An Analysis of Non-Linear Adaptive Output Feedback Control Algorithms for a Robotic Manipulator

by

Aarti Gupta
B.Eng

A thesis submitted to
The Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario

August 2004

© Copyright
2004, Aarti Gupta

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-97411-1

Our file Notre référence

ISBN: 0-612-97411-1

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

The undersigned recommend to
The Faculty of Graduate Studies and Research
Acceptance of the thesis

An Analysis of Non-Linear Adaptive Output Feedback Control Algorithms for a Robotic Manipulator

submitted by

Aarti Gupta, B.Eng

in partial fulfillment for
the degree of Master of Applied Science

Professor H.M.Schwartz
Thesis Supervisor

Professor R.A. Goubran
Chair, Department of Systems and Computer Engineering

Carleton University

August 2004

Abstract

In this thesis seven different algorithms for output feedback nonlinear adaptive control of robotic manipulators are analyzed. Six of these algorithms are based on output feedback nonlinear model reference adaptive control. The other algorithm is based on output feedback using a gain matrix. The primary goal of the thesis is to analyze the stability and convergence region of the algorithms. The conditions under which the algorithms are stable and the type of stability they exhibit are discussed. The secondary goal of the thesis is to analyze the effect of the components of the algorithm on the stability of the algorithm. The components studied include a filter in the adaptation law and the normalizing factor. The analysis of the algorithms is done via the simulation of a two degree of freedom robotic manipulator. The results of the output feedback nonlinear model reference adaptive control algorithms show that the without observer algorithm which uses model values and a normalizing factor has the greatest region of convergence. The analysis of the output feedback using a gain matrix algorithm shows that it has a region of convergence for lower input amplitudes.

Acknowledgments

I would like to thank my supervisor, Professor Schwartz. Thank you for your time, patience and all your help. Thanks for putting up with all my silly questions and explaining the same concepts over and over again. Without your patience and support, this thesis would have been impossible.

I would also like to thank my family for their support and encouragement. Without your support and love I would not have had the courage to pursue my Master's degree. Your faith gave me the confidence that I needed to complete this thesis and my degree. Thanks dad, mom and Sanjiv!

Table of Contents

1.0	Introduction.....	1
1.1	Conceptual Overview	3
1.2	Theoretical Details: 2-degree of freedom robotic manipulator	8
2.0	Output Feedback Nonlinear Adaptive Control using only gains in the feedback loop	10
2.1	Algorithmic Details	10
2.2	Results	17
3.0	Model Reference Nonlinear Output Feedback Adaptive Control (MRAC) - With Observer.....	25
3.1	2 nd Order System	25
3.1.1	Algorithmic Details	25
3.1.2	Results	34
3.2	3 rd Order System.....	49
3.2.1	Algorithmic Details	49
3.2.2	Results	58
4.0	Model Reference Nonlinear Output Feedback Adaptive Control (MRAC) - Without Observer.....	65
4.1	Algorithmic Details	65
4.2	Results	67
5.0	Conclusion	97
	References.....	101

List of Figures

Figure 1: Controller and plant.....	4
Figure 2: Conceptual level adaptive control.....	5
Figure 3: Controller.....	6
Figure 4: Detailed conceptual level adaptive control.....	7
Figure 5: 2-degree of freedom robotic manipulator.....	8
Figure 6: Output feedback nonlinear adaptive control for a two degree of freedom robotic system.....	11
Figure 7: Robotic manipulator system assuming perfect linearization.....	13
Figure 8: OFKx: Trajectory error for amplitude 1, initial parameter estimates: 1,3.....	18
Figure 9: OFKx: Parameter error for amplitude 1, initial parameter estimates: 1,3.....	19
Figure 10: OFKx: Trajectory error for amplitude 1, initial parameter estimates: 10,8.....	20
Figure 11: OFKx: Parameter error for amplitude 1, initial parameter estimates: 10,8.....	21
Figure 12: OFKx: Trajectory error for amplitude 2, initial parameter estimates: 1,2.....	22
Figure 13: OFKx: Parameter error for amplitude 2, initial parameter estimates: 1,2.....	23
Figure 14: MRAC 2-degree of freedom robotic manipulator system.....	26
Figure 15: MRAC perfect linearization system.....	32
Figure 16: MRAC (F): Trajectory Error for Amplitude 1, Initial Parameter Estimates: 1,3.....	35
Figure 17: MRAC (F): Parameter error for amplitude 1, initial parameter estimates: 1,3.....	36
Figure 18: MRAC (F): Trajectory error for amplitude 1, initial parameter estimates: 10,8.....	37
Figure 19: MRAC (F): Parameter error for amplitude 1, initial parameter estimates: 10,8.....	38
Figure 20: MRAC (F): Trajectory error for amplitude 2, initial parameter estimates: 1,3.....	39
Figure 21: MRAC (F): Parameter error for amplitude 2, initial parameter estimates: 1,3.....	40
Figure 22: MRAC (WF): Trajectory error for amplitude 1, initial parameter estimates: 1,3.....	42
Figure 23: MRAC (WF): Parameter error for amplitude 1, initial parameter estimates: 1,3.....	43
Figure 24: MRAC (WF): Trajectory error for amplitude 1, initial parameter estimates: 10,8.....	44
Figure 25: MRAC (WF): Parameter error for amplitude 1, initial parameter estimates: 10,8.....	45
Figure 26: MRAC (WF): Trajectory error for amplitude 2, initial parameter estimates: 1,3.....	46
Figure 27: MRAC (WF): Parameter error for amplitude 2, initial parameter estimates: 1,3.....	47
Figure 28: MRAC 3 rd order system block diagram.....	55
Figure 29: MRAC (3 rd F): Parameter error for amp: 0.3, initial parameter estimates: -1.0,-0.6,0.25.....	59
Figure 30: MRAC (3 rd F): Output error for amp: 0.3, initial parameter estimates: -1.0,-0.6,0.25.....	60
Figure 31: MRAC (3 rd F): Parameter error for amp: 0.3, initial parameter estimates: -1.2,-0.4,0.1.....	61
Figure 32: MRAC (3 rd F): Output error for amp: 0.3, initial parameter estimates: -1.2,-0.4,0.1.....	62
Figure 33: MRAC (NWO): Trajectory error for amplitude 1, initial parameter estimates: 1,3.....	68
Figure 34: MRAC (NWO): Parameter error for amplitude 1, initial parameter estimates: 1,3.....	69
Figure 35: MRAC (NWO): Trajectory error for amplitude 1, initial parameter estimates: 10,8.....	70
Figure 36: MRAC (NWO): Parameter error for amplitude 1, initial parameter estimates: 10,8.....	71
Figure 37: MRAC (NWO): Trajectory error for amplitude 2, initial parameter estimates: 1,3.....	72
Figure 38: MRAC (NWO): Parameter error for amplitude 2, initial parameter estimates: 1,3.....	73
Figure 39: MRAC (NNWO): Trajectory error for amplitude 1, initial parameter estimates: 1,3.....	76
Figure 40: MRAC (NNWO): Parameter error for amplitude 1, initial parameter estimates: 1,3.....	77
Figure 41: MRAC (NNWO): Trajectory error for amplitude 1, initial parameter estimates: 10,8.....	78
Figure 42: MRAC (NNWO): Parameter error for amplitude 1, initial parameter estimates: 10,8.....	79
Figure 43: MRAC (NNWO): Trajectory error for amplitude 2, initial parameter estimates: 1,3.....	80
Figure 44: MRAC (NNWO): Parameter error for amplitude 2, initial parameter estimates: 1,3.....	81
Figure 45: MRAC (MVNNWO):Trajectory error for amplitude 1, initial parameter estimates: 1,3.....	83
Figure 46: MRAC (MVNNWO): Parameter error for amplitude 1, initial parameter estimates: 1,3.....	84
Figure 47: MRAC(MVNNWO):Trajectory error for amplitude 1, initial parameter estimates: 10,8.....	85
Figure 48: MRAC(MVNNWO): Parameter error for amplitude 1, initial parameter estimates: 10,8.....	86
Figure 49: MRAC (MVNNWO):Trajectory error for amplitude 2, initial parameter estimates: 1,3.....	87
Figure 50: MRAC (MVNNWO): Parameter error for amplitude 2, initial parameter estimates: 1,3.....	88

Figure 51: MRAC (MVNWO): Trajectory error for amplitude 1, initial parameter estimates: 1,3	90
Figure 52: MRAC (MVNWO): Parameter error for amplitude 1, initial parameter estimates: 1,3.....	91
Figure 53: MRAC (MVNWO): Trajectory error for amplitude 1, initial parameter estimates: 10,8	92
Figure 54: MRAC (MVNWO): Parameter error for amplitude 1, initial parameter estimates: 10,8.....	93
Figure 55: MRAC (MVNWO): Trajectory error for amplitude 2, initial parameter estimates: 1,3	94
Figure 56: MRAC (MVNWO): Parameter error for amplitude 2, initial parameter estimates: 1,3.....	95

List of Tables

Table 1: OFKx: Other results.....	24
Table 2: MRAC (F): Other results	41
Table 3: MRAC (WF): Other results	48
Table 4: MRAC (3 rd F): Other results.....	63
Table 5: MRAC (3 rd WF): Other results	64
Table 6: MRAC (NWO): Other results.....	74
Table 7: MRAC (NNWO): Other results.....	82
Table 8: MRAC (MVNNWO): Other results	89
Table 9: MRAC (MVNWO): Other results	96
Table 10: Algorithm Results.....	98

List of Acronyms

OFKx	Output feedback nonlinear adaptive control using only gain in the feedback loop.
MRAC(F)	Model reference adaptive control algorithm with filter in the adaptation law.
MRAC(WF)	Model reference adaptive control algorithm without filter in the adaptation law.
MRAC(3rd F)	Model reference adaptive control algorithm with filter in the adaptation law for a 3 rd order system.
MRAC(NWO)	Model reference adaptive control algorithm with the normalizing factor and without the observer.
MRAC(NNWO)	Model reference adaptive control algorithm without the normalizing factor and without the observer.
MRAC(MVNNWO)	Model reference adaptive control algorithm with model values but without the normalizing factor and without the observer.
MRAC(MVNWO)	Model reference adaptive control algorithm with model values and the normalizing factor but without the observer.

