example, for a given initial state($X(state) = [0, 0, \pi, 2]$) the system should output a simple oscillation around the equilibrium position. If the dynamics are run for a time $t = 50s$ the function is called 500 times. The results are shown in the *Figure 1* below. For different time evolutions with different conditions please see the figures in the *Appendix A*.

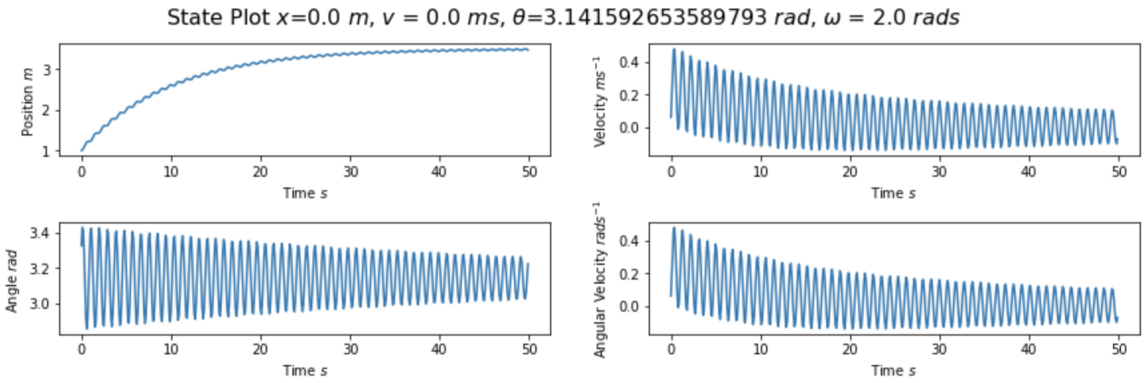


Figure 1: Small Oscillation of the Pendulum around Equilibrium Position

From *Figure 1*, it can be seen that the system performs as expected, with the cart position slowly increasing in an oscillation manner due to the moving of the pendulum.

1.2 Task 1.2 Time Evolution model

In Task 1.2, it was required to study 2 different models that could be used to model the evolution of the system. A model is a function of the current state $\underline{\mathbf{X}}$ which is used to predict the next state. An initial model would be to use the current state $\underline{\mathbf{X}}$ to find $\underline{\mathbf{Y}}$ which is the state after one call call of the function *performAction()*. To check how the model performs a 1-D scan has been conducted where the system X state is varied only in 1 direction while keeping the other dimensions constant. The output can be seen in the *Figure 2* below.

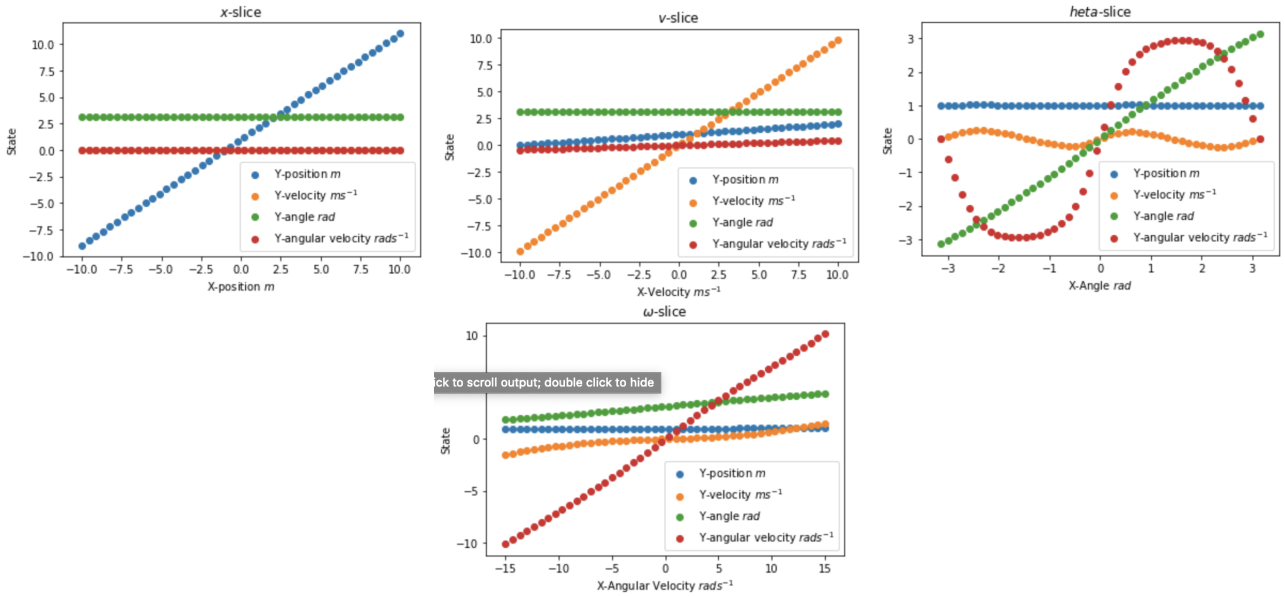


Figure 2: 1D slice of model $\underline{\mathbf{X}}_{n+1} = \underline{\mathbf{Y}}$

As expected, the relationship between $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ is almost linear in all cases apart from, when the angle θ and the angular velocity w_Y of $\underline{\mathbf{Y}}$ is varied.
An alternative model is to look at the change in state. Therefore $\underline{\mathbf{Y}}$ is redefined as $\underline{\mathbf{Y}} = \underline{\mathbf{X}}_{n+1} - \underline{\mathbf{X}}_n$ where n is number of times *performAction()* is called. The results of the 1D scans is shown in the *Figure 3* below. From the figures it can be observed that varying the position x has no effect on the next state.

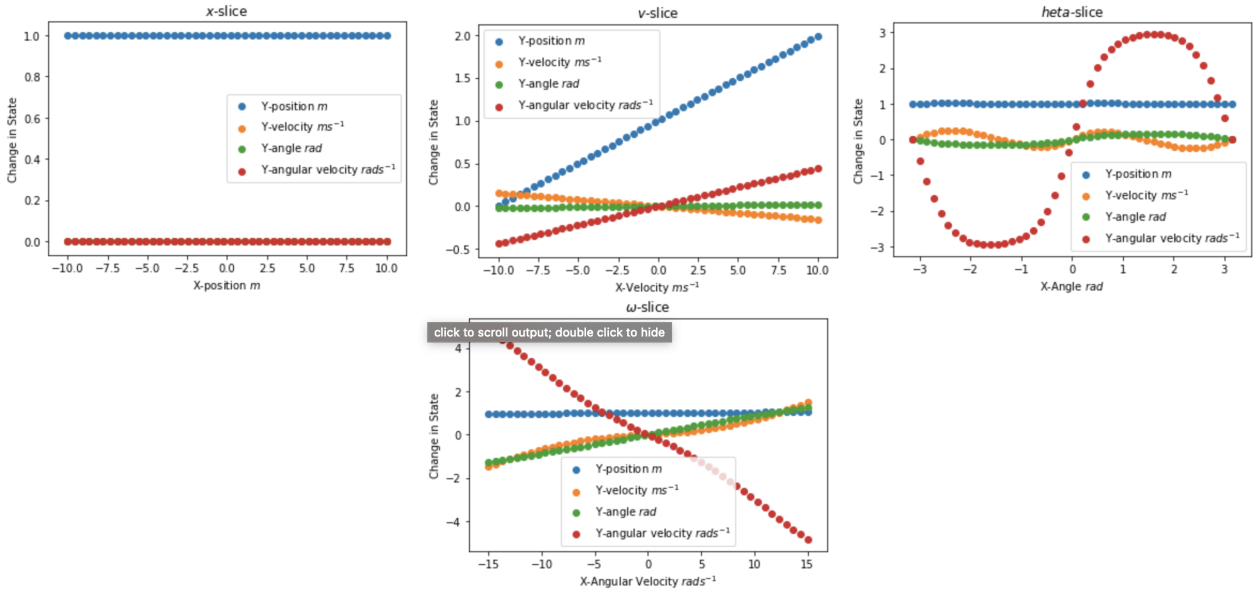


Figure 3: 1D slice of model $\underline{\mathbf{Y}} = \underline{\mathbf{X}}_{n+1} - \underline{\mathbf{X}}_n$

2 Task 2 Modelling

Now that the second model has been chosen to analyze the system it is required to use the model to predict the next state of the system without using the *performAction()* function. Initially, a linear model is analyzed and then a non-linear model.

2.1 Task 2.1 Gathering Data

For the first step of the procedure it is required to gather data points in order to form a linear regression. The function *data(n)* finds $\underline{\mathbf{Y}}$ after one *performAction()* function for n random initialisations of the $\underline{\mathbf{X}}$ variable. The outputs of the function are 2 matrices $\underline{\underline{\mathbf{X}}}$ and $\underline{\underline{\mathbf{Y}}}$ where each random state $\underline{\mathbf{X}}_i$ is and each change of state $\underline{\mathbf{Y}}_i$ represented as a column vector.

2.2 Task 2.2 Linear Regression

Using the data points gathered in the 2 matrices $\underline{\underline{\mathbf{X}}}$ and $\underline{\underline{\mathbf{Y}}}$. A linear regression can be fit $\underline{\underline{\mathbf{Y}}} = \underline{\underline{\mathbf{W}}}^T \underline{\underline{\mathbf{X}}}$ to predict the change in state for a given state where $\underline{\underline{\mathbf{W}}}$ is a 4x4 matrix which contains coefficients of the optimal model. It can be shown that the linear regression solution is just equal to the least squares solution(check *Appendix B* for full derivation) where $\underline{\underline{\mathbf{W}}} = (\underline{\underline{\mathbf{X}}}\,\underline{\underline{\mathbf{X}}}^T)^{-1}\underline{\underline{\mathbf{X}}}\,\underline{\underline{\mathbf{Y}}}^T$.
The function *np.linalg.inv* was initially considered to being used to compute $\underline{\underline{\mathbf{W}}}$ but as it was discovered in the previous section $\underline{\mathbf{Y_i}}$ does not depend on the position of the cart and therefore the matrix $\underline{\underline{\mathbf{X}}}\,\underline{\underline{\mathbf{X}}}^T$ is singular. This problem is overcomed by using the function *np.linalg.lstsq* which computes the least squares solution automatically and it deals with singular value matrices. To perform an initial test of the linear model a new set of data points are gathered by taking a 1D scan of the system(with respect to the angle)$\underline{\underline{\mathbf{X}}}_{test}$ and $\underline{\underline{\mathbf{Y}}}_{test}$ is produced. Using the least squares solution the $\underline{\underline{\mathbf{Y}}}_{predict} = \underline{\underline{\mathbf{W}}}^T \underline{\underline{\mathbf{X}}}_{test}$ is found. In the *Figure 4* below the two solutions are compared. For better detailed comparison between real system and the predicted system please check *Appendix C*

