ECE 6554 FINAL PROJECT: INVERTED CART-PENDULUM

By

Yinzhe Zhang and Zihuan Teng

School of Electrical and Computer Engineering

Georgia Institute of Technology

Apr 2022

TABLE OF CONTENTS

List of Tables iv

List of Figures v

Chapter 1: Introduction and Background 1

Chapter 2: Basic dynamic model for system 2

2.1 Parameters 2

2.2 The dynamic model 3

Chapter 3: linear controller design 5

3.1 System Linearization 5

3.2 Linear Controller 5

3.3 DMRAC 6

Chapter 4: Matched and Unmatched Uncertainty 8

Chapter 5: RBF neuron controller design 10

Chapter 6: Experiment Results 11

6.1 Step1's test: Linear controller and DMRAC 11

6.2 Step3's test: matched adaptive controller 15

6.3 Step2's test: RBF controller 19

Chapter 7: Conclusion 23

LIST OF TABLES

2.1 Params 2

2.2 Params2 3

LIST OF FIGURES

2.1 The cart-pendulum relationship 2

6.1 L-L NP controller 11

6.2 L-L P controller 11

6.3 L-L Noise controller 12

6.4 L-N NP controller 12

6.5 L-N P controller 13

6.6 L-N Noise controller 13

6.7 A-L NP controller 13

6.8 A-L P controller 14

6.9 A-L Noise controller 14

6.10 A-N NP controller 14

6.11 A-N P controller 15

6.12 A-N Noise controller 15

6.13 noise-free and non-perturb control 16

6.14 noise-free and non-perturb control-gain 16

6.15 noise-free and non-perturb control-error 16

6.16 noise and non-perturb control 17

6.17 noise and non-perturb control-gain 17
6.18 noise and non-perturb control-error 17
6.19 noise-free and perturb control 18
6.20 noise-free and perturb control-gain 18
6.21 noise-free and perturb control-error 18
6.22 noise-free RBF control 19
6.23 noise-free RBF control-gain 19
6.24 noise-free RBF control-error 20
6.25 noised RBF control 20
6.26 noised RBF control-gain 20
6.27 noised RBF control-error 21
6.28 trained RBF control 21
6.29 trained RBF control-gain 21
6.30 trained RBF control-error 22

Abstract

This paper is concerned with designing linear controller, DMRAC, mateched adaptive controller, RBF controller to solve the control problem of cart and pendulum. The models are given in the paper and the methods and update formulas used are presented. The experimental results show that we successfully implemented these Controllers.

CHAPTER 1

INTRODUCTION AND BACKGROUND

This project focuses on how to design a controller for the inverted cart-pendulum —— a classic example of an underactuated, naturally unstable, nonlinear system.The goal of the control is to make the angular difference θ between the pendulum and the origin 0.

The project is divided into three steps. In step 1,the aim is how to design a linear controller/DMRAC for a linearized system. And discussions is needed for the performance of both controllers in the presence of nonlinearity or noise. Step 2 focuses on how to design a RBF neural adaptive controller to achieve a predefined control objective. And the final step requires splitting the uncertainty into two parts: matched and unmatched. Then we need to cancel matched uncertainty and try to minimize the impact of unmatched uncertainty.

The main contributions and the structure of the remainder of the paper are shown below:

- Chapter 2 describes mainly the underlying dynamics of the system, the assumptions, and the definition of important parameters.
- Chapter 3 presents how to linearize the original system, and the design method of linear controller and DMRAC.
- Chapter 4 talked about how to use adaptive controller to cancel the unmatched nonlinear term.
- Chapter 5 discusses the principles of the implementation of RBF neural adaptive control.
- Chapter 6 contains our experiments results
- Chapter 7 are conclusions.

CHAPTER 2

BASIC DYNAMIC MODEL FOR SYSTEM

For our project, we consider the equations of motion that are derived from the energy of the cart-pendulum pairing. The cart-pendulum relationship is shown in Figure 2.1.

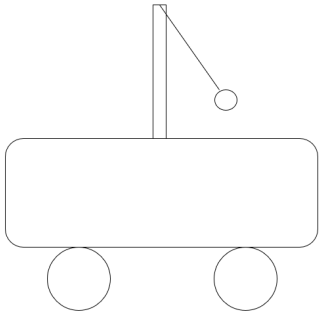

Figure 2.1: The cart-pendulum relationship

2.1 Parameters

The main parameters required for the derivation below are given in Table 2.1 and Table 2.2.