Chapter 1

1.0 Introduction

In 1987, Craig, Hsu and Sastry [1] proposed a stable parameter-adaptive control algorithm that could be used to control mechanical manipulators. This work led to the work by Sastry and Isodori [2] on adaptive control of nonlinear systems that are linearizable by full state feedback. Sastry and Isodori [2] used the specialized robotic manipulator case by Craig, Hsu and Sastry[1] and proposed a more general case for adaptive control. The difficulty with these methods was that they required full state feedback and the highest derivative of the output signal. This causes problems for the robotics case because the resulting control and adaptation laws require the measurement of the link position, link velocity and link acceleration. The velocity and acceleration signals are not always available. Even if they are available they are usually very difficult to measure and are very noisy. For this reason, these algorithms were not very practical for robotic manipulators. Also in 1987, Slotine and Li [3] proposed an algorithm that avoided the use of the acceleration signal. Using the property of the robot dynamics they were able to derive a control law that used only the link positions and link velocities. Since, they used the property of robotic dynamics this algorithm could not be generalized to other nonlinear adaptive control system cases as was done previously by Sastry and Isodori[2]. Khalil [5] and Khalil and Esfandari [4] made contributions by proposing algorithms that required observers. The observers were used to estimate the states of the system. The goal was to remove the dependency of the algorithm on the system states as these states were not always available for measurement. They were able to prove, using Lyapunov stability, that the algorithm was semiglobally stable. Several model reference

adaptive control algorithms were also proposed in the past decade. Schwartz [8] proposed an algorithm for model reference adaptive control that used a linear observer. This algorithm was shown to have a region of convergence via simulations though no theoretical analysis was presented. Schwartz [7] also proposed an algorithm that avoided the use of observers altogether. This algorithm uses the model velocity instead of the estimated value in the adaptation law. The advantage of this algorithm is that it is independent of the link velocity measurement and instead depends on the model value, which is easily available. Not only this, but the algorithm also has a simpler implementation as it does not require an observer. This algorithm was also shown to have a region of convergence via simulations. The theoretical analysis was stated as part of the future work.

As can be seen from the literature review, several algorithms have been proposed and published. We focus our analysis on the model reference adaptive control algorithms proposed by Schwartz [7][8] and the nonlinear output feedback algorithm proposed by Jankovic [6]. The algorithms proposed by Schwartz [7][8] have no proof of stability or convergence region analysis. Jankovic's algorithm [6] has a proof of stability but it is for a general model of adaptive control. We extend Jankovic's algorithm to the robotic manipulator case. We then analyze the region of convergence of these algorithms via simulations of a two-degree of freedom robotic manipulator. As these algorithms were simulated they were studied in more detail. The first algorithm by Schwartz [8] uses a filter in the adaptation law. We study the effect of this filter on the convergence of the algorithm. This algorithm is simulated for both a 2nd and a 3rd order system. The second algorithm by Schwartz [7] uses a normalizing factor. We also study the effect of this

normalizing factor on the convergence of the algorithm. Another element that we study for this particular algorithm is the use of a model regressor. A model regressor uses all model values. We use the model regressor in the adaptation law and study its effect on the performance of the algorithm. As the results will show, the algorithm with the model regressor and the normalizing factor has the largest region of convergence compared to the rest of the model reference adaptive control algorithms.

In the rest of this section we provide a conceptual overview of adaptive control and provide the theoretical details of a two-degree of freedom robotic manipulator. In Chapter 2, the output feedback nonlinear adaptive control (using only gains in the feedback loop) algorithm is analyzed and the simulation results are presented. In Chapter 3, the model reference nonlinear output feedback adaptive control algorithms with the observer are discussed and the results are analyzed. In Chapter 4, the model reference nonlinear output feedback adaptive control algorithms without the observer are discussed and the results are presented. Finally, in Chapter 5 some important conclusions are drawn.

1.1 Conceptual Overview

This section presents a conceptual overview of adaptive control and presents the basic knowledge required to understand the algorithms discussed.

Before discussing adaptive control, it is important to discuss why controllers are required. Controllers are required to change the response of a plant or a process that has already been established and cannot easily be changed. For instance, there might be a plant that is established and is working, though the output of the plant to a specific input

is not what is desired. In this case, instead of changing a well defined plant, one can design a controller and control the plant as desired. Figure 1 demonstrates this.

Figure 1: Controller and plant

In Figure 1, y_r is the reference input signal and y_p is the plant output. One can design the controller of Figure 1 to drive the output of the plant, y_p , to the desired output values. The parameters of the controller can be determined by looking at the parameters of the plant and the desired behaviour of the plant. But the problem arises because the parameters of the plant are unknown. There are several techniques like Recursive Least Squares that can be used to find the parameters of the plant but these techniques do not work for non-linear complex plants. To deal with the complexity adaptive control is required. Adaptive control is where the error between the desired and the actual plant output, y_p , is used as a feedback to the adaptation law to tune the controller parameter values. Since, the plant parameters are unknown, the exact values of the controller parameters cannot be calculated. Hence, the controller is started with some initial parameter estimates and then via adaptive control the parameters are tuned until the plant behaves according to the required specifications. Adaptive control is shown at a conceptual level in Figure 2.

Figure 2: Conceptual level adaptive control

In Figure 2, the reference model indicates the desired behaviour of the plant. We want to design the controller such that the plant behaves like the reference model. The input y_r is fed to both the reference model and the controller. The goal is to make the plant output, y_p , equal to the reference model output, y_m . This goal is accomplished by adapting the controller parameters. The adaptation law is used to adapt the controller parameters. It takes the current error between the plant output, y_p , and the model output, y_m , to determine how the parameters of the controller are to be changed. The parameter values are adjusted such that the error between y_p and y_m is reduced. Once this error reduces to zero, the controller reaches the ideal parameter values and the adaptation stops.

The next step is to determine what is “inside” the controller. The controller contains several pieces. This is shown in Figure 3.

Figure 3: Controller

A linearization block is required in the controller since the plant is nonlinear. As the name suggests, the goal of the linearization block is to make the nonlinear plant appear linear. Therefore, the linearization block along with the nonlinear system creates a linear system that can be used for adaptive control. The control law defines the feedback loop. The observer is required because we are dealing with output feedback. Output feedback indicates that only the output of the plant will be used in the feedback loop as opposed to the entire state. When the entire state is used, it is called full state feedback. There are several problems with full state feedback. First, the full state of the system is generally not available for measurement. Secondly, even if the states are available for measurement, the values measured are often very noisy. Hence, an observer is used to estimate the states. These estimated states may then be used in the linearization law and/or the adaptation law.

Finally, Figure 4 is obtained by putting the entire adaptive control picture together. To be clear that the plant is nonlinear, the plant block has been replaced by a nonlinear system block.

Figure 4: Detailed conceptual level adaptive control

Some of the applications of adaptive control include a missile whose center of gravity and mass changes as the fuel is consumed. Since, the system dynamics are constantly changing, the calculations that use these values must be continuously updated. Several other examples of changing dynamics include a ship where the dynamics change depending on the ship speed, the depth of the water, the waves, current, wind etc. In a paper mill, the moment of inertia of the winding reel changes as the paper is wound on,

so the motor torque required to maintain constant tension in the web must also change. In general, any system influenced by noise and disturbances that cannot be determined in advance must adapt to these changes on a continuous basis and hence, requires adaptive control. [9]

1.2 Theoretical Details: 2-degree of freedom robotic manipulator

The majority of the simulations involve the two degree of freedom robotic manipulator system. A two degree of freedom robotic manipulator is shown in Figure 5. The m_1, m_2 values represent the mass and f represents the link length. For our simulations, both links are considered to be equal in length. The q_1 and q_2 values are the link positions.

Figure 5: 2-degree of freedom robotic manipulator

Since, the manipulator is in the horizontal plane gravity can be ignored. The dynamics of the above system are defined by

$$T = M(q)\ddot{q} + C(q, \dot{q}) \quad (1.1)$$

where

$$M(q) = \begin{bmatrix} m_1 f^2 + 2m_2 f^2 + 2m_2 f^2 \cos(q_2) & m_2 f^2 + m_2 f^2 \cos(q_2) \\ m_2 f^2 + m_2 f^2 \cos(q_2) & m_2 f^2 \end{bmatrix} \quad (1.2)$$

$$C(q, \dot{q}) = \begin{bmatrix} -2m_2 f^2 \dot{q}_1 \dot{q}_2 \sin(q_2) - m_2 f^2 \dot{q}_2^2 \sin(q_2) \\ m_2 f^2 \dot{q}_1^2 \sin(q_2) \end{bmatrix}. \quad (1.3)$$

T represents the torque applied, $M(q)$ represents the mass matrix and $C(q, \dot{q})$ represents the centrifugal and coriolis terms. Equation (1.1) can also be written as a linear regression as follows.

$$T = Y(q, \dot{q}, \ddot{q})P \quad (1.4)$$

where

$$Y = \begin{bmatrix} \ddot{q}_1 & 2\ddot{q}_1 + \ddot{q}_2 + 2\ddot{q}_1 \cos(q_2) - 2\dot{q}_1 \dot{q}_2 \sin(q_2) + \ddot{q}_2 \cos(q_2) - \dot{q}_2^2 \sin(q_2) \\ 0 & \ddot{q}_1 + \ddot{q}_1 \cos(q_2) + \ddot{q}_2 + \dot{q}_1^2 \sin(q_2) \end{bmatrix} \quad (1.5)$$

and

$$P = \begin{bmatrix} m_1 f^2 \\ m_2 f^2 \end{bmatrix}. \quad (1.6)$$

Often the parameter values, $m_1 f^2$ and $m_2 f^2$ are replaced with θ_1 and θ_2 or p_1 and p_2 . These are the parameter values that we tune in the controller until they reach the true values. Once the true values are reached, the desired output equals the current output and the adaptation stops. The input used in the simulations involving the system in Figure 5, is a sequence of unit steps with a period of 8 seconds.