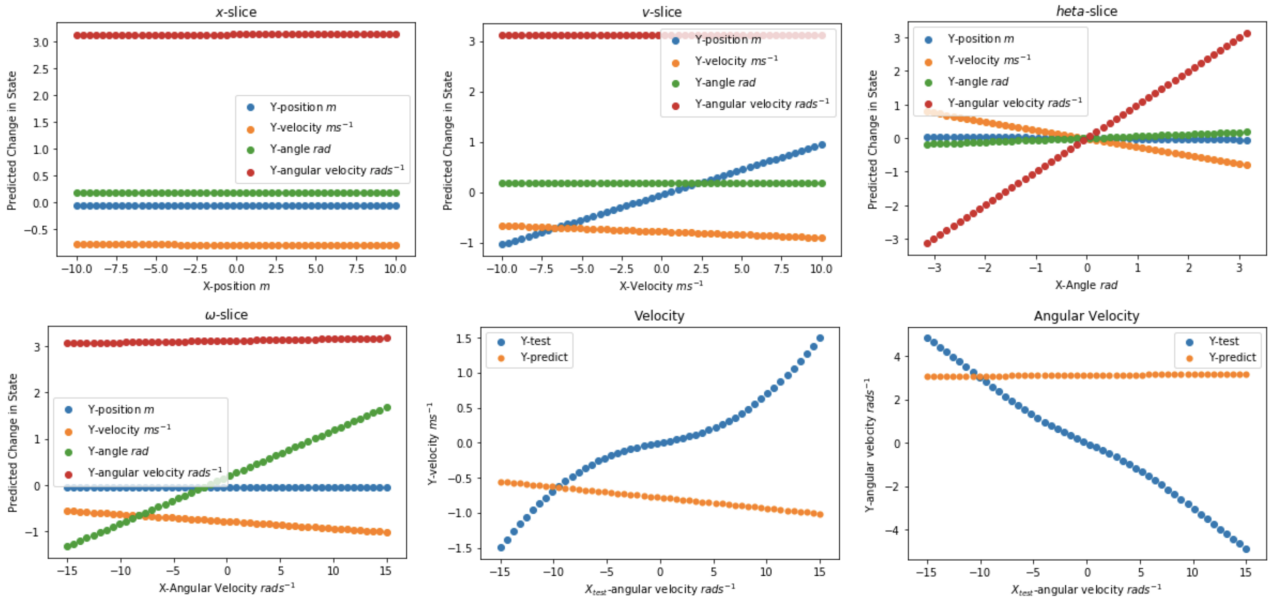


Figure 4: Comparison between $\underline{\underline{\mathbf{Y}}}_{test}$ and $\underline{\underline{\mathbf{Y}}}_{predict}$ using 1D scans from $\underline{\underline{\mathbf{X}}}_{test}$ along with a closer look at 2 of previous worse linear prediction found by 1D scan along v and ω

Comparing *Figure 3* and *Figure 4* it can be observed the linear regression does not fit very well the model of the system. This is because the model of the system contains non-linearities. For 1D scans predictions of position(x) and velocity(v) the linear regression almost correctly predicts the model because these variables evolve linearly as expected. However, the regression lines seemed to be shifted in the y-axis by a small value. This could be because the 1D scan is taken from a uniform distribution corresponding to one variable while $\underline{\underline{\mathbf{W}}}$ is fitted on a uniform distribution in all directions.

2.3 Task 2.3 Testing the Linear Prediction

The true test is to see if the linear regression correctly predicts the change of state is to evaulate the dynamics of the system from an initial state. The prediction of the dynamics is found by taking a random initialization point $\underline{\mathbf{X}}_0$ and the next state is found by $\underline{\mathbf{X}}_1 = \underline{\underline{\mathbf{W}}}^T \underline{\mathbf{X}}_0 + \underline{\mathbf{X}}_0$. By applying this in an iterative way for n times the system state after $t = n * 10s$ can be predicted: $\underline{\mathbf{X}}_n = \underline{\underline{\mathbf{W}}}^T \underline{\mathbf{X}}_{n-1} + \underline{\mathbf{X}}_{n-1}$. The solution to the predictive dynamics is compared to the true dynamics which corresponds to *Task 1.1* solution. The results is shown in *Figure 5* below for an for a given initial state($X(state) = [0, 0, -0.641, 0]$).

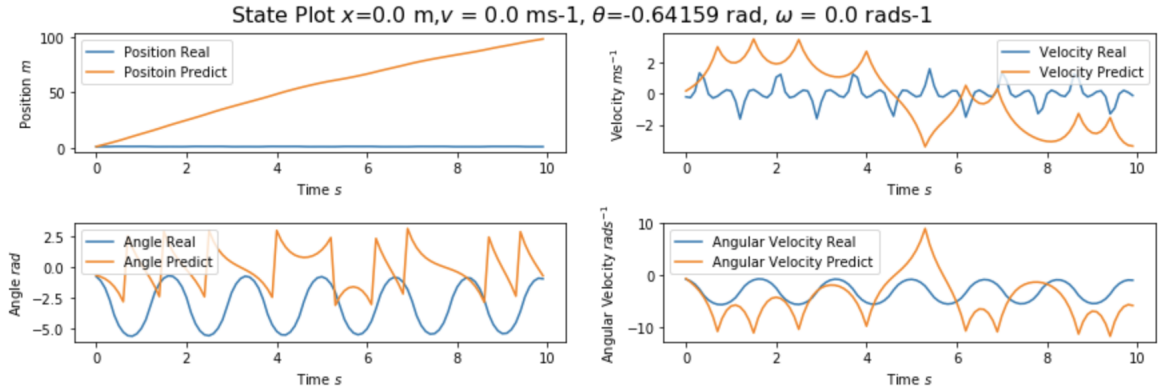


Figure 5: Prediction of the dynamics using linear regression compared to the actual dynamics of the system for a time $t = 10s$

As observed the model diverges very quickly. In the *Appendix C*, it can be seen that if the dynamics are run for a longer period of time $t = 50s$ the model diverges by a big amount. The only parameter that doesn't diverge is the angle θ because of the *remapangle()* function. However, the other variables will continue to increase linearly because of the linear fit model observed in the *Task 2.2*. By considering another initial state $X(state) = [0, -8.0, -641, 0]$ the predicted dynamics perform even worse(*Appendix C*)

2.4 Task 2.4 Non-Linear Model

2.4.1 Building the Model

As the linear model fails to predict the other variables required to predict the full dynamics of the system, a non-linear model is considered. The model function is:

$$\underline{\mathbf{Y}} = \underline{\underline{\mathbf{W}}}^T \mathbf{K}(\underline{\mathbf{X}}, \underline{\underline{\mathbf{X}}}_{\text{basis}}) \tag{1}$$

$\mathbf{K}(\underline{\mathbf{X}}, \underline{\underline{\mathbf{X}}}_{\text{basis}})$ is defined as a kernel matrix and $\underline{\underline{\mathbf{W}}}$ as the weights matrix for non-linear model. The kernel function is a radial basis function with the means positioned at the points contained in the matrix $\underline{\underline{\mathbf{X}}}_{\text{basis}}$. The basis function is evaluated at the $\underline{\underline{\mathbf{X}}}$ which contain the training data set. The Kernel matrix is equal to:

$$\mathbf{K}(\underline{\underline{\mathbf{X}}}, \underline{\underline{\mathbf{X}}}_{\text{basis}})_{ij} = e^{-\sum_a \frac{(\mathbf{x}_i^{(a)} - \mathbf{x}_{\text{basis}\,j}^{(a)})^2}{\sigma_a}} \tag{2}$$

Where $\underline{\mathbf{X}}_i^{(a)}$ refers to the ath component of $\underline{\mathbf{X}}_i$ state vector which corresponds to the ith column of $\underline{\underline{\mathbf{X}}}$. However, because the *angle* variable is periodic instead of having $(\theta_i - \theta_{\text{basis}\,j})^2$, the periodicity is taken into account and it is changed to $sin(\frac{\theta_i - \theta_{\text{basis}\,j}}{2})^2$. The number of mean points taken for $\underline{\underline{\mathbf{X}}}_{\text{basis}}$ is $\mathbf{M}$. The number of training data points for $\underline{\underline{\mathbf{X}}}$ is $\mathbf{N}$. The non-linear weights are found by finding the regularised least squares solution:

$$\underline{\underline{\mathbf{W}}} = (\underline{\underline{\mathbf{K}}}_{\mathbf{NM}}^T \underline{\underline{\mathbf{K}}}_{\mathbf{NM}} + \lambda \underline{\underline{\mathbf{K}}}_{\mathbf{MM}})^{-1} \underline{\underline{\mathbf{K}}}_{\mathbf{NM}}^T \underline{\underline{\mathbf{Y}}}^T \tag{3}$$

Where:

- $\underline{\underline{\mathbf{K}}}_{\mathbf{NM}}$ is the kernel matrix evaluatated at $\mathbf{K}(\underline{\underline{\mathbf{X}}}, \underline{\underline{\mathbf{X}}}_{\text{basis}})$
- $\underline{\underline{\mathbf{K}}}_{\mathbf{MM}}$ is the kernel matrix evaluatated at $\mathbf{K}(\underline{\underline{\mathbf{X}}}_{\text{basis}}, \underline{\underline{\mathbf{X}}}_{\text{basis}})$,further discussed in *Section 2.4.5*
- λ is a parameter which can be varied that adjusts how much of $\underline{\underline{\mathbf{K}}}_{\mathbf{MM}}$ is added.

2.4.2 1D-scans

To find $\underline{\underline{\mathbf{W}}}$ the function *np.linalg.lstsq* is used which solves equation of the type $\underline{\underline{\mathbf{A}}}\,\underline{\mathbf{x}} = \underline{\mathbf{b}}$. By making $\underline{\underline{\mathbf{A}}} = \underline{\underline{\mathbf{K}}}^T_{\mathbf{NM}}\underline{\underline{\mathbf{K}}}_{\mathbf{NM}} + \lambda\underline{\underline{\mathbf{K}}}_{\mathbf{MM}}$ and $\underline{\mathbf{b}} = \underline{\underline{\mathbf{K}}}^T_{\mathbf{NM}}\underline{\mathbf{Y}}^T$, $\underline{\underline{\mathbf{W}}}$ can be found using the *numpy* function. By performing a 1D scan as in *Task 2.3* the non-linear model is tested. The results of 1D scan for $\mathbf{M} = 600$ and $\lambda = 2e-4$ are shown in the *Figure 6* below.