Table 2.1: Basic Parameters

Parameter	Meaning	Value
m_c	mass of cart	0.5kg
m_p	mass of pendulum	0.1kg
J_p	moment of inertia	0.006kg m^2
l	length of pendulum	0.3m
δ_θ	coefficient	0.05 N m / rad s
δ_x	coefficient	0.1 N m / s
g	gravity coefficient	9.8 m/ s^2

Table 2.2: Auxiliary Parameters

Parameter	Formula
M	$\frac{m_c+m_p}{m_p l}$
L	$\frac{J_p}{m_p l} + l$
b	$\frac{1}{m_p l}$
D_θ	$\frac{\delta_\theta}{m_p l}$
D_x	$\frac{\delta_x}{m_p l}$

2.2 The dynamic model

After those parameters determined, we can define state vector and then do the math calculations to find the control matrices:

The state vector will be a 4-elements row vector S:

$$S = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^{\mathbf{T}} \tag{2.1}$$

Note for $\ddot{x}$ and $\ddot{\theta}$, if we define γ_0 as $\frac{1}{ML-\cos^2\theta}$, we will have

$$\begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \gamma_0 \begin{bmatrix} L\sin(\theta)\dot{\theta}^2 - g\cos(\theta)\sin(\theta) - LD_x\dot{x} + D_\theta\cos(\theta)\dot{\theta} + Lbu \\ -\cos(\theta)\sin(\theta)\dot{\theta}^2 + Mg\sin(\theta) + D_x\cos(\theta)\dot{x} - MD_\theta\dot{\theta} - b\cos(\theta)u \end{bmatrix} \tag{2.2}$$

Further, plug-in the state vector here:

$$\dot{S} = AS + Bu + CS^2 + D \tag{2.3}$$

where:

$$A = \gamma_0 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -L \cdot D_x & D_\theta \cdot cos(\theta) \\ 0 & 0 & D_x \cdot cos(\theta) & -M \cdot D_\theta \end{bmatrix} \tag{2.4}$$

$$B = \gamma_0 \begin{bmatrix} 0 \\ 0 \\ L \cdot b \\ -b \cdot cos(\theta) \end{bmatrix} \tag{2.5}$$

$$C = \gamma_0 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & L \cdot sin(\theta) \\ 0 & 0 & 0 & -cos(\theta) \cdot sin(\theta) \end{bmatrix} \tag{2.6}$$

$$D = \gamma_0 \begin{bmatrix} 0 \\ 0 \\ -g \cdot cos(\theta) \cdot sin(\theta) \\ M \cdot g \cdot sin(\theta) \end{bmatrix} \tag{2.7}$$

This is the model we want to control.

CHAPTER 3

LINEAR CONTROLLER DESIGN

3.1 System Linearization

To linearize the equations of motion of the system, we need to provide the linearized equations. In this project, we will begin with deriving a tracking controller for the system so that it can track a cart trajectory while maintaining the pendulum position upright.

3.2 Linear Controller

To design a linear controller that stabilizes the system from non-zero angular deflection values, we need to linearize the system first from (2.3). Here we assume this linear system reaches the stability. Hence, when value of θ stabilize at 0, sin(θ) = 0 and cos(θ) = 1. With γ_0 updates to $\gamma_0' = \frac{1}{ML-1}$, we have

$$\begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \gamma_0' \begin{bmatrix} L\theta\dot{\theta}^2 - g\theta - LD_x\dot{x} + D_\theta\dot{\theta} + Lbu \\ -\theta\dot{\theta}^2 + Mg\theta + D_x\dot{x} - MD_\theta\dot{\theta} - bu \end{bmatrix} \tag{3.1}$$

where:

$$A' = \gamma_0' \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -L \cdot D_x & D_\theta \\ 0 & 0 & D_x & -M \cdot D_\theta \end{bmatrix} \tag{3.2}$$

$$B' = \gamma_0' \begin{bmatrix} 0 \\ 0 \\ L \cdot b \\ -b \end{bmatrix} \tag{3.3}$$

$$C' = \gamma_0' \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & L \cdot \theta \\ 0 & 0 & 0 & -\theta \end{bmatrix} \tag{3.4}$$

$$D' = \gamma_0' \begin{bmatrix} 0 \\ 0 \\ -g \cdot \theta \\ M \cdot g \cdot \theta \end{bmatrix} \tag{3.5}$$

Next, we design the control output as below:

$$u = -kS \tag{3.6}$$

where k is a state-feedback matrix such that its eigenvalues do not have a multiplicity greater than the number of inputs. Here, for linear dynamics, the system dynamics S is

$$\dot{S} = A'S + B'u \tag{3.7}$$

for nonlinear dynamics, the system dynamics is just what we got in Chap2.