Chapter 2

2.0 Output Feedback Nonlinear Adaptive Control Using Only Gains In The Feedback Loop

This algorithm, as proposed by Jankovic [6], is for a general model of nonlinear output feedback control. The advantage of this algorithm is that it has a proof of stability. The algorithm is proved to be semi-globally stable. We extend this algorithm to the robotics case and analyze its region of convergence and stability properties. Though the algorithm has a proof of stability, we cannot conclude that the extension of the algorithm to the two degree of freedom robotic manipulator is also semi-globally stable. This is because the original algorithm [6] was modified when applied to the robotics case. But the simulations do show that this robotic extension has a region of convergence though it is not globally stable. The analysis of the proof of stability provided by Jankovic [6] and its extension to the robotics case is out of scope of this thesis and is left as part of the future work.

2.1 Algorithmic Details

The output feedback nonlinear adaptive control system of a two degree of freedom robotic manipulator is shown in Figure 6.

Figure 6: Output feedback nonlinear adaptive control for a two degree of freedom robotic system

For each link, the error E between the desired and actual link position is defined as

$$E = q_d - q \quad (2.1)$$

where q_d is the desired link position and q is the actual link position.

Similarly, for each link, the error between the desired and the actual link velocity, $\dot{E}$, is defined as

$$\dot{E} = \dot{q}_d - \dot{\hat{q}} \quad (2.2)$$

where $\dot{q}_d$ is the desired link velocity and $\dot{\hat{q}}$ is the actual link velocity.

The vector x is defined as the composite of both the errors,

$$x = \begin{bmatrix} E \\ \dot{E} \end{bmatrix}. \quad (2.3)$$

For a two-degree of freedom robot the vector x would be a 4x1 vector as E and $\dot{E}$ would both have two components, one for each link. Therefore, x would be

$$x = \begin{bmatrix} e_1 \\ e_2 \\ \dot{e}_1 \\ \dot{e}_2 \end{bmatrix} \quad (2.4)$$

where e_i and $\dot{e}_i$ represent the position and velocity errors of link i .

This vector x is then used in determining the virtual control law, v . v is called the virtual control law as it is not directly fed into the Robot (Figure 6). Instead, v goes through a linearization block before being fed to the robot. The linearization law is used to make the nonlinear robotic manipulator system appear linear. In Figure 6, the dashed lines show the linearization block and the robot. Within the dashed lines, the linearization block has been shown with the details. It is composed of the mass matrix estimate, $\hat{M}$, and the coriolis matrix estimate, $\hat{C}$, along with the summer. The virtual control law is defined as

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \ddot{q}_{d1} \\ \ddot{q}_{d2} \end{bmatrix} - Kx \quad (2.5)$$

The matrix K in Equation (2.5) is a composite of the two constants, k_D and k_P . K constitutes the feedback loop of the system,

$$K = \begin{bmatrix} k_P & 0 & k_D & 0 \\ 0 & k_P & 0 & k_D \end{bmatrix} \quad (2.6)$$

The k_P and k_D values can be found by assuming perfect linearization. If perfect linearization is assumed then the closed loop transfer function must equal the model transfer function, W_m , shown below in Equation (2.7).

$$W_m(s) = \frac{1}{s^2 + 10s + 25} . \quad (2.7)$$

The model transfer function in Equation (2.7) is chosen to be a critically damped system with a bandwidth of five radians/sec.

Redrawing Figure 6, assuming perfect linearization, yields the system shown in Figure 7.

Figure 7: Robotic manipulator system assuming perfect linearization

By performing block diagram reduction on the inner loop (with constant k_D) we get the following transfer function

$$TF_1 = \frac{\frac{1}{s}}{1 + \frac{k_D}{s}} = \frac{1}{s + k_D} . \quad (2.8)$$

Using TF_1 and the second integrator as a series system and performing block diagram reduction on the outer loop (with constant k_p) yields the following closed loop transfer function

$$W(s) = \frac{\left(\frac{1}{s}\right)\left(\frac{1}{s+k_D}\right)}{1 + \frac{k_p}{s(s+k_D)}} = \frac{1}{s^2 + k_D s + k_p}. \quad (2.9)$$

Since, perfect linearization is assumed the closed loop transfer function $W(s)$ should equal the model transfer function as shown in Equation (2.7). Equating Equation (2.7) and (2.9) we can determine the k_D and k_p values to be 10 and 25 respectively.

Once the virtual control law has been calculated, the linearization law is used to calculate the torque applied to the robot. The linearization law is defined as

$$T_L = \hat{M}v + \hat{C}. \quad (2.10)$$

The estimated mass matrix, $\hat{M}$, and the estimated Coriolis matrix, $\hat{C}$, are used to calculate the torque because the true parameter values are unknown. Similar to Equation (1.4), Equation (2.10) can also be written in linear regression form as

$$T_L = \hat{Y}\hat{P}, \quad (2.11)$$

where $\hat{P}$ is the vector containing the estimated parameter for link 1, $\hat{p}_1$, and the estimated parameter for link 2, $\hat{p}_2$, as shown in Equation (2.12).

$$\hat{P} = \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \end{bmatrix} \quad (2.12)$$

$\hat{Y}$ is defined as shown in Equation (2.13).

$$\hat{Y} = \begin{bmatrix} v_1 & 2v_1 + v_2 + 2v_1 \cos(q_2) - 2\dot{\hat{q}}_1 \dot{\hat{q}}_2 \sin(q_2) + v_2 \cos(q_2) - \dot{\hat{q}}_2^2 \sin(q_2) \\ 0 & v_1 + v_1 \cos(q_2) + v_2 + \dot{\hat{q}}_1^2 \sin(q_2) \end{bmatrix} \quad (2.13)$$

The $\dot{\hat{q}}_1, \dot{\hat{q}}_2$ values in Equation (2.13) are the observer estimated values of the link velocities. This observer in general form is given by

$$\begin{bmatrix} \dot{\hat{q}}_i \\ \ddot{\hat{q}}_i \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{q}_i \\ \dot{\hat{q}}_i \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} (q_i - \hat{q}_i) \quad (2.14)$$

where $i = 1, 2$ for a two degree of freedom robotic manipulator. The l_1 and l_2 values in Equation (2.14) are chosen such that the observer bandwidth is two times faster than the model bandwidth. The model shown has a double pole at $s=-5$ as is easily determined by the transfer function in Equation (2.7). Therefore, the vector L is chosen such that the observer has double poles at $s=-10$. These $s=-10$ poles yield a characteristic equation of $s^2 + 20s + 100 = 0$. (2.15)

The value of the vector L can then be determined by setting the $\det(sI - (A - LC))$ equal to the characteristic equation of the desired observer bandwidth, Equation (2.15). The values of matrix A, I and C are

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}. \quad (2.16)$$

Using vector $L = \begin{bmatrix} l_1 \\ l_2 \end{bmatrix}$, $\det(sI - (A - LC))$ is determined as follows,

$$\det(sI - (A - LC)) = \det \left(s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \right) \quad (2.17)$$

$$= \det \left(s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} l_1 & 0 \\ l_2 & 0 \end{bmatrix} \right) \right) \quad (2.18)$$

$$= \det \left(s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} -l_1 & 1 \\ -l_2 & 0 \end{bmatrix} \right) \quad (2.19)$$

$$= \det \begin{pmatrix} s + l_1 & -1 \\ l_2 & s \end{pmatrix} \quad (2.20)$$

$$= s^2 + l_1 s + l_2. \quad (2.21)$$

We now set Equation (2.21) equal to Equation (2.15). Therefore, the values of l_1 and l_2 are 20 and 100 respectively. The observer can be determined by substituting the values of A, b and L into Equation (2.14). This yields the following observer,

$$\begin{bmatrix} \dot{\hat{q}}_i \\ \ddot{\hat{q}}_i \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{q}_i \\ \dot{\hat{q}}_i \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_i + \begin{bmatrix} 20 \\ 100 \end{bmatrix} (q_i - \hat{q}_i). \quad (2.22)$$

Using these estimated values the torque can be calculated using Equation (2.11).