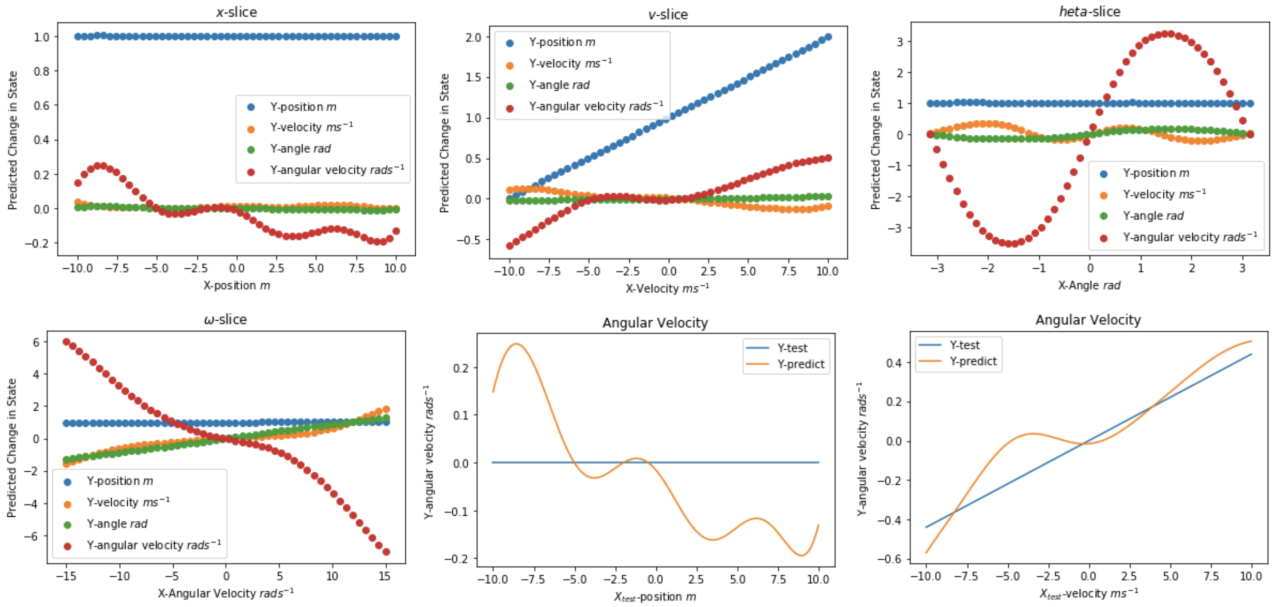


Figure 6: Comparison between $\underline{\underline{\mathbf{Y}}}_{test}$ and $\underline{\underline{\mathbf{Y}}}_{predict}$ using 1D scans with $\mathbf{M} = 800$ and $\lambda = 2e-4$ from $\underline{\underline{\mathbf{X}}}_{test}$ along with a closer look at 2 of the worse non-linear prediction found by 1D scan along x and v

As expected the non-linear models correctly predicts the 1D-scan along the *Angle* and *Angular Velocity* as seen from the *Figures 3* and *6*. This was expected as both of those variables are non-linear. However the problem with the non-linear model is shown in the last two figures of *Figure 6*. The non-linear model fails to predict the change in position and therefore the error in angular velocity is significant. The error in the position is very big because the non-linear model is trying to predict the other variables of the system from the position where it is. Knowing that the all other variables are independent of the position of the cart the non-linear model will result in variables diverging away from the real model. Because of this reason and the fact that the position is linear this variable could be left out of the non-linear model and use the linear model to predict the position as it is still desired to have a good prediction of the position.
Looking at the last figure from *Figure 6* the 1D scan along the velocity has errors in the non-linear prediction. This time these errors can be reduced by taking more data points(increasing $\mathbf{N}$) and increasing the number of basis functions(increasing $\mathbf{M}$). The reason for this is the fact that as more points and basis functions are used the better the linear combinations of basis functions which is used to find the predicted solution.

2.4.3 Roll-Out

To the true test of the non-linear prediction a roll-out test is performed. The results can be observed in *Figure 7* below from an initial state, $X(state) = [0, -2, 0.641, 8]$ with $\mathbf{M} = 800$ and $\lambda = 5e-5$. The

results for to other initial states can be observed in the *Appendix D*

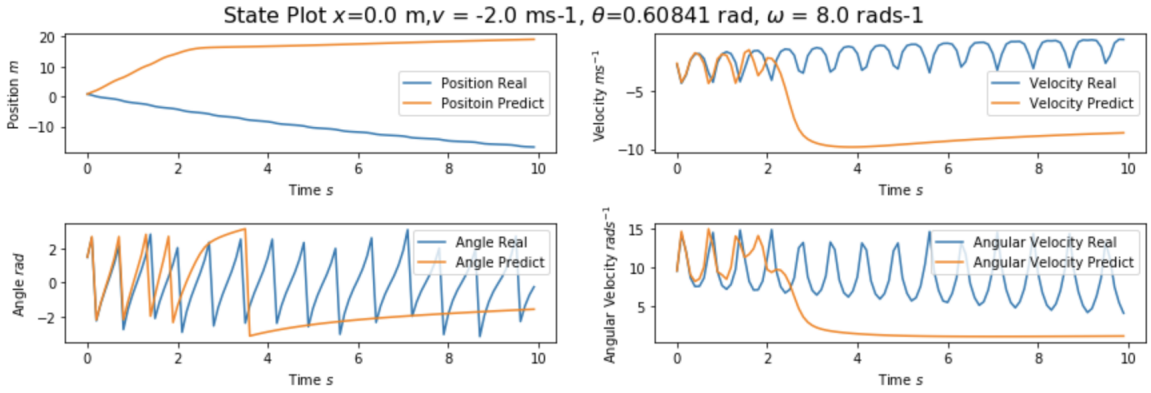


Figure 7: Prediction of the dynamics using non-linear regression compared to the actual dynamics of the system for a time $t = 10s$

As expected the non-linear model is better than the linear model but only work predicts correctly the model for couple of seconds before it diverges. As expected the *position* variable has the worst errors. Even if initialised in any other state the predicted dynamics will be the same as the real dynamics again only for only a couple of seconds(see *Appendix D*).

2.4.4 Task 2.5 Non-Linear + Linear Model

As remarked in the previous sections, both linear and non-linear model have some limitations. What if they are combined? In theory it desired to find the *Position* variable by the linear model so that the other variables will not be dependent on the position and the rest of the variables by non-linear model. This way the errors cause by the position are avoided and the variables will not diverge after a couple of seconds. The results of the 1D-scans with non-linear and linear model are seen in the *Figure 8* below. For more detailed comparison please check *Appendix E*:

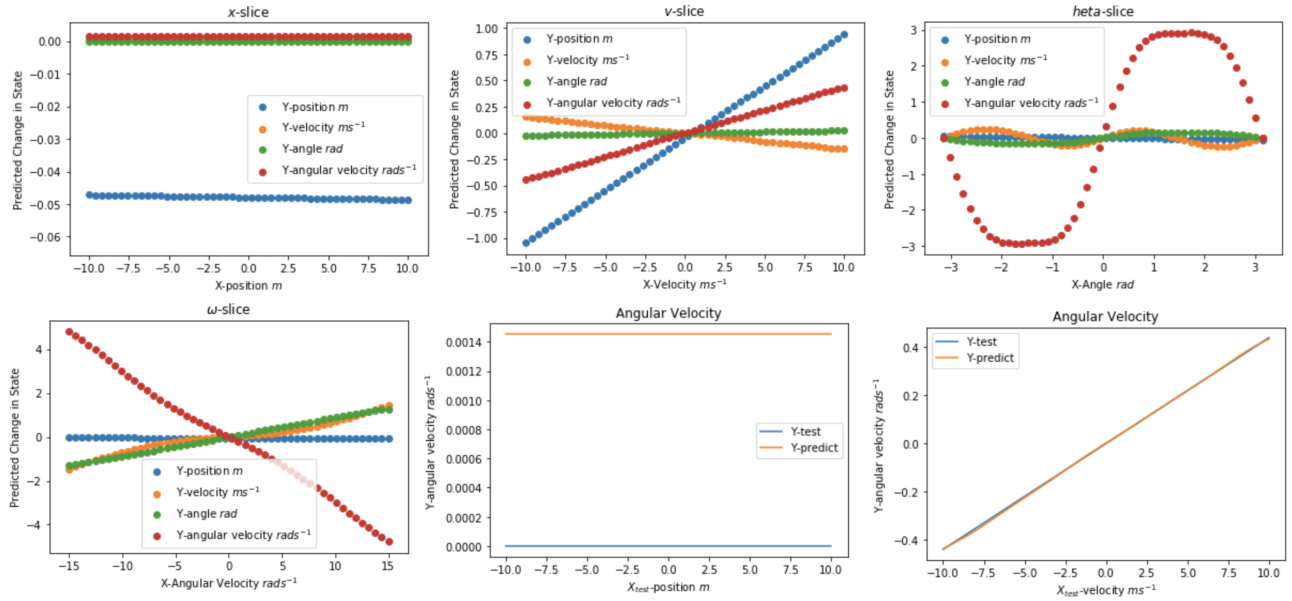


Figure 8: Comparison between $\underline{\underline{\mathbf{Y}}}_{test}$ and $\underline{\underline{\mathbf{Y}}}_{predict}$ using 1D scans with $\mathbf{M} = 800$ and $\lambda = 2e - 4$ from $\underline{\underline{\mathbf{X}}}_{test}$ along with a closer look at 2 of previous worse non-linear prediction found by 1D scan along x and v

Compared with *Figure 3* it can be observed that the 1D scans of the linear + non-linear are very accurate.

The next test is to check the how the roll-out performs. By taking the same initial state as in the non-linear case ($X(state) = [0, -2, 0.641, 8]$) the result of the roll-out can be seen in *Figure 9* below.

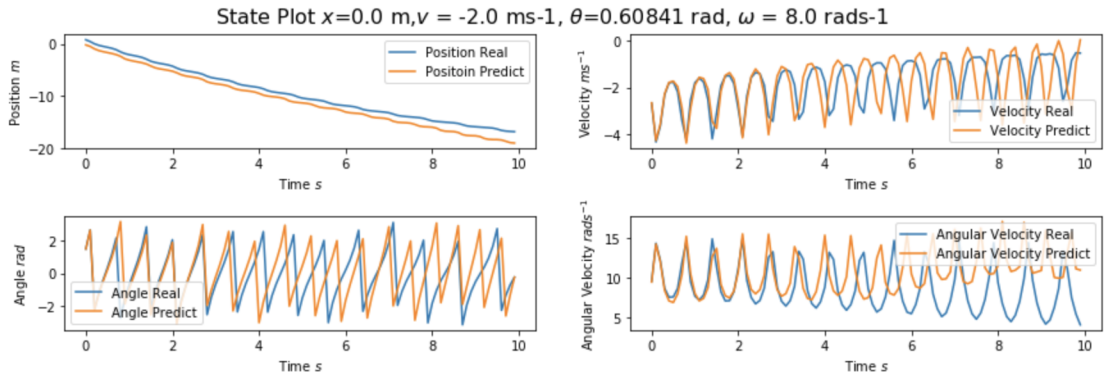


Figure 9: Prediction of the dynamics using linear + non-linear regression compared to the actual dynamics of the system for a time $t = 10s$

As expected the predicted dynamics of the model are much better compared to the other models. For the other roll-outs with different initial conditions can be seen in *Appendix E*. A further improvement on the accuracy can be found by taking more data points and basis functions($\mathbf{N} = 5000$,$\mathbf{M} = 2500$ and $\lambda = 5e - 5$). More data points and more basis function decreases the change that there could be a value in the hyper-cube which cannot be interpolated accurately.The result can be seen in *Figure 10*.Becaue of the interpolation error it is expected that at extreme values such as $v = 10\text{ms}^{-1}$, the non-linear + linear model will not work. On the account of this, at such extreme values there is not enough data points to be interpolated to good approximated value so the predicted model diverges.Having a good model to predict the dynamics allows to move to next chapter of the project involving the control of the system but there is still a question of what hyperparameters to choose.

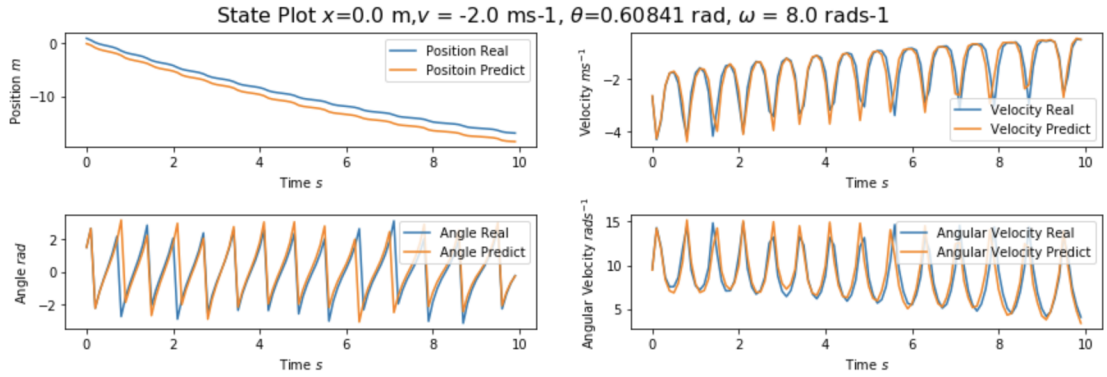


Figure 10: Prediction of the dynamics using linear + non-linear model for a larger a amount of dataset compared to the actual dynamics of the system for a time $t = 10s$

2.4.5 Hyperparameters

There are 2 parameters which can be varied to produce different non-linear weights, $\mathbf{M}$ and λ. In the *Figure 11* below the *Root Mean Square Error*(RMSE) has been calculated using a test dataset. Firstly, a 1D-scan have been performed, initially by keeping λ constant and varying $\mathbf{M}$ and then vice-versa.Secondly a grid search has been performed using a range of values for $\mathbf{M}$ and λ.