3.3 DMRAC

To implement the adaptive system dynamics, we introduce the system in perturbed parameters case and then quantify the control effort with reference model (Non Perturbable System) twice. Once with the incorrect values, then once with the final adaptive coefficients from the first run as that initial adaptive coefficients for the second run.

Next we design the control output as below use direct feedback form

$$u = k_x S + k_r r \tag{3.8}$$

with adaptive laws

$$\dot{k_x} = -\gamma_x \cdot S \cdot e \cdot sgn(B') \tag{3.9}$$

$$\dot{k_r} = -\gamma_r \cdot r \cdot e \cdot sgn(B') \tag{3.10}$$

and system dynamics S above applied.e represent error here.

CHAPTER 4

MATCHED AND UNMATCHED UNCERTAINTY

Fortunately, the main structure of this part has changed very little compared to Chap3. Notice that here we need to implement two functions at the same time with one adaptive controller, so we need to follow the design approach from chapter 14 of UIUC notes.

We start with the derivation of the f_m term. Now, just compare two equations:

$$\dot{S} = AS + Bu + CS^2 + D \tag{4.1}$$

and

$$\dot{S} = As + Bu + Bf_m + f_u \tag{4.2}$$

It is clear that f_m and f_u correspond exactly to the last two items.And according to the principle of coefficient correspondence matching, it is obvious to find that f_m corresponds to the item CS^2.This means we can represent it as a scalar:

$$f_m = W * \Phi(S) \tag{4.3}$$

Where

$$\Phi(S) = sin(\theta) * (\dot{\theta})^2 \tag{4.4}$$

where W is an unknown weight needs to be learned.Choose λ equal to 1 will allow us to update the parameters with the following 4 formulas.

$$
\begin{aligned}
\dot{k}_x(t) &= \Gamma_x \operatorname{Proj}\left(k_x(t), -S(t)e^\top(t)Pb\right) \\
\dot{k}_r(t) &= \gamma_r \operatorname{Proj}\left(k_r(t), -r(t)e^\top(t)Pb\right) \\
\dot{\hat{W}}(t) &= \Gamma_W \operatorname{Proj}\left(\hat{W}(t), \Phi(S(t))e^\top(t)Pb\right) \\
\dot{\psi(t)} &= \gamma_\psi \operatorname{Proj}(\psi(t), e^\top(t)Pb \tanh\left(\frac{e^\top(t)Pb}{\delta}\right))
\end{aligned}
\tag{4.5}
$$

As for controller,

$$
u(t) = k_x^\top(t)x(t) + k_r(t)r(t) - \hat{W}^\top(t)\Phi(S(t)) - \psi(t)\tanh\left(\frac{e^\top(t)Pb}{\delta}\right) \tag{4.6}
$$

Here $Proj()$ is the projection function.The third term of u is guaranteed to cancel f_m, while the fourth term is used to give the bound of f_u.

CHAPTER 5

RBF NEURON CONTROLLER DESIGN

The RBF neuron controller's design is simpler. Here, note the reference model is the linear system, thus we know we need to use RBF controller to cancel $CS^2 + D$ term.

Before we start, a point to be declared is that we assume that the network does not need to learn terms related to x, which means that $\dot{x}$ is not to be considered. Based on this, the basic implementation step is to design a set of RBF neural networks to fit the Nonlinear term. initially we should define a set of Centers Cen, which is determined using the pre-tested range R. Using the Meshgrid function they can be combined into a 4-D shape for the next step, with each node of this cube is a center.

We choose to use 20 neurons here, so the shape will have 3*20*20*20 dimensions. Then:

- For each frame, get the difference of the input state S and centers. This gives us the differencial matrix $Diff$
- From $Diff$ we could calculate norm, then apply gaussian kernel and covariance to get the RBF output matrix.
- After get this matrix, we sum over it in 3 directions, this will generate 3 20*1 vector for each state variable, known as Φ_i
- Use projection function update coefficients α_i, then sum $\alpha_i \Phi_i$ we find the f_{NN} term to cancel the nonlinearity.

CHAPTER 6

EXPERIMENT RESULTS

For analysis and conclusions, please refer to Chap7.Chap6 only has test results.