This torque is then used to find the link acceleration which is given by

$$\ddot{q} = M^{-1}(T_L - C), \quad (2.23)$$

where

$$M(q) = \begin{bmatrix} m_1 f^2 + 2m_2 f^2 + 2m_2 f^2 \cos(q_2) & m_2 f^2 + m_2 f^2 \cos(q_2) \\ m_2 f^2 + m_2 f^2 \cos(q_2) & m_2 f^2 \end{bmatrix} \quad (2.24)$$

and

$$C(q, \dot{q}) = \begin{bmatrix} -2m_2 f^2 \dot{q}_1 \dot{q}_2 \sin q_2 - m_2 f^2 \dot{q}_2^2 \sin q_2 \\ m_2 f^2 \dot{q}_1^2 \sin q_2 \end{bmatrix}. \quad (2.25)$$

Here we use the true values for the parameters because Equation (2.23) represents the actual output of the robot. Finally, the adaptation law for the parameter error, ϕ , is defined as

$$\dot{\phi} = -c_1 \hat{Y} M^{-1} (\dot{e} - \eta), \quad (2.26)$$

where

$$c_1 = 15 \quad (2.27)$$

$$\dot{e} = \begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \end{bmatrix} \text{ as in Equation (2.4)} \quad (2.28)$$

$$\dot{\eta} = c_o(\dot{e} - \eta) - Kx \quad (2.29)$$

and

$$c_o = 2. \quad (2.30)$$

In Equation (2.26), the inverse of the true mass matrix, M , is used. The use of the true mass matrix is a drawback for using this extension algorithm for robotic manipulators. The mass matrix depends on the parameter values, $m_1 f^2$ and $m_2 f^2$, which are unknown and hence this true mass matrix is also unknown in a real situation. As a result, using this extension of Jankovic's algorithm is not feasible for practical situations of robotic manipulators. A more practical algorithm would be to use the estimate of M , $\hat{M}$, instead of the true matrix M . The algorithm using $\hat{M}$ was tried for a couple of simulation scenarios and the algorithm converged. The results for these scenarios were very similar to the results of the algorithm using the true mass matrix, M . The detailed analysis of the algorithm with the $\hat{M}$ matrix is not included in this thesis and is left as part of the future work.

2.2 Results

The performance of the algorithm was studied through various simulations. In the simulations the observer was initialized to zero. In this section we present the results of the simulations. The graphs for three scenarios are shown. The rest of the scenario results are summarized in a table at the end of the section. The data captured includes the amplitude of the input signal, the initial parameter estimates, the convergence/divergence

of the algorithm and the time it takes for the parameter error, e_p , to reduce to zero if the algorithm converges. In the different scenarios tried, the input amplitude and the initial parameter estimates were varied. The goal is to determine how varying these values affects the convergence of the algorithm.

First, we analyze the scenario with input amplitude of 1 and initial parameter estimates set at (1,3). The true values for the parameters are (1,2). The graph in Figure 8 shows the trajectory error for the two links. This error is reduced to zero and hence the algorithm converges.

Figure 8: OFKx: Trajectory error for amplitude 1, initial parameter estimates: 1,3

The acronym OFKx is used to represent this output feedback nonlinear adaptive control using only gain in the feedback loop algorithm.

Next we show the parameter error. Figure 9 shows the errors between the estimated parameter values and the true parameter values.

Figure 9: OFKx: Parameter error for amplitude 1, initial parameter estimates: 1,3

The parameter error also reduces to zero. Therefore, with an input amplitude of 1 and initial parameter estimates set to (1,3), the algorithm converges.

We now consider a higher initial parameter estimation error scenario. This time the initial parameter estimates are set to (10,8) whereas the input amplitude is still set to 1. The goal is to determine how a higher initial parameter estimation error at a low input amplitude effects the algorithm's convergence. The graphs for the trajectory error and the parameter error are shown in Figure 10 and Figure 11.

Figure 10: OFKx: Trajectory error for amplitude 1, initial parameter estimates: 10,8

Figure 11: OFKx: Parameter error for amplitude 1, initial parameter estimates: 10,8

The trajectory error and parameter error both reduce to zero which indicates that the algorithm has converged. We now want to determine the effect of a higher input amplitude on the convergence. So, we set the input amplitude to 2 and the initial parameter estimates to (1,3). This scenario diverged. The initial parameter estimates were then set to the true values of (1,2) at input amplitude 2 to determine if any convergence region existed at this amplitude. The algorithm converged with the true initial parameter values. The trajectory error and parameter errors for this scenario are shown in Figure 12 and Figure 13.

Figure 12: OFKx: Trajectory error for amplitude 2, initial parameter estimates: 1,2

Figure 13: OFKx: Parameter error for amplitude 2, initial parameter estimates: 1,2

From Figure 12 and Figure 13 it can be seen that the algorithm is able to track the desired trajectory if the initial parameters are set to the true values. As expected the trajectory and parameter errors start at zero and stay at zero. The fact that this scenario converged means that some region of convergence exists at input amplitude 2. Various other simulations were tried to study the convergence region further. These results are summarized in Table 1. The times for $e_p \rightarrow 0$ column were determined by considering any $e_p < 0.002$ as zero. Therefore, the time recorded was the first time that $e_p < 0.002$.

Table 1: OFKx: Other results

Input Amplitude	Initial Parameter Estimates (True Values: (1,2))	Converges/Diverges	$e_p \rightarrow 0$ Time (sec)
1	(1,3)	C	1000
1	(0.5,1)	D	-
1	(1.5, 3)	C	900
1	(0.2,0.4)	D	-
1	(5,5)	C	800
1	(10,8)	C	1000
2	(1,3)	D	-
2	(1,3.1)	D	-
2	(1,3.2)	D	-
2	(1,2)	C	0
3	(1,3)	D	-
3	(1,2)	D	-
10	(1,2)	D	-

From the above results, we can conclude that this algorithm has a region of convergence for low input amplitudes of 1 and 2. For higher amplitudes, the algorithm is unstable and has no region of convergence because it is unable to converge even with the initial parameter estimates set to the true values. This can be seen from the input amplitude 3 and 10 scenario where the initial parameter estimates were set to the true values of (1,2) and the algorithm diverged.

Chapter 3

3.0 Model Reference Nonlinear Output Feedback Adaptive Control (MRAC) - With Observer

In this section, the MRAC algorithm [1] with a linear observer is discussed. An observer is used to estimate the states since all the states are not available for measurement. The primary goal is to study the region of convergence and the stability properties of the algorithm. The secondary goal is to determine if the proposed filter in the algorithm has an effect on the stability and convergence of the algorithm. For this algorithm we study both a 2nd and a 3rd order system. Hence, four algorithms are discussed. The 2nd order system, with and without the filter is discussed in Section 3.1 and the 3rd order system, with and without the filter is discussed in Section 3.2.

Figure 14 shows the structure of Model Reference Nonlinear Output Feedback Adaptive Control. Notice that in this structure, the feedback loop is not a simple K gain as in the previous algorithm structure shown in Figure 6.

3.1 2nd Order System

3.1.1 Algorithmic Details

This algorithm is as presented by Schwartz [8]. The algorithm was simulated using a two-degree of freedom robot. This system is shown in Figure 14:

Figure 14: MRAC 2-degree of freedom robotic manipulator system

The model used for each link is a critically damped system with a bandwidth of two radians/second. This model is defined by the transfer function shown in Equation (3.1).

$$W_m(s) = \frac{4}{s^2 + 4s + 4} \quad (3.1)$$

In state space form this model is

$$\begin{bmatrix} \dot{q}_{mi} \\ \ddot{q}_{mi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -4 & -4 \end{bmatrix} \begin{bmatrix} q_{mi} \\ \dot{q}_{mi} \end{bmatrix} + \begin{bmatrix} 0 \\ 4 \end{bmatrix} q_{ci} \quad (3.2)$$

The virtual control law for each link can be computed by assuming perfect linearization.

This control law is given by

$$v_i = \frac{-4}{s+10} v_i + \frac{400}{s+10} q_i - 44q_i + 4q_{ci} \quad (3.3)$$

The details of this virtual control derivation can be found from Equation (3.37) – Equation (3.47). The values found in Equation (3.47) by assuming perfect linearization are substituted back into Equation (3.37) to get the virtual control law shown in Equation (3.3).

Once the virtual control law is calculated, the torque applied to the robot is computed by the following equation

$$T_L = Y(q, \dot{q}, v) \hat{P} \quad (3.4)$$

where

$$Y = \begin{bmatrix} v_1 & 2v_1 + v_2 + 2v_1 \cos(q_2) - 2\dot{\hat{q}}_1 \dot{\hat{q}}_2 \sin(q_2) + v_2 \cos q_2 - \dot{\hat{q}}_2^2 \sin(q_2) \\ 0 & v_1 + v_1 \cos(q_2) + v_2 + \dot{\hat{q}}_1^2 \sin(q_2) \end{bmatrix} \quad (3.5)$$

and

$$\hat{P} = \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \end{bmatrix}. \quad (3.6)$$

The estimated $\dot{\hat{q}}_1$ and $\dot{\hat{q}}_2$ values in Equation (3.5) are calculated using the linear observer shown in Equation (3.7).

$$\begin{bmatrix} \dot{\hat{q}}_i \\ \ddot{\hat{q}}_i \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{q}_i \\ \dot{\hat{q}}_i \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v_i + \begin{bmatrix} 20 \\ 100 \end{bmatrix} (q_i - \hat{q}_i) \quad (3.7)$$

Again, the values of the vector $L = \begin{bmatrix} 20 \\ 100 \end{bmatrix}$, can be determined as shown in Equation(2.16)–

Equation (2.21). Finally, the adaptation law for the parameter error, ϕ , is defined as follows.

$$\dot{\phi} = \hat{E}(t) e_v \quad (3.8)$$

$$\text{where } e_v = v - \ddot{q}_m \quad (3.9)$$

and e_v represents the virtual error. If perfect linearization occurs then the virtual error is zero. This means that

$$v = \ddot{q}_m. \quad (3.10)$$

The term $\hat{E}(t)$ is defined as

$$\hat{E}(t) = W_\phi(s)Y(q, \dot{q}, v) \quad (3.11)$$

$$\text{and } W_\phi(s) = \frac{44s + 44}{s^3 + 14s^2 + 44s + 40}. \quad (3.12)$$