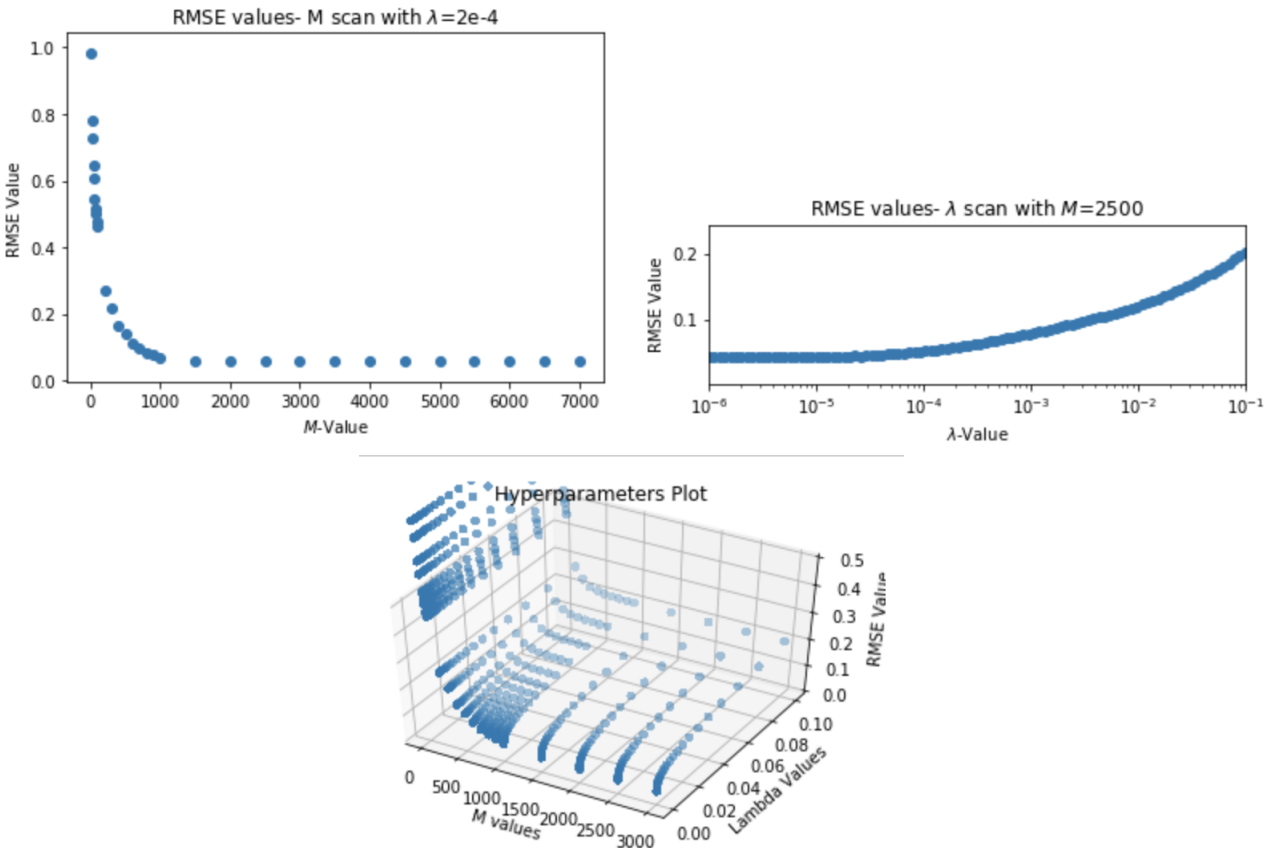


Figure 11: Grid Search for hyperparameters $\mathbf{M}$ and λ

In theory λ can be thought of like a parameter which controls how much of "prior"($\underline{\underline{\mathbf{K}}}_{\mathbf{MM}}$) should be taken into account. If there are two Gaussian with the center close together of opposite value it would be expected to see a sharp transition in the prediction. By using λ these situations are penalised so that a smooth function would be observed instead. From *Figure 11* it is observed that as lambda gets smaller the accuracy of the model improves until it saturates at $\lambda \approx 3e-5$. Further increase of λ from this point has an insignificant improvement and it would stop penalising the opposite Gaussians which is still desirable. $\mathbf{M}$ varies the number of means basis function data points. Increasing $\mathbf{M}$ it would be expected that the accuracy of non-linear model to increase.However, at a certain value ,$\mathbf{M} \approx 200$, the accuracy will saturate. Further increase in $\mathbf{M}$ does not bring any improvement in RMSE. From the last figure of *Figure 11* it can observed that by increasing $\mathbf{M}$, λ becomes less and less important in providing an accurate model. There could be a third hyperparameter which could be varied. It is the variance of the variables σ_j. Making the variance very small the Gaussians become delta functions which is not desirable but making it very big reduces the non-linearity performance. By making σ_j equal to the variance of the training dataset it produces a good non-linear + linear model.

3 Task 3 Control

The purpose of the this is task is to perform control of the predicted model which would balance the upright starting from the initial position where the pole is equilibrium position. Therefore a force is introduced, that is applied to the cart, only in a horizontal direction.

3.1 Task 3.1 Adapting Model for Force

3.1.1 Real Dynamics

For the model to be retrained to account for the force it is necessary to observe how the real system behaves when a force is applied to the cart.The force that can be applied to the cart is between the range [−maxF,maxF] where $maxF = 20$. However because of the tanh function the action force that is actually accounted for as applied to the cart is between [−2maxF,2maxF].The best way to remodel the system is to look again at the change in state which it is required to predict. To see how each variables depends on the *Force*, a 1D-scan is applied to the system with varying the force F. The scan is displayed in the *Figure 12* aside. No other scans were displayed because they would look exactly the same as the one in *Figure 3* for which good model has already been achieved. In *Figure 12* it is observed that the all of the variables varies non-linearly with the Force. Therefore the non-linear + linear model that was used previously cannot be used anymore as position will vary quadratically. Therefore it is expected that only a non-linear model would be able to model the system correctly. To further explore how to the force acts on the system the Roll-Out Test must be performed as well. The system is initialised from an equilibrium with a small force. The Roll-Out test for the a small action force can be seen in *Figure 13* below:

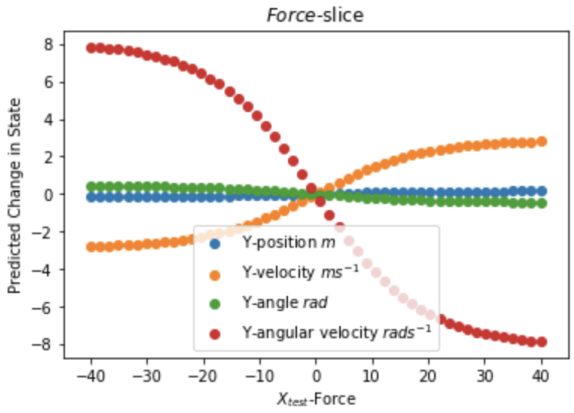


Figure 12: 1D Scan of the Force

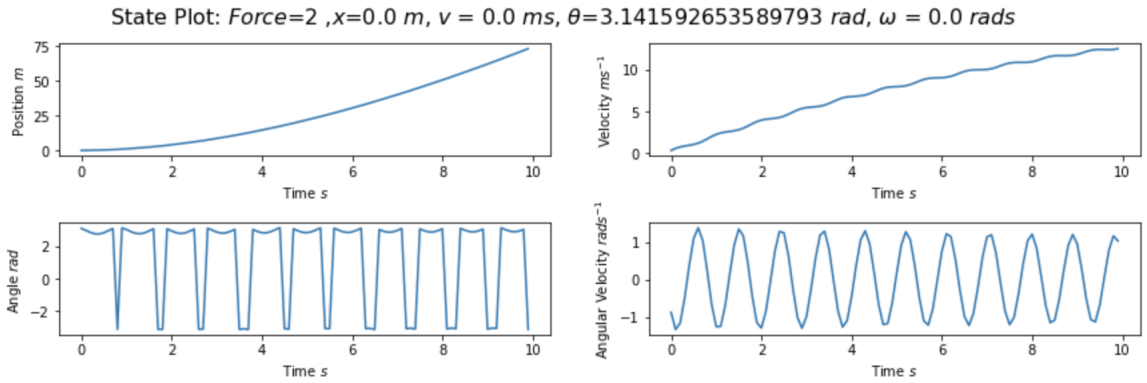


Figure 13: Roll-Out of a system when a force $F = 2$ is applied throughout the time evolution

3.1.2 Non-Linear Model

As mentioned before, the *Force* causes non-linearities to occur, as a result, only a non-linear model is considered. This time one the number of inputs changes from 4 to 5 as the *Force* is included but the same method is applied as in *Section 2.4.1*.As discussed in the previous section for a good predicted solution,

a lambda value was set to $\lambda = 5e-5$, $N = 5000$ data points and the number of basis functions equal to $M = 2500$. To test how good the non-linear model is compared to the real dynamics the 1D scan with respect to *Force* and the Roll-Out test is performed. The results are shown in the *Figure 14* below:

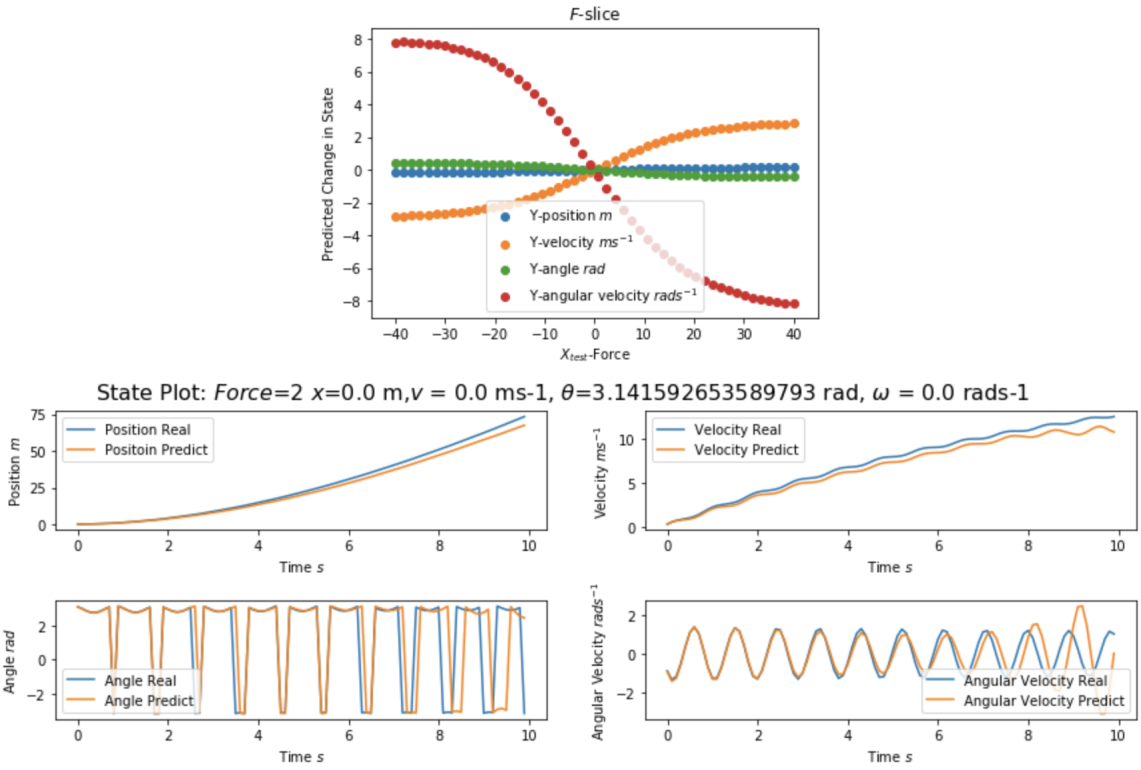


Figure 14: 1D scan with respect to force and Roll-Out of a system when a force $F = 2$ is applied throughout the time evolution