6.1 Step1's test: Linear controller and DMRAC

The result of linear non perturbable linear controller (L-L NP) that implements linearized dynamics looks as below:

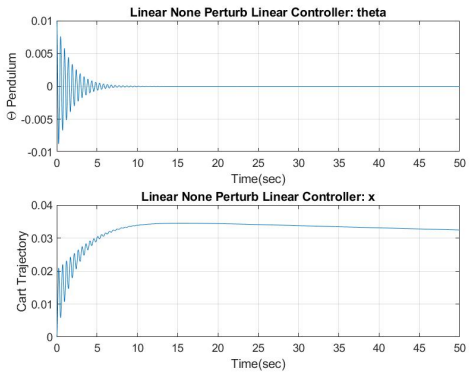

Figure 6.1: L-L NP controller

Result of linear perturbable linear controller (L-L P) looks as below:

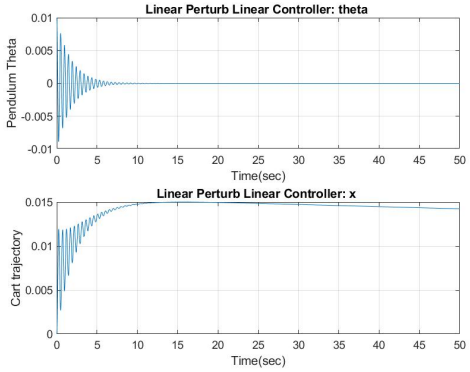

Figure 6.2: L-L P controller

Result of linear noise linear controller (L-L Noise) looks as below:

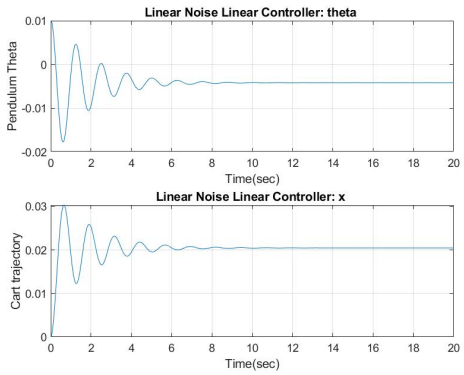

Figure 6.3: L-L Noise controller

The result of linear non perturbable nonlinear controller (L-N NP) that implements linearized dynamics looks as below:

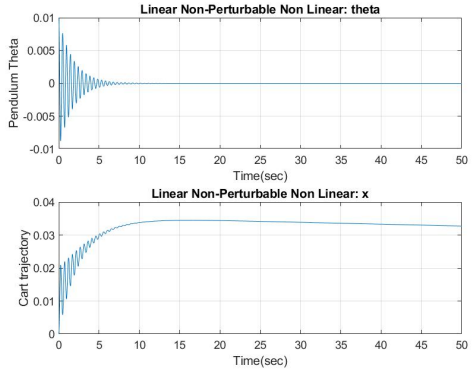

Figure 6.4: L-N NP controller

Result of linear perturbable nonlinear controller (L-N P) looks as below:

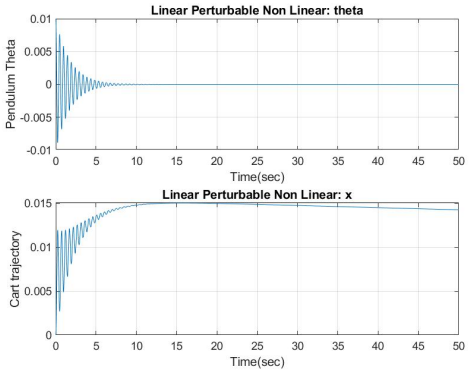

Figure 6.5: L-N P controller

Result of linear noise nonlinear controller (L-N Noise) looks as below:

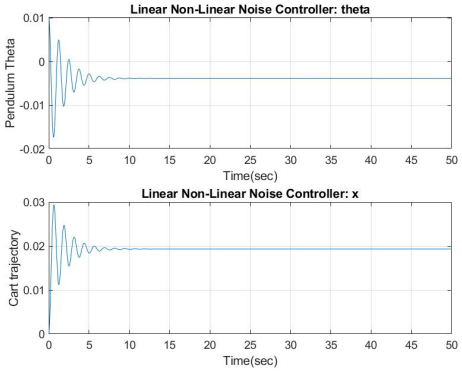

Figure 6.6: L-N Noise controller

Result of adaptive non perturbable linear controller (A-L NP) looks as below:

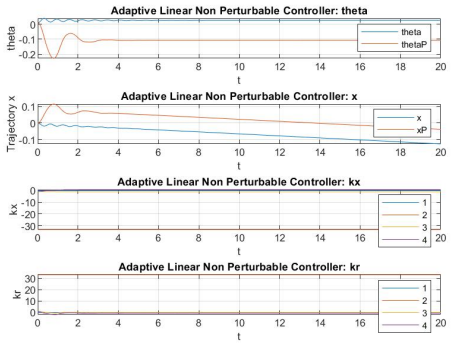

Figure 6.7: A-L NP controller

Result of adaptive perturbable linear controller (A-L P) looks as below:

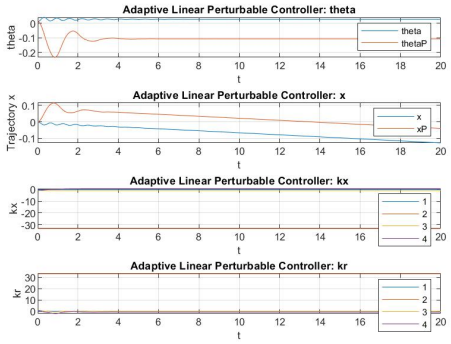

Figure 6.8: A-L P controller

Result of adaptive noise linear controller (A-L Noise) looks as below:

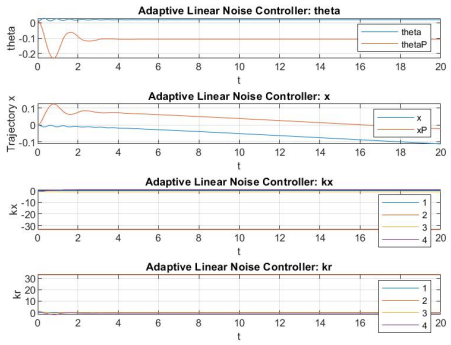

Figure 6.9: A-L Noise controller

Result of adaptive non perturbable nonlinear controller (A-N NP) looks as below:

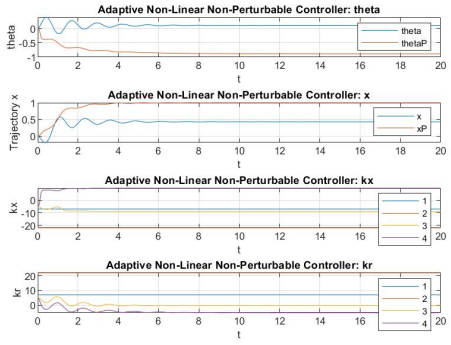

Figure 6.10: A-N NP controller

Result of adaptive perturbable nonlinear controller (A-N P) looks as below:

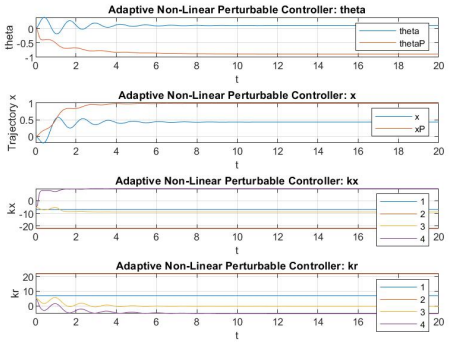

Figure 6.11: A-N P controller

Result of adaptive noise nonlinear controller (A-N Noise) looks as below:

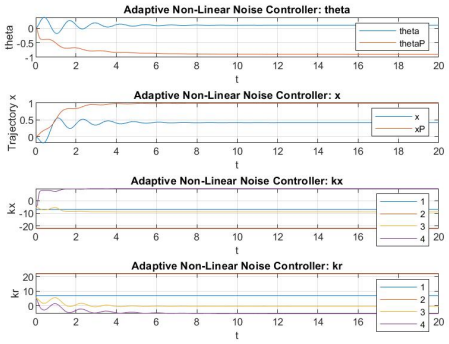

Figure 6.12: A-N Noise controller

6.2 Step3's test: matched adaptive controller

I conducted several sets of experiments to determine the initial range of values that the controller could withstand. The results show that when initial value of θ is bigger than 0.4 rad, the system will fail.