The filter in Equation (3.12) is derived using the relationship between the virtual error and the parameter error. Before we can derive this relationship we briefly discuss the general solution of the control of nonlinear systems. The idea is to differentiate the output until the input to the nonlinear system is seen in the derivative. As seen from Figure 14, for our nonlinear 2-degree of freedom robotic manipulator, the input to the nonlinear robot is the linearization law, T_L . Therefore, we take the derivative of the output until, T_L , appears in the derivative. The derivatives of the output are as follows.

$$\begin{aligned} \varphi_1 &= q \\ \varphi_2 &= \dot{q} \\ \varphi_3 &= \ddot{q} = b + aT_L \end{aligned} \quad (3.13)$$

where a and b are arbitrary constants. In this case, the input to the nonlinear robot, T_L , is seen in the second derivative of the output and hence, $\ddot{q}$ is some function that includes the input, T_L . We choose, the linearization law, T_L , such that the output, $\ddot{q}$, equals the external input acceleration. In our case, from Figure 14, this external input acceleration is v . Therefore, we choose the linearization law as follows:

$$b + aT_L = v \quad (3.14)$$

$$T_L = \frac{v - b}{a} = -\frac{b}{a} + \frac{v}{a}. \quad (3.15)$$

The linearization law in Equation (3.15) can also be written in regression form as

$$T_L = Y(q, \dot{q}, v)P \quad (3.16)$$

which is the same form as shown in Equation (3.4). Substituting Equation (3.16) into Equation (3.14) we get Equation (3.17) below. To simplify the notation the variables of $Y(q, \dot{q}, v)$ are not shown explicitly and Y is used to represent $Y(q, \dot{q}, v)$.

$$b + aYP = v \quad (3.17)$$

Also, note that, substituting Equation (3.15) back into Equation (3.13) will yield

$$\ddot{q} = v \quad (3.18)$$

which occurs when perfect linearization is achieved. This means that the linearization block and the nonlinear robot shown in Figure 14, act as a double integrator when perfect linearization is achieved. Now, we can start the derivation for the filter in Equation(3.12). The parameter error is defined as

$$\phi = P - \hat{P}. \quad (3.19)$$

Substituting for $\hat{P}$ from Equation (3.19) into Equation (3.4), the linearization law can also be written as

$$T_L = Y(P - \phi) = YP - Y\phi. \quad (3.20)$$

Substituting the linearization law from Equation (3.20) into the general form equation shown in Equation (3.13) yields

$$\ddot{q} = b + a\{YP - Y\phi\} \quad (3.21)$$

$$\ddot{q} = b + aYP - aY\phi. \quad (3.22)$$

The first part of Equation (3.22) is equal to Equation (3.17), which represents the exact linearization control law. Substituting Equation (3.17) into Equation (3.22) yields

$$\ddot{q} = v - aY\phi. \quad (3.23)$$

The output of the system in the Laplace domain is $q(s)$ and can be written as

$$q(s) = \left(\frac{1}{s^n} \right) \ddot{q}(s) \quad (3.24)$$

where $n=2$. Equation (3.24) is just integrating the second derivative, $\ddot{q}$, twice to get q .

Substituting Equation (3.23) into Equation (3.24) gives

$$q(s) = \left(\frac{1}{s^n} \right) [v(s) - aY\phi] \quad (3.25)$$

$$q(s) = \left(\frac{1}{s^n} \right) v(s) - \left(\frac{1}{s^n} \right) aY\phi. \quad (3.26)$$

From Figure 14, the virtual control law, $v(s)$, can also be written as

$$v(s) = kq_c(s) + \frac{d_1(s)}{\lambda(s)} v(s) + \frac{d_2(s)}{\lambda(s)} q(s) + d_o q(s). \quad (3.27)$$

In Equation (3.27) the $d_1(s), d_2(s)$ terms are the numerator polynomials of the transfer functions $D_1^T(sI - \Lambda)^{-1}b$ and $D_2^T(sI - \Lambda)^{-1}b$ respectively. The $\lambda(s)$ term is the denominator polynomial of the $D_1^T(sI - \Lambda)^{-1}b$ and $D_2^T(sI - \Lambda)^{-1}b$ transfer functions.

Substituting for $q(s)$ from Equation (3.26) into Equation (3.27) yields,

$$v(s) = kq_c(s) + \frac{d_1(s)}{\lambda(s)} v(s) + \frac{d_2(s)}{\lambda(s)} \left[\left(\frac{1}{s^n} \right) v - \left(\frac{1}{s^n} \right) aY\phi \right] + d_o \left[\left(\frac{1}{s^n} \right) v - \left(\frac{1}{s^n} \right) aY\phi \right]. \quad (3.28)$$

Rearranging Equation (3.28) we get

$$\left[\frac{s^n(\lambda(s) - d_1(s)) - d_2(s) - d_o\lambda(s)}{s^n\lambda(s)} \right] v(s) = kq_c(s) - \left(\frac{1}{s^n} \right) \left[\frac{d_2(s)}{\lambda(s)} + d_o \right] aY\phi \quad (3.29)$$

Solving for $v(s)$ gives

$$v(s) = \frac{s^n\lambda(s)kq_c(s) - [d_2(s) + d_o\lambda(s)]aY\phi}{s^n(\lambda(s) - d_1(s)) - d_2(s) - d_o\lambda(s)}. \quad (3.30)$$

Assuming perfect linearization, the model transfer function is given by

$$W_m(s) = \frac{\lambda(s)k}{s^n(\lambda(s) - d_1(s)) - d_2(s) - d_0\lambda(s)} . \quad (3.31)$$

This means that Equation (3.30) can be written as

$$v(s) = s^n W_m(s) q_c(s) - \frac{d_2(s) + d_o\lambda(s)}{s^n(\lambda(s) - d_1(s)) - d_2(s) - d_0\lambda(s)} [aY\phi] \quad (3.32)$$

where

$$s^n W_m(s) q_c(s) \quad (3.33)$$

represents the n^{th} derivative of the model output. This is true because $q_m(s) = W_m(s)q_c(s)$ and the s^n represents the n^{th} derivative in the Laplace domain. (For a 2nd order desired model for each link, n equals 2.)

Recall that the virtual error is given by

$$e_v = v - \ddot{q}_m . \quad (3.34)$$

Therefore, the relationship between the virtual error and the parameter error is given by

$$e_v(s) = - \frac{d_2(s) + d_o\lambda(s)}{s^n(\lambda(s) - d_1(s)) - d_2(s) - d_0\lambda(s)} [aY\phi] . \quad (3.35)$$

And finally, we define

$$W_\phi(s) = - \frac{d_2(s) + d_o\lambda(s)}{s^n(\lambda(s) - d_1(s)) - d_2(s) - d_0\lambda(s)} . \quad (3.36)$$

Assuming perfect linearization, and treating each link as a double integrator, one can calculate the values for k , $d_1(s)$, $d_2(s)$, d_o and λ . Using the $d_1(s)$, $d_2(s)$, $\lambda(s)$ polynomials and assuming perfect linearization, the system can be modeled as shown in Figure 15.

Figure 15: MRAC perfect linearization system

From Figure 15, the virtual control law can be written as

$$v = q_c k + \frac{d_1(s)}{\lambda(s)} v + \frac{d_2(s)}{\lambda(s)} q + d_o q. \quad (3.37)$$

Due to perfect linearization, $v = s^2 q$. Substituting this into Equation (3.37) we get

$$s^2 q = q_c k + \frac{d_1(s)}{\lambda(s)} s^2 q + \frac{d_2(s)}{\lambda(s)} q + d_o q. \quad (3.38)$$

Rearranging Equation (3.38) we get

$$q_c k = s^2 q - \frac{d_1(s)}{\lambda(s)} s^2 q - \frac{d_2(s)}{\lambda(s)} q - d_o q. \quad (3.39)$$

The cancelable zero for this system has been chosen to be $s=-10$. This means that

$\lambda(s) = (s+10)$. Substituting for $\lambda(s)$ in Equation (3.39) we get

$$q_c k = s^2 q - \frac{d_1(s)}{(s+10)} s^2 q - \frac{d_2(s)}{(s+10)} q - d_o q. \quad (3.40)$$

$$q_c k = \frac{(s+10)s^2 q - d_1(s)s^2 q - d_2(s)q - (s+10)d_o q}{(s+10)}. \quad (3.41)$$

Rearranging Equation (3.41) to determine the transfer function we get

$$\frac{q}{q_c} = \frac{k(s+10)}{(s+10)s^2 - d_1(s)s^2 - d_2(s) - (s+10)d_o} \quad (3.42)$$

$$\frac{q}{q_c} = \frac{k(s+10)}{s^3 + (10 - d_1(s))s^2 - d_o s + 10 - d_2(s)} \quad (3.43)$$

We know that Equation (3.43) has to equal the model of the system. From Equation (3.1) the model is

$$W_m(s) = \frac{4}{s^2 + 4s + 4} \quad (3.44)$$

It is easier to determine the values in Equation (3.43) if we multiply the top and bottom of the model by the cancelable zero $(s+10)$.

$$W_m(s) = \frac{4(s+10)}{(s^2 + 4s + 4)(s+10)} \quad (3.45)$$

Simplifying the above equation we get

$$W_m(s) = \frac{4(s+10)}{(s^3 + 14s^2 + 44s + 40)} \quad (3.46)$$

Comparing Equation (3.46) with Equation (3.43) we can determine that

$$\begin{aligned} k &= 4 \\ 10 - d_1(s) &= 14 \Rightarrow d_1(s) = -4 \\ -d_o &= 44 \Rightarrow d_o(s) = -44 \\ -10d_o - d_2(s) &= 40 \Rightarrow d_2(s) = 400 \end{aligned} \quad (3.47)$$

Therefore, the values are $k=4$, $\lambda(s)=s+10$, $d_1(s)=-4$, $d_2(s)=400$, $d_o(s)=-44$ and $b=1$.