It is observed that the non-linear model correctly predicts the system time evolution. However in *Appendix F* a Roll-Out test with a large action force applied to the cart is tested on the non-linear model. As expected the model behaves very well until the velocity goes bigger than $v = 10\text{ms}^{-1}$. At such an extreme value the non-linear model fails to predict well because the basis function have been taken from a range of $[-10, 10]$. Therefore the non-linear model cannot interpolate correctly at high values of action force.

3.2 Task 3.2 Linear Policy

3.2.1 Theory

In order to control the pole it is required to define a *policy* $p(\underline{\mathbf{X}})$ that defines what the action should be given to the current state. For the purpose of this system the policy is equal to the action force applied to the cart.
Initially only a linear policy is considered:

$$p(\underline{\mathbf{X}}) = \underline{\mathbf{p}}^T \underline{\mathbf{X}} \tag{4}$$

Where $\underline{\mathbf{p}}$ is a 4-dimensional vector with unknown coefficients.
The next step is define an *objective function*(*loss function*) which is a function that contains what it is desired to achieve by this *policy*. For the objective defined by the project it is required to place at the pole at $\theta = 0°$ and stay there, preferably with a very small angular velocity ω. Therefore one possible *objective*

function(L) for given state $\underline{\mathbf{X}}_i$ is given below:

$$L = \underbrace{(1 - \exp{(-\frac{\theta_i^2}{2\alpha^2})})}_{\text{Angle Loss}} + \underbrace{(\exp{-\frac{\theta_i^2}{2\alpha^2}})\omega^2\beta}_{\text{Angular Velocity Loss}} \tag{5}$$

To check how well the *Objective Function* is performing a contour plot is helpful to visualise this function. The plot is below in *Figure 15*:

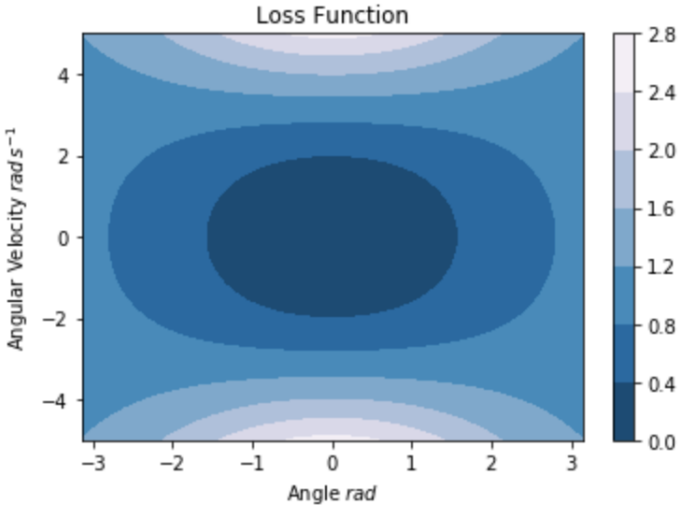


Figure 15: *Objective Function* contour plot

From *Figure 15* it is clearly seen that the loss function does what it is required. It is also noticed that the angular velocity ω is less penalised than the angle θ which is again desirable.

3.2.2 1D and 2D scans of $\underline{\mathbf{p}}$

Now that the *Policy* and the *Objective Function* have been defined it is time to look how these two depend on each other. Since vector $\underline{\mathbf{p}}$ has 4 coefficients, a solution is to look at 1D and 2D scan of these coefficients with respect to a the *Total Loss* of a Roll-Out from an initial state which would be close to $\theta = 0°$. A good initial state is: $x = 0$, $v = 0$, $\theta = 0.4$ and $\omega = 0$. The *Total Loss* is computed by finding the *loss* at each state throughout the time evolution of the Roll-Out for time $t = 5$ seconds. The plots for the 1D scan are displayed below in *Figure 16*:

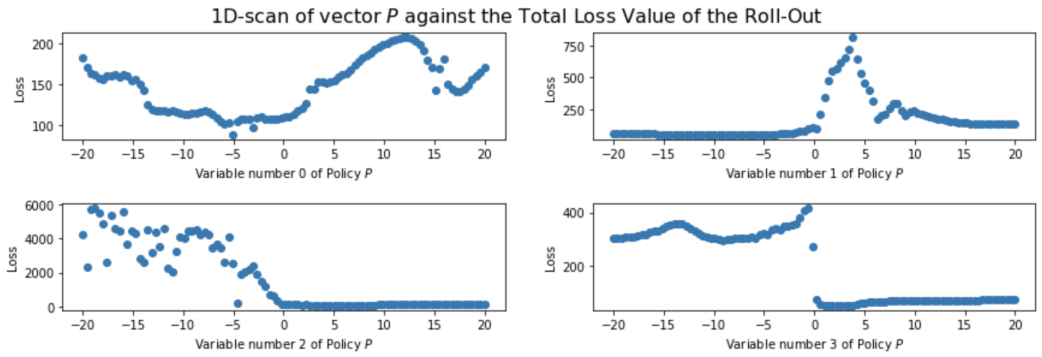


Figure 16: 1D scan of coefficients of $\underline{\mathbf{p}}$

Looking at *Figure 16* it is hard to to spot some good coefficients values which give a low *Total Loss*. However from the figure it is deduced and intuitively it makes sense that Variable number 2 is of $\underline{\mathbf{p}}$ is the most important as this coefficient gives a force which is proportional to the angle θ of the pole. Also, intuitively it is expected that the Variable number 0 play has the least important role as this this coefficient gives a force which is proportional to the position x of the cart which is not what it is desired at all. Therefore it is sensible to make this coefficient equal to 0.
To find more information about the other 3 variables of $\underline{\mathbf{p}}$ a 2D scan of these coefficients should be investigated. The results are seen in the *Figure 17* below.

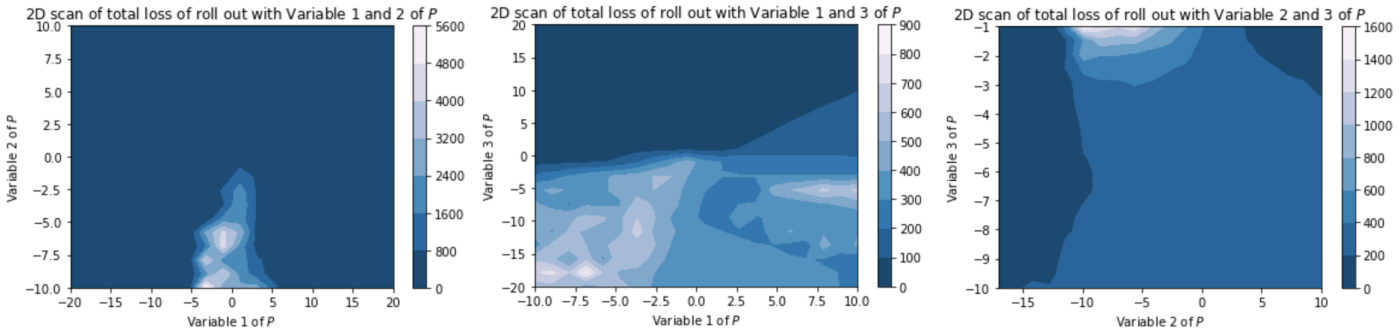


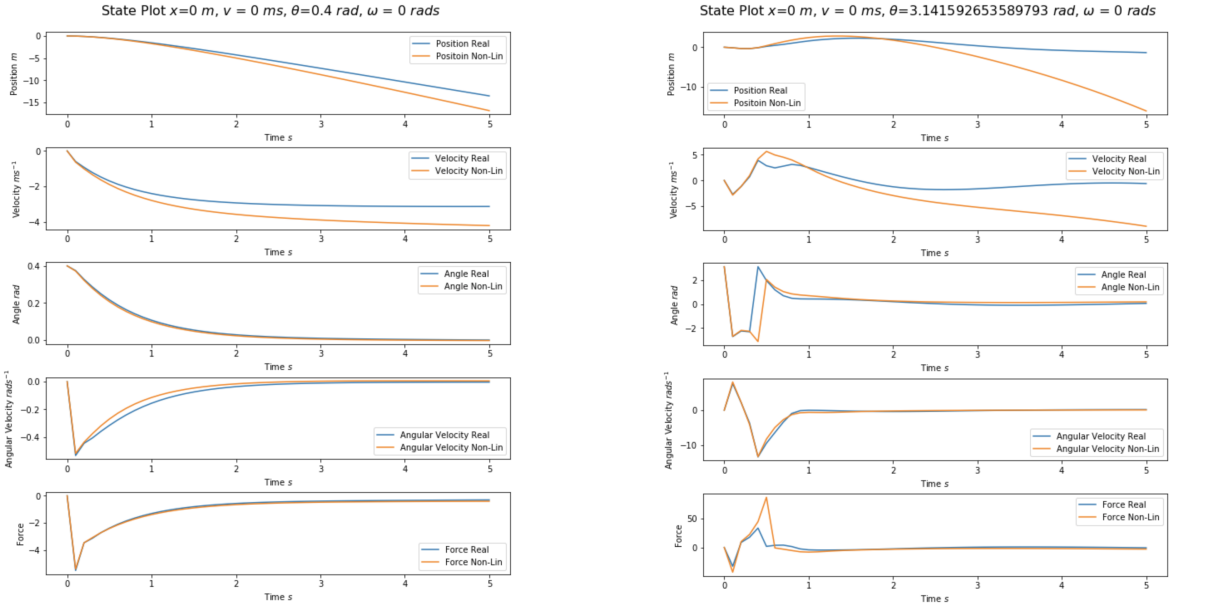
Figure 17: 2D scan of coefficients of $\underline{\mathbf{p}}$

From the 2D-scan results and again from an intuitive thinking the coefficient 1 of $\underline{\mathbf{p}}$ should be close but equal to 0 as velocity v should not be able to have a big influence on the force applied to the cart. Therefore a good place to look at the starting coefficients for variable 2 and 3 is the last RHS figure from *Figure 17*. It can be seen that a good starting value for these coefficients would be: $p_2 = -15$ and $p_3 = -5$. Consequently, the initial value of $\underline{\mathbf{p}}$ is: $\underline{\mathbf{p}}_{init} = [0, 0, -15, -5]$