Therefore we choose the case with θ has an initial value of 0.1 rad as example, and it contains three condition:

Result of noise-free and non-perturb control:

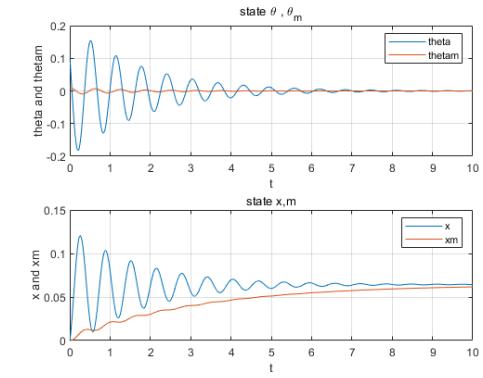

Figure 6.13: noise-free and non-perturb control

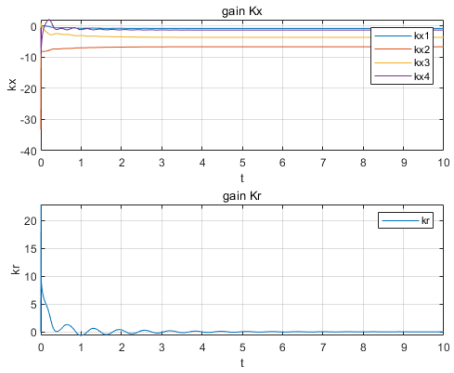

Figure 6.14: noise-free and non-perturb control-gain

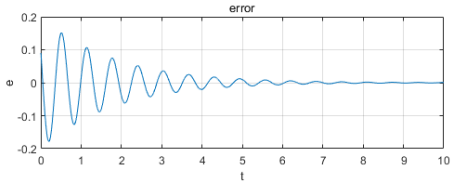

Figure 6.15: noise-free and non-perturb control-error

Result of noise and non-perturb control, with variance of gaussian noise is 0.25:

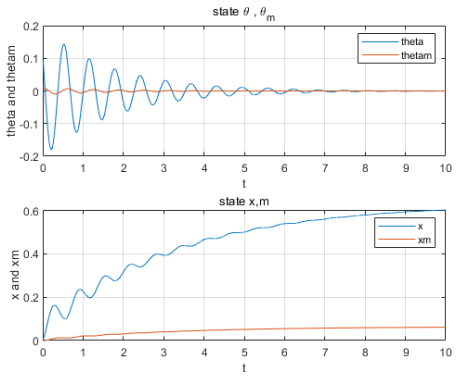

Figure 6.16: noise and non-perturb control

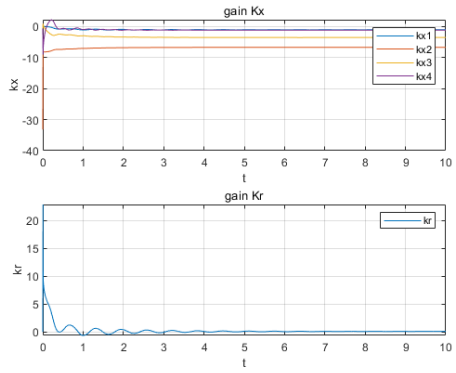

Figure 6.17: noise and non-perturb control-gain

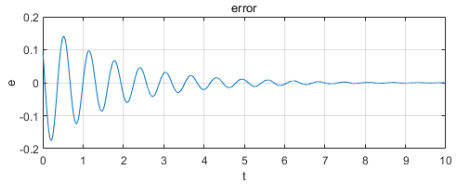

Figure 6.18: noise and non-perturb control-error

Result of noise-free and perturb control, with m_c increased 10%, δ_x decreased 10%:

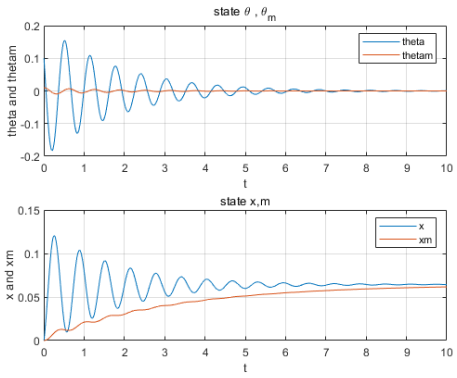

Figure 6.19: noise-free and perturb control

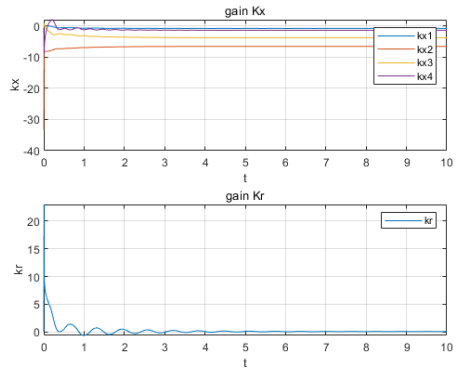

Figure 6.20: noise-free and perturb control-gain

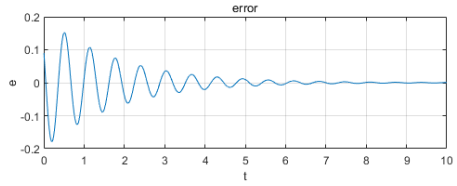

Figure 6.21: noise-free and perturb control-error

6.3 Step2's test: RBF controller

Again, we first tested the maximum value that the RBF controller can tolerate. The max initial value is 0.6 rad. Again, we chose 0.1 rad as an example of the initial state to prove the correctness of our implementation. This time, however, we only tested the case with or without noise and the case where the trained weights were used for inference.

Result of noise-free RBF control:

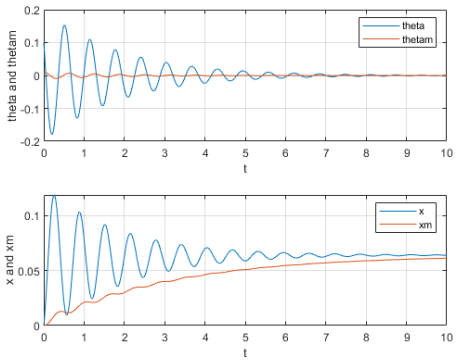

Figure 6.22: noise-free RBF control

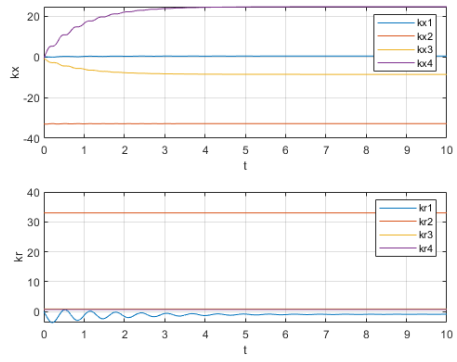

Figure 6.23: noise-free RBF control-gain

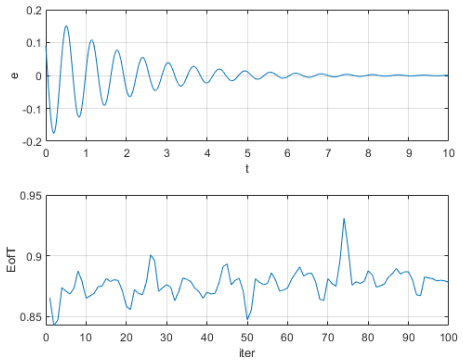

Figure 6.24: noise-free RBF control-error

Result of noised RBF control, with variance of noise is 0.25:

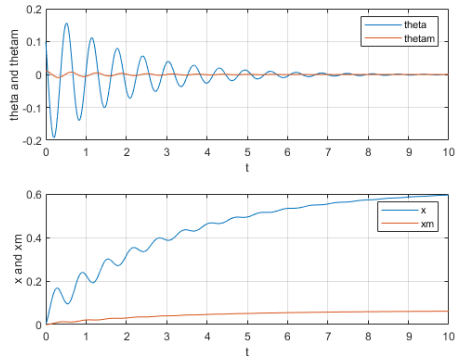

Figure 6.25: noised RBF control

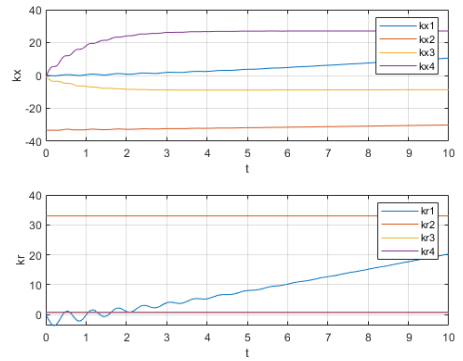

Figure 6.26: noised RBF control-gain

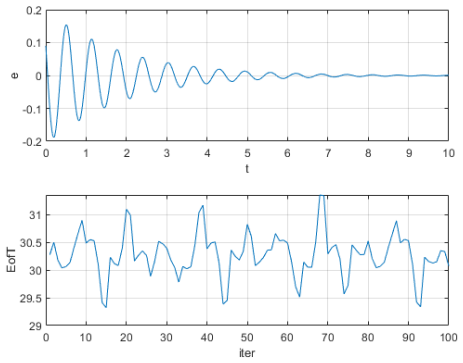

Figure 6.27: noised RBF control-error

Result of noise-free RBF control, use 0.1 rads' final weight for 0.6 rads initialization:

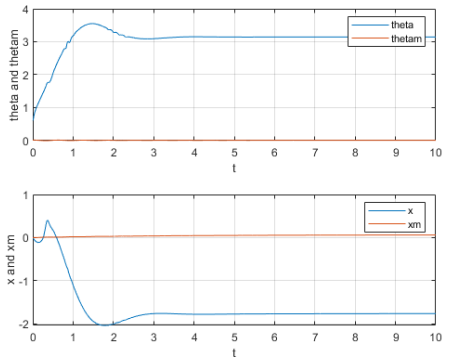

Figure 6.28: trained RBF control

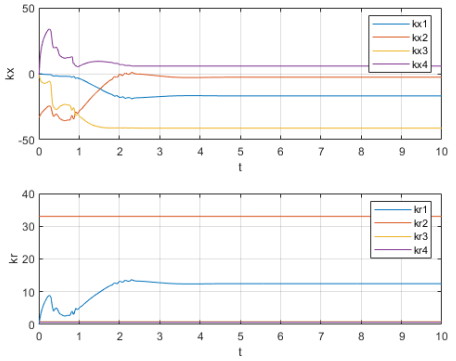

Figure 6.29: trained RBF control-gain

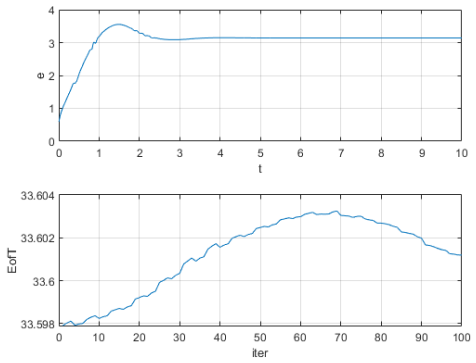

Figure 6.30: trained RBF control-error

CHAPTER 7
CONCLUSION

From the results in Chap6, we can draw the following conclusions:

- With our derived linearized equations to design the linear controller that implemented on linear dynamics, we can bring the system back to steady state. By implementing this linear dynamics on three different cases with purturbance (by varying l and J_p variable), without perturbance, and with noise, all three cases bring back to steady states in the end.

- The proposed nonlinear dynamics design can also bring the system back to steady state in three different cases shown above. And we saw that for the case with perturbances applied, the system takes longer time to reach steady state.

- Using the adaptive controller implemented on linear dynamics, the system will converge at a faster rate than linear controller. After 20s of system running, the x subsytem is off by around 0.1m. For the nonlinear dynamics adaptive controller, it tracks the system better than the linear adaptive controller.

- Using a matched noise-cancelling adaptive controller, for small angular perturbations, we can bring it back to the steady state after enough time, while the trajectory does not deviate much.This is even more than we expected, because in theory we can only control the unmatched term through the gain term instead of eliminating it.

- As you can see, adding noise does have an effect on the matched controller. Although the angle value can eventually be guaranteed to still converge to zero, its trajectory is clearly not controlled and returns to the origin.However, the effect of the perturbation

is minimal, and the results of our tests do not differ much from those without adding the perturbation, and the system converges in both angle and displacement.

- The RBF controller has better capabilities. It has a higher tolerance of initial values than the matched adaptive controller, in addition to perfectly accomplishing the task of noise cancellation. (An initial value of at least 0.5 rad does not make it fail, but 0.4 is enough to make the trajectory of the matched adaptiver controller diverge).

- Noise also has a significant impact on the RBF controller. With the same noise effect, the RBF shows similar displacement drift as the matched controller, while the angle values can still be controlled well.

- Unfortunately, the generalization ability of the RBF controller obtained after training does not seem to be strong. It is unable to stabilize the angle and displacement to the origin when controlling for the initial value of 0.6 rad, but arrives at two fixed constants. We think this may be due to the fact that we still did not train it enough (we only gave it 10s of simulations for training) and to the RBF center parameters.

- Taken together, even the simplest linear controller can achieve good results in this problem by setting the poles of the system. However, adaptive control can provide faster convergence, which is one of its major advantages. In addition, adaptive controllers like RBF can still guarantee convergence when extreme initial conditions/noise and perturbations are encountered. In other words, they are more robust.