Substituting these values into Equation (3.36) one can get the value of the filter,

$$W_\phi(s) = -\frac{400 + -44(s+10)}{s^2((s+10) - (-4)) - 400 - (-44)(s+10)} \quad (3.48)$$

which simplifies to

$$W_{\phi}(s) = \frac{44s + 44}{s^3 + 14s^2 + 44s + 40} \quad (3.49)$$

which equals Equation (3.12).

3.1.2 Results

In this section, we analyze the performance of the algorithm through various simulation scenarios. The initial values are all set to zero except the initial parameter estimates that vary from scenario to scenario. The data captured includes the amplitude of the input signal, the initial parameter estimates, the convergence/divergence of the algorithm and the time it takes for the parameter error, e_p , to reach zero if the algorithm converges. In our analysis, if the parameter error goes to zero then the algorithm is considered to have converged else its considered to have diverged. The true parameters are (1,2).

First we look at the lower input amplitudes. For input amplitude 1 and initial parameter estimates set at (1,3) the algorithm converges. Here the initial parameter estimates are set to 0% and 50% error for parameter 1 and parameter 2 respectively. The graph in Figure 16 shows the trajectory error for this simulation.

Figure 16: MRAC (F): Trajectory Error for Amplitude 1, Initial Parameter Estimates: 1,3

The acronym MRAC (F) is used to represent this Model Reference Nonlinear Output Feedback Adaptive Control with filter in the adaptation law algorithm.

This algorithm converges since the trajectory error goes to zero. The error reduces to zero in about 35 seconds. Next we analyze the parameter error graph. Parameter error, e_p , also reduces to zero as shown by the Figure 17 below.

Figure 17: MRAC (F): Parameter error for amplitude 1, initial parameter estimates: 1,3

Next we consider a greater initial parameter estimate at the same input amplitude. For this simulation, we use initial parameter estimates of (10,8). This represents 1000% error and 400% error which is quite a large error. The goal here is to determine the effect of a higher initial parameter estimation error on the stability of the algorithm. The graphs of the trajectory error and the parameter error are shown in Figure 18 and Figure 19.

Figure 18: MRAC (F): Trajectory error for amplitude 1, initial parameter estimates: 10,8

Figure 19: MRAC (F): Parameter error for amplitude 1, initial parameter estimates: 10,8

Since, the errors are reduced to zero, this algorithm converges. Also, as expected, due to the larger initial parameter estimation error, e_p takes longer to converge to zero. Here it takes just about 170 seconds. From this simulation we can conclude that the time it takes for the algorithm to converge depends on the initial parameter estimates. This conclusion was confirmed via various other simulations that are summarized in Table 2.

Next we consider a higher input amplitude and study its effect on the convergence region. For the next simulation we start the parameter values at (1,3) with an input amplitude of 2. The trajectory error and parameter error graphs are shown in Figures 20 and Figure 21.

Figure 20: MRAC (F): Trajectory error for amplitude 2, initial parameter estimates: 1,3

Figure 21: MRAC (F): Parameter error for amplitude 2, initial parameter estimates: 1,3

From Figures 20 and Figure 21 we can see that both the trajectory and the parameter errors are oscillating. Since, the error is not reducing to zero, the algorithm is considered to have diverged. The same simulation with amplitude 1 had converged (Figure 16 and Figure 17). Therefore, we see that as the input amplitude increases, the region of convergence decreases.

Several other simulation scenarios were tried to further analyze the effect of the initial parameter estimates and the input amplitude on the convergence of the algorithm. The results obtained for the two-degree of freedom robotic manipulator are summarized in the Table 2 below. Again, the times for $e_p \rightarrow 0$ column were determined by

considering any $e_p < 0.002$ as zero. Therefore, the time recorded was the first time that $e_p < 0.002$.

Table 2: MRAC (F): Other results

Input Amplitude	Initial Parameter Estimates (True Values: (1,2))	Converges/Diverges	$e_p \rightarrow 0$ Time (sec)
1	1,3	Converges	35
1	0.5,1	Converges	40
1	1.5,3	Converges	50
1	0.2,0.4	Converges	15
1	0,0	Converges	25
1	5,5	Converges	120
1	10,8	Converges	170
2	1,3	Diverges	-
2	1,3.1	Diverges	-
2	1,3.2	Diverges	-
2	1,2	Converges	0
3	1,3	Diverges	-
3	1,2	Diverges	-
10	1,2	Diverges	-

As seen from Table 2 above, for input amplitude of 1, it can take between 35-170 seconds for the parameter error to reduce to zero, depending on the initial parameter estimates. Also, as we increase the input amplitude, the algorithm starts to diverge. This indicates that as the input amplitude increases the region of convergence decreases. If the input amplitude is increased high enough then the algorithm diverges even when the initial parameter estimates are set to the true values. This can be seen by the simulations in Table 2 where the input amplitudes of 3 and 10 were simulated with the initial parameter estimates set to the true values. At these amplitudes the algorithm is unstable.

The $W_\phi(s)$ filter used in Equation (3.12) is quite complex. Our secondary goal is to study the effect of this filter and determine if the implementation can be made simpler

by removing this filter. We also want to determine if this filter has any effect on the stability of the algorithm. We modified the above algorithm by changing the adaptation law from Equation (3.8) to

$$\dot{\phi} = Y(q, \dot{q}, v)e_v \quad (3.50)$$

where

$$e_v = v - \ddot{q}_m. \quad (3.51)$$

The only difference is that we made $W_\phi(s) = 1$.

We now look at the simulation results of this new algorithm. First, we consider a low input amplitude simulation. Here the initial parameter estimates were set to (1,3) and the amplitude was set to 1. The graph for the trajectory error is shown in Figure 22.

Figure 22: MRAC (WF): Trajectory error for amplitude 1, initial parameter estimates: 1,3

The acronym MRAC (WF) is used to represent this Model Reference Nonlinear Output Feedback Adaptive Control without the filter in the adaptation law algorithm.

Since, the trajectory error reduces to zero we can conclude that the robot is able to track the desired trajectory. Next we look at the parameter error.

Figure 23: MRAC (WF): Parameter error for amplitude 1, initial parameter estimates: 1,3

We see that it takes about 40 seconds for the parameter error to reduce to zero. This is approximately the same time as it took with the filter, $W_\phi(s)$. Therefore, no significant difference is noticed at the lower amplitude and a small initial parameter estimation error. Next, we consider a higher initial parameter estimation error. We now

set the initial parameter estimates to (10,8). This simulation diverges. The trajectory error is shown in Figure 24.

Figure 24: MRAC (WF): Trajectory error for amplitude 1, initial parameter estimates: 10,8

The error in Figure 24 the trajectory error is oscillating and hence, the algorithm has diverged. The divergence is more apparent with the parameter error graph shown below in Figure 25.

Figure 25: MRAC (WF): Parameter error for amplitude 1, initial parameter estimates: 10,8

As we can see, parameter 1 error is continuously increasing and is at -15 within 100 seconds of the simulation. We can compare this simulation to the previous simulation which had the filter $W_\phi(s)$ in the adaptation law. The previous simulation is shown in Figure 18 and Figure 19. Comparing these two simulations we notice that the previous simulation had converged where as this simulation has diverged. This indicates that there is some effect of the filter on the stability of the algorithm.

Next we consider our final simulation with an increase in the amplitude. We now consider an input amplitude of 2 with the initial parameter set at (1,3). The trajectory error and parameter error graphs are in Figure 26 and Figure 27.

Figure 26: MRAC (WF): Trajectory error for amplitude 2, initial parameter estimates: 1,3

Figure 27: MRAC (WF): Parameter error for amplitude 2, initial parameter estimates: 1,3

The trajectory and parameter errors have reduced to zero in about 115 seconds. Looking at the corresponding simulation in Figure 21, which had the filter in the adaptation law, we notice that that simulation had diverged. This means, that removing the filter has a positive effect for higher amplitude simulations. To study the effect of the filter in more detail, various other scenarios were simulated. These results are shown in Table 3.

Table 3: MRAC (WF): Other results

Input Amplitude	Initial Parameter Estimates (True Values: (1,2))	Converges/Diverges	$e_p \rightarrow 0$ Time (sec)
1	1,3	Converges	40
1	0.5,1	Converges	55
1	1.5,3	Converges	70
1	0.2,0.4	Converges	60
1	0,0	Converges	50
1	5,5	Converges	450
1	10,8	Diverges	-
2	1,3	Converges	115
2	1,3.1	Converges	120
2	1,3.2	Converges	140
3	1,3	Diverges	-
3	1,2	Diverges	-
10	1,2	Diverges	-

The results indicate that this algorithm has a region of convergence for input amplitude of 2. In the previous algorithm where the filter was part of the adaptation law, none of the amplitude 2 scenarios converged except when the parameter values were set to the true values. In this algorithm we see that all of the scenarios tried for amplitude 2 converged. Hence, this algorithm works better than the previous one for higher amplitudes. Though, if we analyze the lower amplitude scenarios we see that the time it takes for the parameter error to reduce to zero has increased. Not only this but the scenario with amplitude 1 and initial parameter estimates (10,8) that converged before has now diverged.