3.2.3 Control Real Dynamics vs Non-Linear Model Dynamics

Now that, a good initial solution for $\underline{\mathbf{p}}$ has been been found, it is now required to perform an optimisation to find the best values for the unknowns in $\underline{\mathbf{p}}$ that would give a minimum *Total Loss Value* of a Roll-Out for a given $\underline{\mathbf{p}}_{init}$, at a time t and an initial state. Because for our model the gradient is not available, *Nelder-Mead* optimisation method is chosen.It is a direct search method that uses the concept of a simplex(a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions). The algorithm is trying to minimise simplex by adding and removing the variables that form the simplex shape so that the function gets continously minimised([Wik]). For this system the simplex is a tetrahedral pyramid. However, the Nelder–Mead technique is a heuristic search method that can converge to non-stationary points. This is why the initial simplex is very important. A too small initial simplex can lead to a local search, therefore depending on the nature of the problem the initial simplex should be defined accordingly so that the *Nelder-Mead* method will converge. As discussed in the previous section a good starting simplex is $[0, 0, -15, -5]$.
Having now, estlabished a good optimising method with predifined intial conditions, it is now applied both to the real dynamics and the non-linear model predicted dynamics. At first an initial state which is not far away from the $\theta = 0°$ such as state $= [0, 0, 0.4, 0]$ is used and for a short period of time $t = 5s$. The results can be seen in *Figure 18a* below. It can be observed that the both the real system and the non-linear model system stabilises at $\theta = 0°$ after $t \approx 1s$. The next step is to see if both models will stabilise from the equilibrium initial state: state $= [0, 0, \pi, 0]$. The results are shown in *Figure 18b*. Analysing the figure, it is noticed that the real-dynamics control works perfectly with the pole being balanced at $\theta = 0°$. The non-linear model performed relatively well achieving the its objective but with some observable differences compared to the real dynamics. The reason why it deviates away from the real model is the fact that

at action forces higher than 50 there are no points from which it can interpolate as the sobol grid was defined up to 50. Therefore as soon as the action force is higher than 50 all the other parameters deviates from the real model. Because of the optimisation constrained the force is quickly adjusted to make the angle θ and angular velocity ω approach 0. Due to this fact the force is now is readjusted to values within the sobol grid and the non-linear model continues to approximate really well the real model for the *angle* and *angular velocity parameters*. But because the optimisation is not trained to make *velocity* small, it will continue to increase but since the other variables apart from *position* does not depend on the *velocity* unless it has a big gradient.



(a) $t = 5s$ and initial state $[0, 0, 0.4, 0]$ (b) $t = 5s$ and initial state $[0, 0, \pi, 0]$

Figure 18: Control Real Dynamics vs Non-Linear Dynamics

It is expected that in the long term ($t = 50$) the velocity will saturate at $v = 10$ because non-linear model cannot interpolate at higher values. Because of this it is expected that in the long-run the non-linear will keep the pole upright while the real-dynamics will fail due to small oscillations in the action force which gradually increases the *angular velocity*. The resulting control dynamics for the long term Roll-Out are observed in the *Appendix H*. Examining these figures it is noticed that real system and the non-linear system behaves as expected.

The next step is find the limit of this linear *policy* model which has been implemented. It is expected that by initialising the cart away from the equilibrium position and away from the objective position the linear *policy* should fail to stabilise the pole at $\theta = 0°$.However even by intialising away from this points the linear *policy* model and optimisation could still find optimal values of $\underline{\mathbf{p}}$ for which the objective function is met. For example a particular bad initial point is: state $= [0, 2, \pi, 1]$.

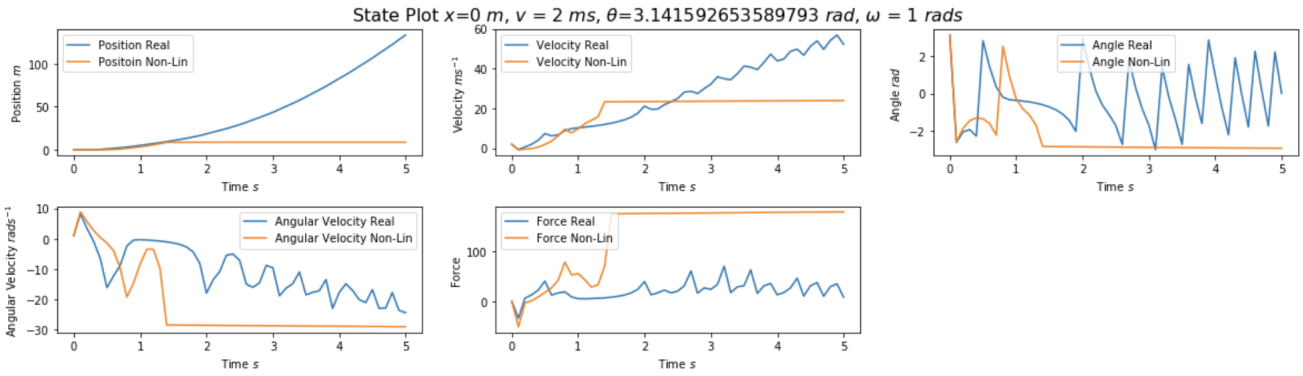


Figure 19: Unstable control of both Real System and Non-Linear System

It is noticed that as soon as the *Force* gets too big all the other variable for non-linear model are diverging. The long run test can be observed in the *Appendix H*. As expected the oscillation become worse and worse. To be able to control the system from these initial conditions a *non-linear policy* can be taken into account instead of the *linear policy* where the initial state is expended using the RBF function. Therefore, it is possible to optimise not only the magnitude of each, but also the location of Gaussian centres and the widths in each dimension.

4 Conclusion

- the real dynamics have been studied and it has been found the *Position* and *Velocity* vary linearly while *Angle* and *Angular Velocity* vary non-linearly.
- Initially, a linear model is used first to predict the dynamics of the system and as expected it could only predict well the linear parameters
- A non-linear is then used to predict the dynamics. The non-linear model worked well in predicting the *Angle* and *Angular Velocity* but because of the errors appeared in *Position* it could not be used and therefore a combination of non-linear and linear model is used
- When the action force is introduced it is observed that all parameters will vary non-linearly therefore the non-linear model would be a suitable to use as a prediction
- A linear policy is used to balance the poll upright. After the objective function is defined as the function to minimise the *Nelder-Mead* optimisation is used
- The results of the control for the real and predicted system are promising the pole being balanced upright from initial equilibrium position. However in the long-run the real model fails while the predicted model still holds the pole upright. A better solution would be to use a non-linear policy.

References

[Wik] Wikipedia. *Nelder–Mead method*. `https://en.wikipedia.org/wiki/NelderMead_method`. [Online; accessed 7-June-2019].

5 Appendix

5.1 Appendix A: Real Dynamics Roll-Out

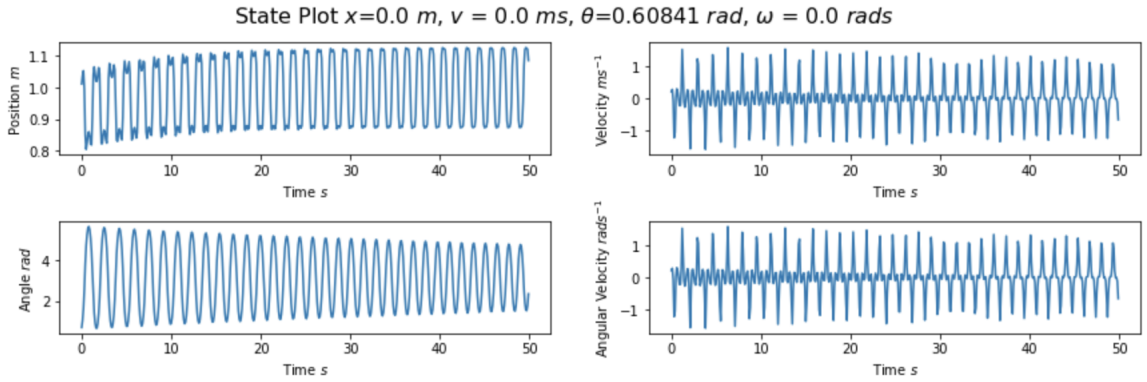


Figure 20: Dynamics of the system when the pendulum is released at $\theta = 0.608rad$

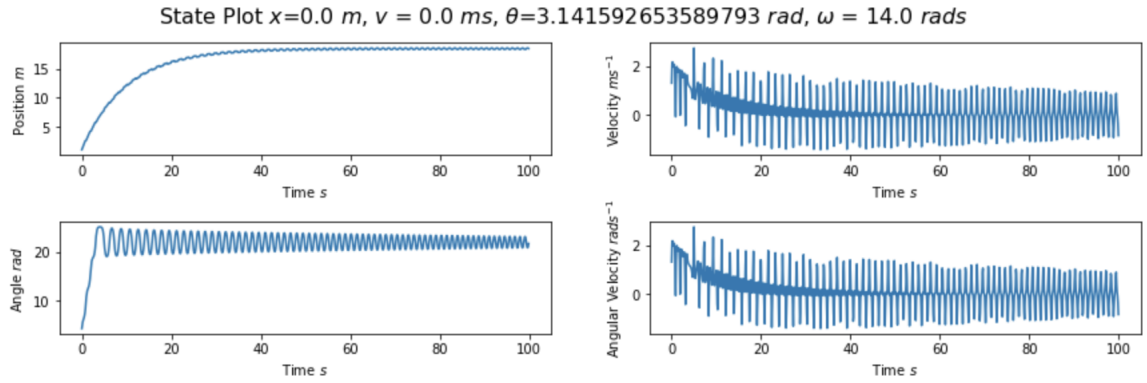


Figure 21: Dynamics of the system for complete rotation of the pendulum

5.2 Appendix B: Theoretical Proofs

Derivation of Linear Regression Model: $\underline{\underline{\mathbf{X}}}(M\text{x}N)$ is defined as matrix containing random state initialisations and $\underline{\underline{\mathbf{Y}}}$ $(K\text{x}N)$ as a matrix containing the change in state given the current initialisation. So the linear regression model is:

$$\underline{\mathbf{X}}_{i+1} = \underline{\mathbf{Y}}_i + \underline{\mathbf{X}}_i \,\text{where}\, \underline{\mathbf{Y}}_i = \underline{\underline{\mathbf{W}}}^T \underline{\mathbf{X}}_i \tag{6}$$

So in the linear regression model:

$$\underline{\underline{\mathbf{Y}}} = \underline{\underline{\mathbf{W}}}^T \underline{\underline{\mathbf{X}}} + \underline{\underline{\Omega}} \,\text{where}\, \underline{\underline{\Omega}} \sim \mathcal{N}(0, \sigma \underline{\underline{\mathbf{I}}}) \tag{7}$$

So the log likelihood($\mathcal{L}$) of the of the model is:

$$\begin{aligned} \mathcal{L} &= p(\underline{\underline{\mathbf{Y}}} | \underline{\underline{\mathbf{X}}}, \underline{\underline{\mathbf{W}}}) = \Sigma_{n=1}^{N} \ln(\mathcal{N}(\underline{\mathbf{Y}}_n | \underline{\underline{\mathbf{W}}}^T \underline{\mathbf{X}}_n, \sigma \underline{\underline{\mathbf{I}}}) \\ &= \frac{NK}{2} \ln(\frac{1}{2\pi\sigma}) - \frac{1}{2\sigma} \Sigma_{n=1}^{N} (\underline{\mathbf{Y}}_n - \underline{\underline{\mathbf{W}}}^T \underline{\mathbf{X}}_n)^2 \end{aligned} \tag{8}$$

Differentiating this *Equation 3* with respect to $\underline{\underline{\mathbf{W}}}$ gives:

$$\underline{\underline{\mathbf{W}}}_{ML} = (\underline{\underline{\mathbf{X}}} \, \underline{\underline{\mathbf{X}}}^T)^{-1} \underline{\underline{\mathbf{X}}} \, \underline{\underline{\mathbf{Y}}}^T \tag{9}$$

So it can be observed that the *ML* solution for the linear regression is equal to the least squares solution

5.3 Appendix C:Linear Model 1D-scans and Roll-Out

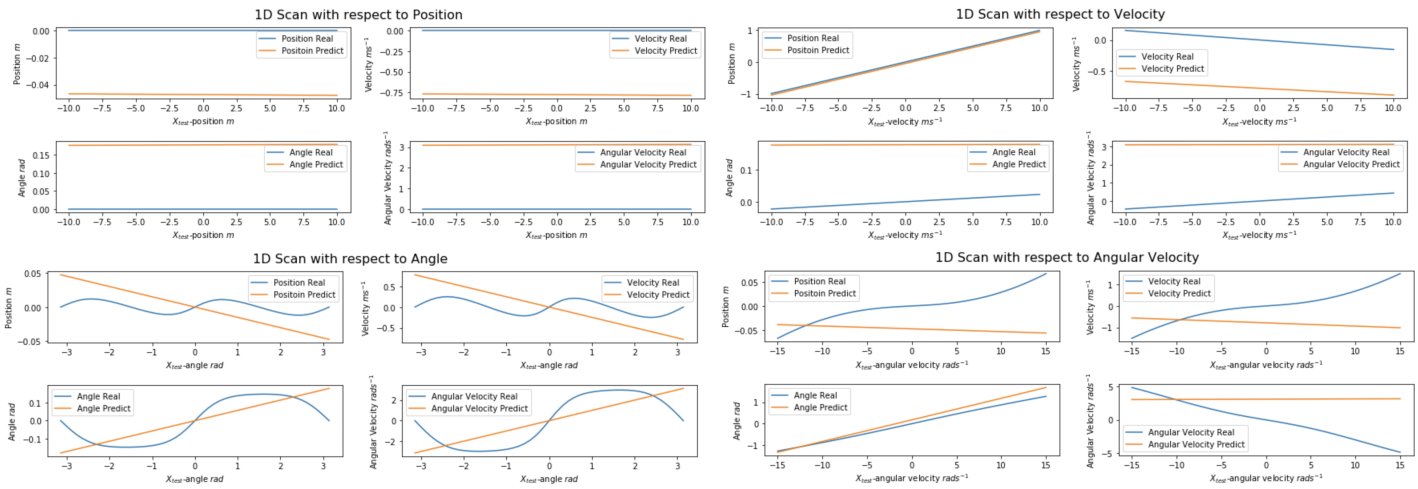


Figure 22: Detail Comparison between 1D-scan of the Real Model against the Linear Model

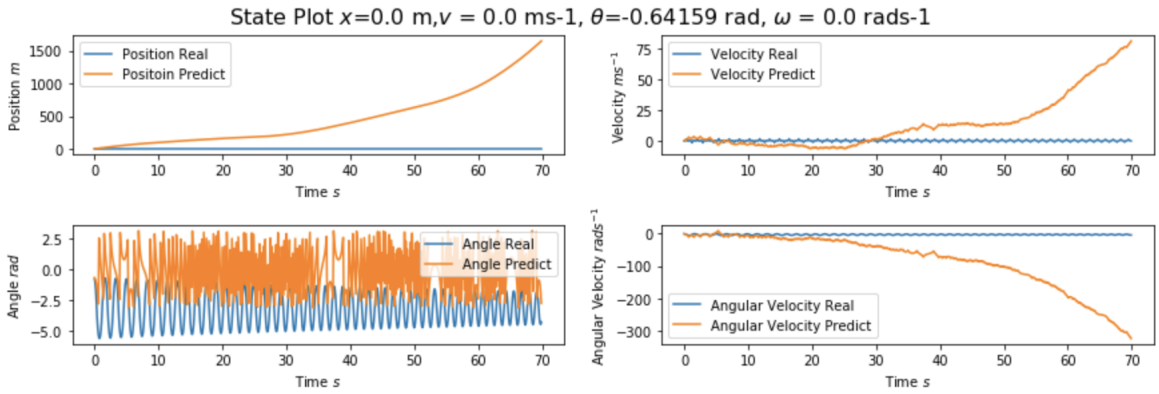


Figure 23: Prediction of the dynamics using linear regression compared to the actual dynamics of the system for a time $t = 70s$

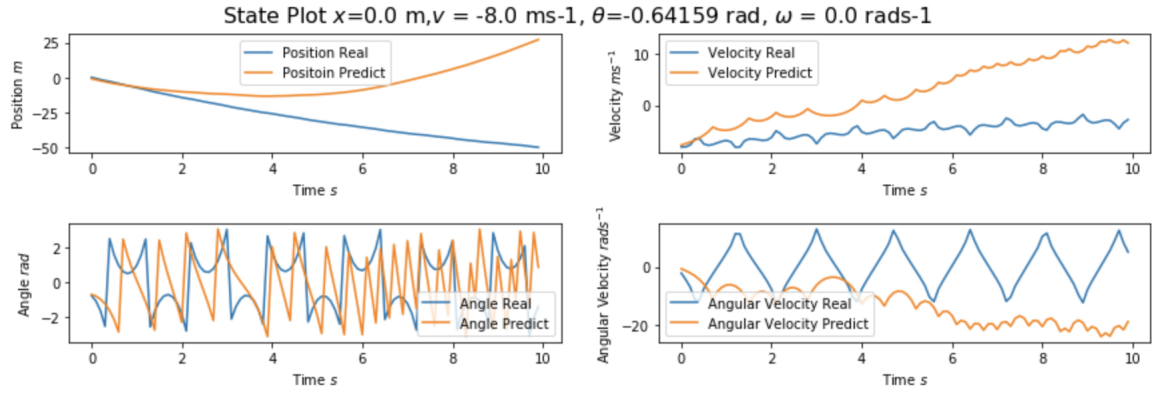


Figure 24: Prediction of the dynamics using linear regression from initial state $X(state) = [0, -8.0, -641, 0]$

5.4 Appendix D: Non-Linear Model 1D-scans and Roll-Out

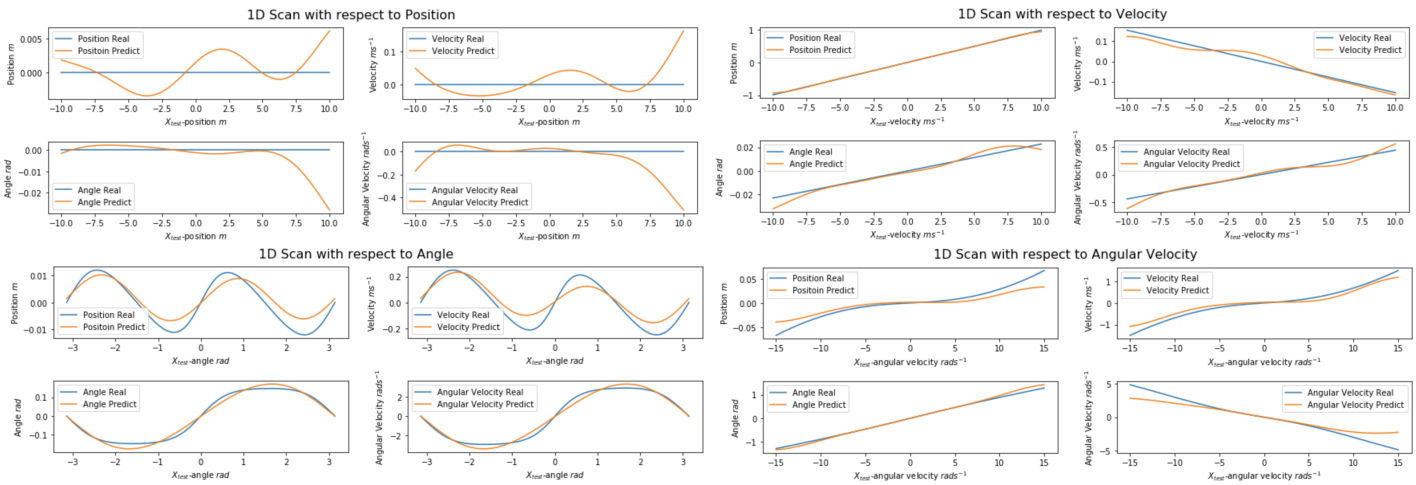


Figure 25: Detail Comparison between 1D-scan of the Real Model against the Non-Linear Model

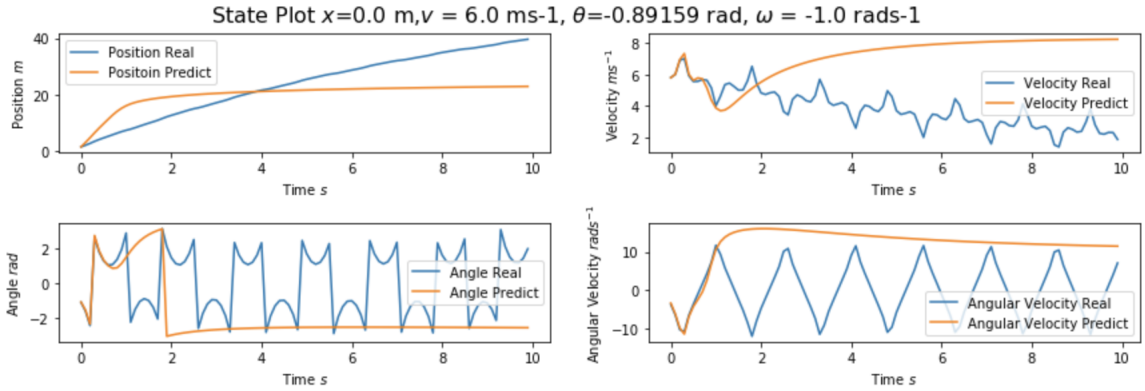


Figure 26: Prediction of the dynamics using non-linear regression from initial state $X(state) = [0, 6, -0.892, -1]$

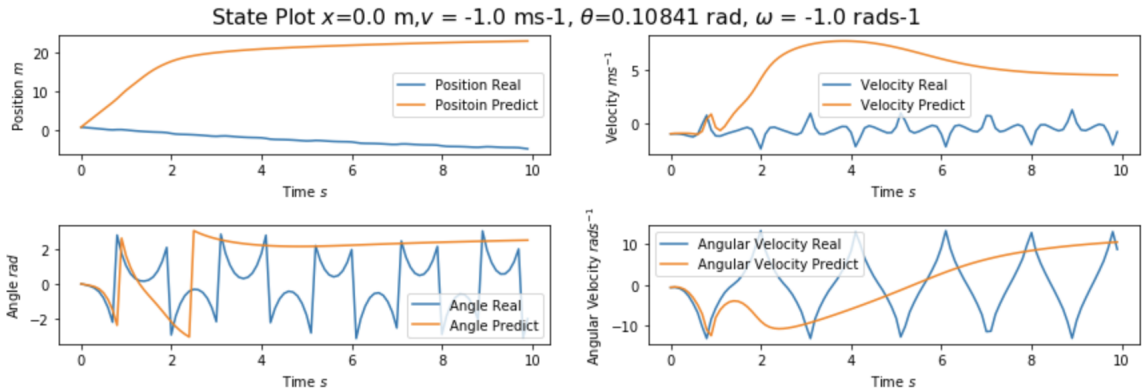


Figure 27: Prediction of the dynamics using non-linear regression from initial state $X(state) = [0, -1, 0.108, -1]$