From these simulations we can conclude that the algorithm without the filter works better for higher amplitudes, whereas the algorithm with the filter in the adaptation law works better for lower amplitudes. To develop further insights into the effect of the filter on the algorithm, we now analyze the same algorithm on a third order system.

3.2 3rd Order System

3.2.1 Algorithmic Details

In this section we determine the effect of the filter on the 3rd order system. Since, we require a 3rd order system, the robotic manipulator described in Section 1.2 cannot be used. The 3rd order system used here is a Single Input, Single Output (SISO) system [8]. The equations for this system are given by

$$\begin{aligned}\dot{x}_1 &= x_2 + 2x_1^2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= 5u\end{aligned}\tag{3.52}$$

The output is given by

$$y = x_1\tag{3.53}$$

As described in Section 3.1.1, the goal is to take the derivative of the output, y , until the input, u , is seen in the derivative. A new state, z , is defined which contains the derivatives of the output, y . A state transformation is performed to find the new state z , which is given by

$$\Phi(x) = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} x_1 \\ \dot{x}_1 \\ \ddot{x}_1 \end{bmatrix}.\tag{3.54}$$

We can determine the value of $\ddot{x}_1$ by taking the derivative of $\dot{x}_1$ which is defined in Equation (3.52). Therefore,

$$\ddot{x}_1 = \dot{x}_2 + 4x_1\dot{x}_1.\tag{3.55}$$

Again substituting in for $\dot{x}_1$ from Equation (3.52) into Equation (3.55) we get

$$\ddot{x}_1 = \dot{x}_2 + 4x_1[x_2 + 2x_1^2].\tag{3.56}$$

Simplifying Equation (3.56) yields

$$\ddot{x}_1 = \dot{x}_2 + 4x_1x_2 + 8x_1^3. \quad (3.57)$$

Substituting the values $\ddot{x}_1$, $\dot{x}_1$ and $\dot{x}_2$ from Equation (3.57) and Equation (3.52), into the new state space in Equation (3.54) we get

$$\Phi(x) = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 + 2x_1^2 \\ x_3 + 4x_1x_2 + 8x_1^3 \end{bmatrix}. \quad (3.58)$$

The input of the system has not appeared in the output derivative yet. So, the linearizing control law cannot be determined. Below we determine the third derivative of the output, i.e. $\ddot{y}$, by taking the derivative of $\ddot{y}$ from Equation (3.58).

$$\ddot{y} = \dot{\ddot{y}} = \dot{x}_3 + 4x_1\dot{x}_2 + 4\dot{x}_1x_2 + 24x_1^2\dot{x}_1 \quad (3.59)$$

Substituting for $\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$ from Equation (3.52) we get

$$\ddot{y} = 5u + 4x_1x_3 + 4x_2^2 + 8x_1^2x_2 + 24x_1^2x_2 + 48x_1^4. \quad (3.60)$$

We can now see the input, u , in the derivative and hence, we can determine the linearizing law by using Equation (3.15).

$$u = \frac{v}{a} - \frac{b}{a} = \frac{v}{5} - \frac{4x_1x_3 + 4x_2^2 + 32x_1^2x_2 + 48x_1^4}{5} \quad (3.61)$$

where $v = \ddot{y}$.

In order to express the linearizing law, u , in terms of the new state, z , we have to find an inverse transformation that will transform the x state into the z state.

This inverse transformation is given by

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 - 2z_1^2 \\ z_3 - 4z_1z_2 \end{bmatrix}. \quad (3.62)$$

The values in the inverse transformation in Equation (3.62) can be found as shown next.

From Equation (3.58) we know that

$$x_1 = z_1. \quad (3.63)$$

Now, from the expression for $\dot{x}_1$ in Equation (3.52)

$$x_2 = \dot{x}_1 - 2x_1^2. \quad (3.64)$$

From Equation (3.58) we can get the relationships

$$\dot{y} = \dot{x}_1 = z_2 \quad (3.65)$$

and

$$y = x_1 = z_1. \quad (3.66)$$

We now substitute the values from Equation (3.65) and Equation (3.66) into Equation (3.64) to get Equation (3.67).

$$x_2 = z_2 - 2z_1^2 \quad (3.67)$$

Finally, using the second equation in Equation (3.52) we can substitute for x_3 as

$$x_3 = \dot{x}_2. \quad (3.68)$$

Substituting for $\dot{x}_2$ by taking the derivative of Equation (3.64) yields

$$x_3 = \ddot{x}_1 - 4x_1\dot{x}_1. \quad (3.69)$$

Substituting for $\dot{x}_1$ from Equation (3.52) we get

$$x_3 = \ddot{x}_1 - 4x_1(x_2 + 2x_1^2). \quad (3.70)$$

Substituting for $\ddot{x}_1$ from Equation (3.54) and x_1 from Equation (3.66) and x_2 from Equation (3.67) into Equation (3.70) yields

$$x_3 = z_3 - 4z_1(z_2 - 2z_1^2) - 8z_1^3 \quad (3.71)$$

which can be simplified to

$$x_3 = z_3 - 4z_1z_2. \quad (3.72)$$

Substituting x_1 , x_2 , x_3 values from Equation (3.63), Equation (3.67) and Equation (3.72) yields the inverse transform shown in Equation (3.62). Using the inverse transform, the linearization control law can be defined as

$$\begin{aligned} u &= \frac{v}{5} - \frac{4z_1[z_3 - 4z_1z_2] + 4[z_2 - 2z_1^2]^2 + 32z_1^2[z_2 - 2z_1^2] + 48z_1^4}{5} \\ &= \frac{v}{5} + \frac{-4z_1[z_3 - 4z_1z_2] - 4[z_2 - 2z_1^2]^2 - 32z_1^2[z_2 - 2z_1^2] - 48z_1^4}{5} \\ &= \frac{v}{5} + \frac{-4z_1z_3 + 16z_1^2z_2 - 4[z_2^2 - 4z_1^2z_2 + 4z_1^4] - 32z_1^2z_2 + 64z_1^4 - 48z_1^4}{5} \\ &= \frac{v}{5} + \frac{-4z_1z_3 + 16z_1^2z_2 - 4z_2^2 + 16z_1^2z_2 - 16z_1^4 - 32z_1^2z_2 + 64z_1^4 - 48z_1^4}{5} \\ &= \frac{v}{5} + \frac{-4z_1z_3 - 4z_2^2 + 32z_1^2z_2 - 32z_1^2z_2 + 64z_1^4 - 64z_1^4}{5} \\ &= \frac{v}{5} + \frac{-4z_1z_3 - 4z_2^2}{5} \end{aligned} \quad (3.73)$$

$$\text{Therefore, } u = 0.2v - 0.8z_1z_3 - 0.8z_2^2 \quad (3.74)$$

which can also be written as

$$u = Y(z_1, \hat{z}_2, \hat{z}_3, v)\hat{P} \quad (3.75)$$

where

$$Y = [\hat{z}_2^2, z_1\hat{z}_3, v] \quad (3.76)$$

and the true parameter vector is $P = [-0.8, -0.8, 0.2]^T$.

The estimates of the z states are used in Equation (3.75) since the true states are not available for measurement. These estimates are determined using the following observer.

$$\dot{\hat{z}} = A\hat{z} + bv + L(z_1 - \hat{z}_1) \quad (3.77)$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad L = \begin{bmatrix} 30 \\ 300 \\ 1000 \end{bmatrix} \quad (3.78)$$

The value of L can be calculated by setting $\det(sI - (A - LC))$ to equal the desired characteristic equation of the observer. The reference model as shown in Equation (3.106) has triple poles at $s=-1$. We want the observer dynamics to be much faster than the model dynamics and hence, we set the poles of the observer at $s=-10$. This yields a characteristic equation of

$$s^3 + 30s^2 + 300s + 1000 = 0. \quad (3.79)$$

Setting the vector L to be $\begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix}$, we calculate the value for $\det(sI - (A - LC))$.

$$\det(sI - (A - LC)) = \det \left(s \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \right) \right) \quad (3.80)$$

$$= \det \left(s \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} l_1 & 0 & 0 \\ l_2 & 0 & 0 \\ l_3 & 0 & 0 \end{bmatrix} \right) \quad (3.81)$$

$$= \det \left(s \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} -l_1 & 1 & 0 \\ -l_2 & 0 & 1 \\ -l_3 & 0 & 0 \end{bmatrix} \right) \quad (3.82)$$

$$= \det \left(\begin{bmatrix} s+l_1 & -1 & 0 \\ l_2 & s & -1 \\ l_3 & 0 & s \end{bmatrix} \right) \quad (3.83)$$

$$= s^3 + l_1 s^2 + l_2 s + l_3. \quad (3.84)$$

The value of L can be found by comparing Equation (3.84) to Equation (3.79). Vector L is

$$L = \begin{bmatrix} 30 \\ 300 \\ 1000 \end{bmatrix} \quad (3.85)$$

where the values of l_1, l_2, l_3 are 30, 300, 1000 respectively.