5.5 Appendix E: Non-Linear + Linear Model 1D-scans and Roll-Out

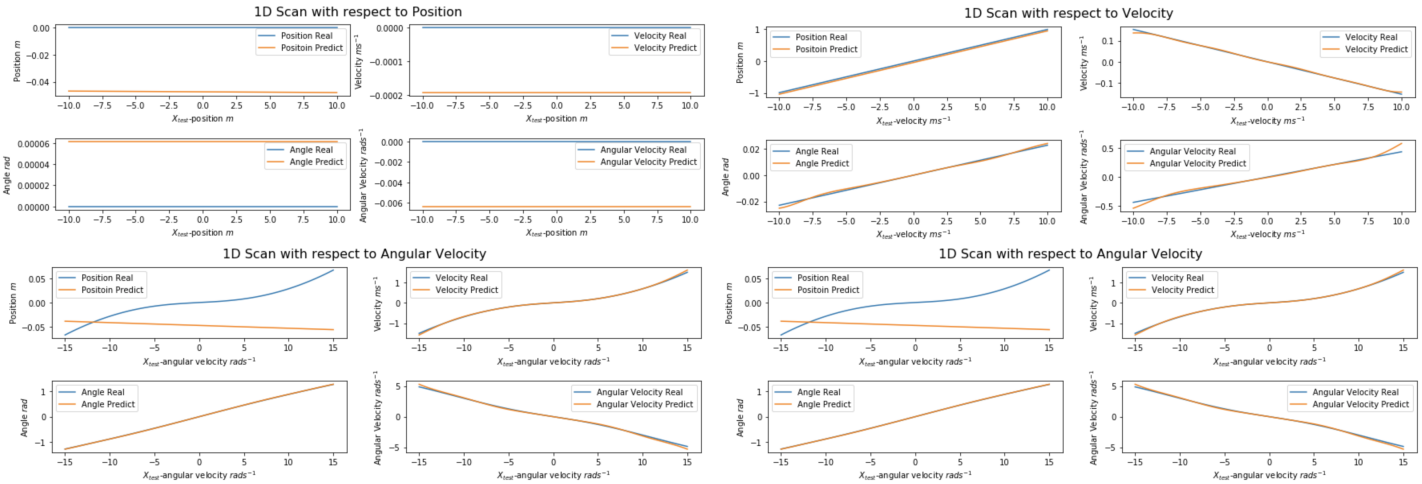
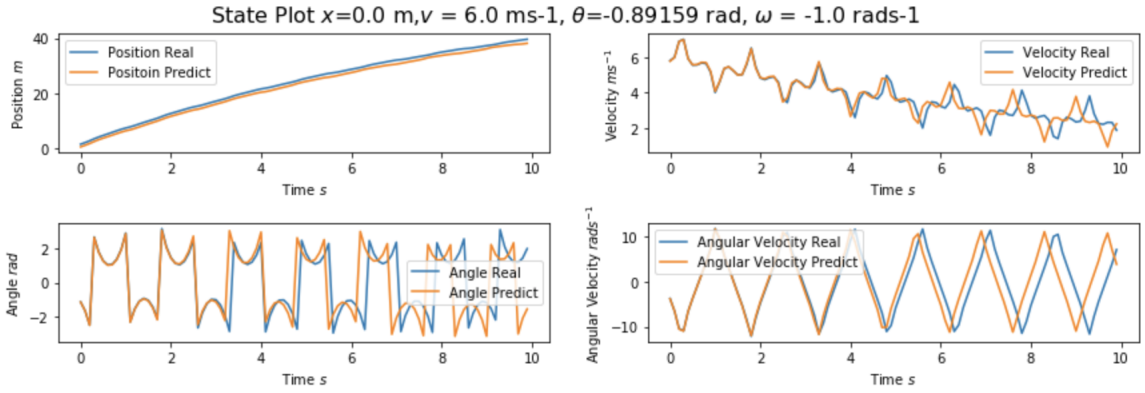
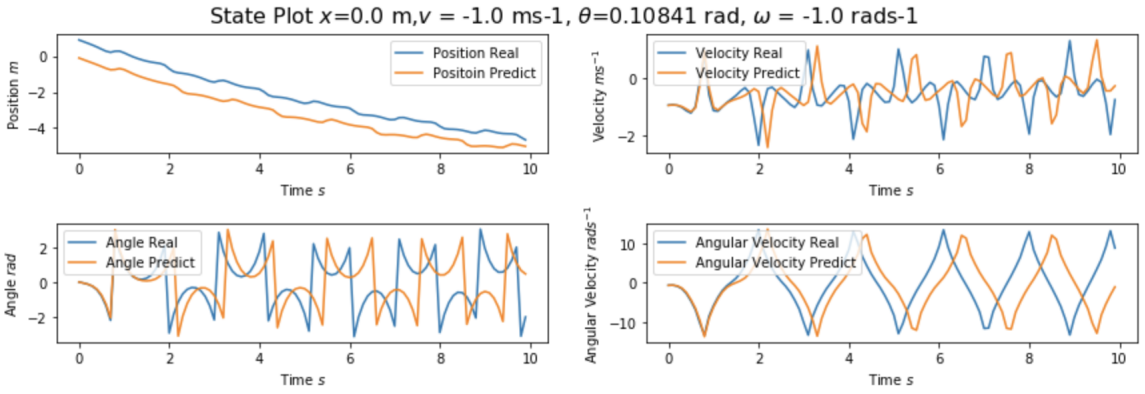


Figure 28: Detail Comparison between 1D-scan of the Real Model against the Non-Linear + Linear Model



E fig1.png

Figure 29: Prediction of the dynamics using non-linear regression from initial state $X(state) = [0, 6, -0.892, -1]$



E fig 2.png

Figure 30: Prediction of the dynamics using non-linear regression from initial state $X(state) = [0, -1, 0.108, -1]$

5.6 Appendix F: Non-Linear Model with Action Force

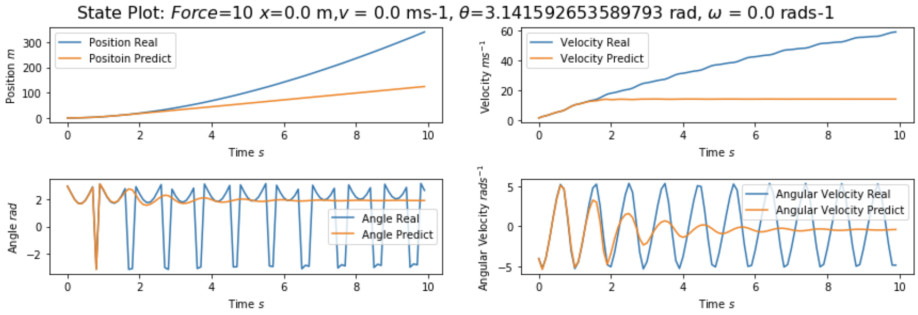
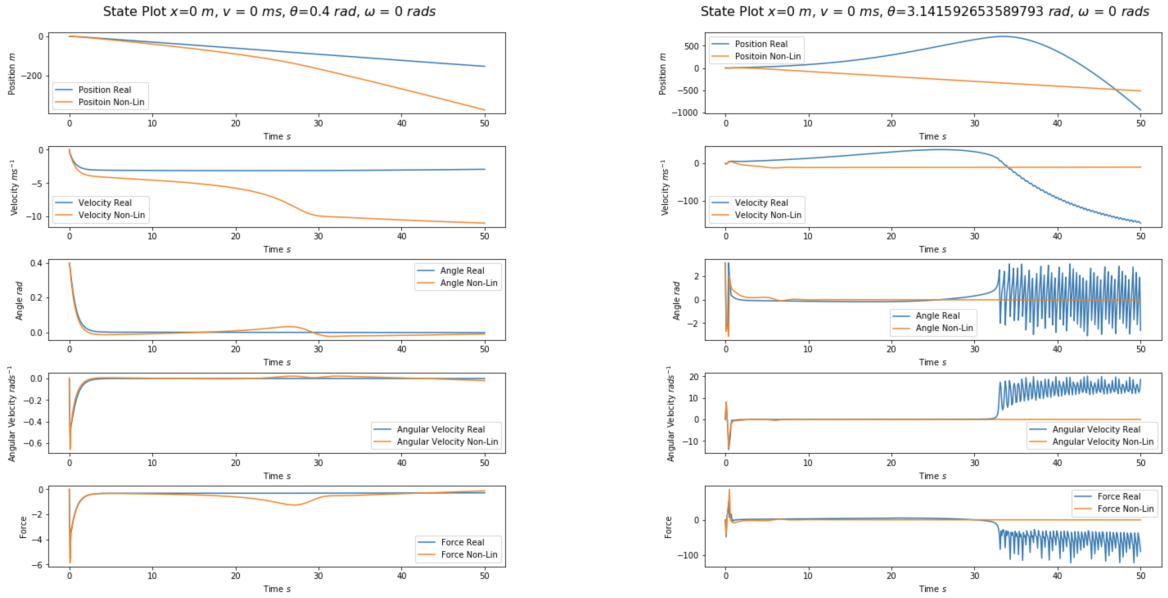


Figure 31: Roll-Out test on the predicted dynamics when a force of $F = 10$ is applied

5.7 Appendix H: Control Real Dynamics and Non-Linear Model Dynamics



(a) $t = 50$ and initial state $[0, 0, 0.4, 0]$ (b) $t = 50s$ and initial state $[0, 0, \pi, 0]$

Figure 32: Control Real Dynamics vs Non-Linear Dynamics

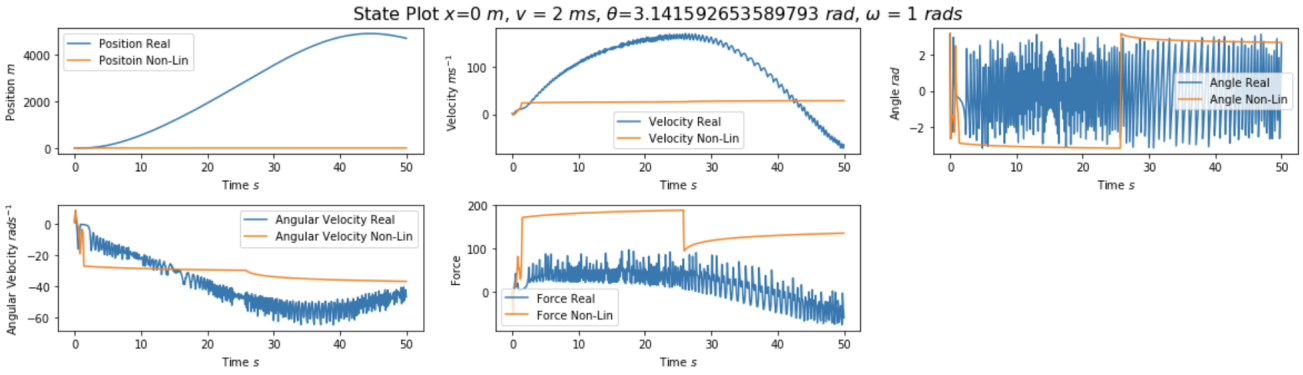


Figure 33: Unstable control of both Real System and Non-Linear System