In order to calculate the linearization law in Equation (3.74), the virtual control law is required. Using Figure 28 [7], the virtual control law can be defined as

$$v = rk + D_1^T (sI - \Lambda)^{-1} bv + D_2^T (sI - \Lambda)^{-1} by + d_o y \quad (3.86)$$

where

$$\Lambda = \begin{bmatrix} 0 & 1 \\ -4 & -4 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$D_1^T = [-15, -3] \quad (3.87)$$

$$D_2^T = [96, 84]$$

$$d_o = -25$$

$$k=1.$$

Figure 28: MRAC 3rd order system block diagram

The values of A and b in Equation (3.87) can be determined by converting the transfer function obtained by the cancelable zeros into its controllable canonical state space form. A general transfer function is shown in Equation (3.88), followed by its controllable canonical state space form.

$$\frac{y}{u} = \frac{1}{s^2 + a_1 s + a_0} \quad (3.88)$$

$$\dot{X} = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} X + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u = \Lambda X + bu \quad (3.89)$$

$$y = [1 \ 0]X \quad (3.90)$$

The cancelable zeros for this system are set at $s^2 + 4s + 4$. Using the concepts in Equation (3.88) – Equation (3.90), Equation (3.91) will have the controllable canonical form shown in Equation (3.92).

$$TF_{\Lambda} = \frac{1}{s^2 + 4s + 4} \quad (3.91)$$

$$\dot{X} = \begin{bmatrix} 0 & 1 \\ -4 & -4 \end{bmatrix} X + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} X \quad (3.92)$$

Therefore, for our system, $\Lambda = \begin{bmatrix} 0 & 1 \\ -4 & -4 \end{bmatrix}$ and $b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

The other values in Equation (3.87) are determined by assuming perfect linearization and setting the closed loop transfer function to equal the model transfer function given in Equation (3.106). These values are derived next.

The virtual control law in Equation (3.86) is

$$v = rk + D_1^T (sI - \Lambda)^{-1} bv + D_2^T (sI - \Lambda)^{-1} by + d_o y. \quad (3.93)$$

Assuming perfection linearization, $v = s^3 y$, Equation (3.93) can be written as

$$s^3 y = rk + D_1^T (sI - \Lambda)^{-1} b(s^3 y) + D_2^T (sI - \Lambda)^{-1} by + d_o y. \quad (3.94)$$

We let the unknown vectors D_1^T and D_2^T equal

$$D_1^T = [d_1 \quad d_2], \quad D_2^T = [d_3 \quad d_4]. \quad (3.95)$$

Substituting, Equation (3.95), Λ , I and b into Equation (3.94) we get

$$s^3 y = rk + [d_1 \quad d_2] \left[s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -4 & -4 \end{bmatrix} \right]^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} s^3 y + [d_3 \quad d_4] \left[s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -4 & -4 \end{bmatrix} \right]^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} y + d_o y. \quad (3.96)$$

Simplifying Equation (3.96) we get

$$s^3 y = rk + [d_1 \quad d_2] \begin{bmatrix} s & -1 \\ 4 & s+4 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} s^3 y + [d_3 \quad d_4] \begin{bmatrix} s & -1 \\ 4 & s+4 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} y + d_o y \quad (3.97)$$

$$= rk + [d_1 \quad d_2] \left(\frac{1}{s(s+4)+4} \right) \begin{bmatrix} s+4 & 1 \\ -4 & s \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} s^3 y + [d_3 \quad d_4] \left(\frac{1}{s(s+4)+4} \right) \begin{bmatrix} s+4 & 1 \\ -4 & s \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} y + d_o y \quad (3.98)$$

$$= rk + [d_1 \quad d_2] \begin{bmatrix} 1 \\ s \end{bmatrix} \left(\frac{1}{s(s+4)+4} \right) s^3 y + [d_3 \quad d_4] \begin{bmatrix} 1 \\ s \end{bmatrix} \left(\frac{1}{s(s+4)+4} \right) y + d_o y \quad (3.99)$$

$$= rk + (d_1 + d_2 s) \left(\frac{1}{s(s+4)+4} \right) s^3 y + (d_3 + d_4 s) \left(\frac{1}{s(s+4)+4} \right) y + d_o y \quad (3.100)$$

$$= \frac{rk(s^2 + 4s + 4) + (d_1 + d_2 s)s^3 y + (d_3 + d_4 s)y + (s^2 + 4s + 4)d_o y}{s^2 + 4s + 4}. \quad (3.101)$$

Multiplying both sides by $s^2 + 4s + 4$ yields

$$s^3 y(s^2 + 4s + 4) = rk(s^2 + 4s + 4) + (d_1 + d_2 s)s^3 y + (d_3 + d_4 s)y + (s^2 + 4s + 4)d_o y. \quad (3.102)$$

Rearranging Equation (3.102) gives

$$y[s^3(s^2 + 4s + 4) - (d_3 + d_4 s) - (s^2 + 4s + 4)d_o y - (d_1 + d_2 s)s^3] = rk(s^2 + 4s + 4) \quad (3.103)$$

$$\frac{y}{r} = \frac{k(s^2 + 4s + 4)}{[s^3(s^2 + 4s + 4) - (d_3 + d_4 s) - (s^2 + 4s + 4)d_o y - (d_1 + d_2 s)s^3]} \quad (3.104)$$

$$\frac{y}{r} = \frac{k(s^2 + 4s + 4)}{s^5 + (4 - d_2)s^4 + (4 - d_1)s^3 - d_o s^2 + (-d_4 - 4d_o)s + (-d_3 - 4d_o)}. \quad (3.105)$$

Assuming a perfect linearization, Equation (3.105) must equal the model,

$$W_m(s) = \frac{1}{(s+1)^3}. \quad (3.106)$$

By equating Equation (3.105) and Equation (3.106) we can find the values for k, d_o, d_1, d_2, d_3 and d_4 . It is easier to find the values if we multiply both the top and bottom of the model shown in Equation (3.106) by the cancelable zeros. Therefore, we get

$$W_x(s) = \frac{(s^2 + 4s + 4)}{(s+1)^3(s^2 + 4s + 4)} = \frac{(s^2 + 4s + 4)}{s^5 + 7s^4 + 19s^3 + 25s^2 + 16s + 4}. \quad (3.107)$$

By equating Equation (3.105) and Equation (3.107) we determine the values of the unknowns to be

$$\begin{aligned}
k &= 1 \\
(4 - d_2) &= 7 \Rightarrow d_2 = -3 \\
(4 - d_1) &= 19 \Rightarrow d_1 = -15 \\
-d_o &= 25 \Rightarrow d_o = -25 \\
-4d_o - d_4 &= 16 \Rightarrow d_4 = 84 \\
-4d_o - d_3 &= 5 \Rightarrow d_3 = 96.
\end{aligned} \tag{3.108}$$

Therefore, the virtual control law shown in Equation (3.86) can be computed using these values. Finally, the adaptation law for the system is given by

$$\dot{\phi} = \hat{\xi}(t)e_v \tag{3.109}$$

where

$$e_v = v - \ddot{y}_m \tag{3.110}$$

and

$$\hat{\xi}(t) = W_\phi(s)Y(z_1, \hat{z}_2, \hat{z}_3, v) \tag{3.111}$$

and

$$W_\phi(s) = \frac{25s^2 + 16s + 4}{s^5 + 7s^4 + 19s^3 + 25s^2 + 16s + 4}. \tag{3.112}$$

The filter in Equation (3.112) is derived by looking at the relationship between virtual error and the parameter error. This relationship is determined similar to that of the second order system derived in Equation (3.19) to Equation (3.36).

3.2.2 Results

The input signal used was a series of step functions with a period of 20 seconds. The initial values are all set to zero except the initial parameter estimates that vary from scenario to scenario. First we consider a low input amplitude of 0.3 with initial

parameter estimation error at 25%. Then we increase the initial parameter estimation error to 50%. This will allow us to study the effect of an increasing initial parameter estimation error on the convergence of the algorithm.

The graphs for input amplitude of 0.3 and 25% initial parameter estimation error are shown in Figure 29 and Figure 30. The parameter error is shown in Figure 29 and the output error graph is shown Figure 30.

The acronym MRAC (3^{rd} F) is used to represent this 3^{rd} order Model Reference Nonlinear Output Feedback Adaptive Control system with filter in the adaptation law algorithm.

Figure 29: MRAC (3^{rd} F): Parameter error for amp: 0.3, initial parameter estimates: -1.0,-0.6,0.25

As can be seen from Figure 29, the parameter error for all three parameters reduces to zero within 120 seconds. The output error also goes to zero as seen from Figure 30.

Figure 30: MRAC (3rd F): Output error for amp: 0.3, initial parameter estimates: -1.0,-0.6,0.25

This algorithm converges with 25% error in the initial parameter estimates. Next we increase the error to 50%. This algorithm still converges as seen from the graphs in Figure 31 and Figure 32.

Figure 31: MRAC (3rd F): Parameter error for amp: 0.3, initial parameter estimates: -1.2,-0.4,0.1

Figure 32: MRAC (3rd F): Output error for amp: 0.3, initial parameter estimates: -1.2,-0.4,0.1

Next the error was increase to 100% and the algorithm diverged. The algorithm also diverged when the input amplitude was increased to 1. Neither the 25% nor the 50% initial parameter estimation error simulations could converge for an input amplitude of 1. In fact, even when the intial parameters were set to the true values divergence occurred. As such one can conclude that the system with input set at amplitude 1 is unstable. The results obtained with the filter in the adaptation law are summarized in Table 4.

Table 4: MRAC (3rd F): Other results

Input Amplitude	Initial Parameter Estimates (True Values: (-0.8,-0.8,0.2))	Converges/Diverges	$e_p \rightarrow 0$ Time (sec)
0.3	(-1,-0.6,0.25) 25% error	Converges	120
0.3	(-1.2,-0.4,0.1) 50% error	Converges	160
0.3	(0,0,0)	Diverges	-
1	(-1,-0.6,0.25)	Diverges	-
1	(-0.8,-0.8,0.2)	Diverges	-

Like the second order system, if the input amplitude is increased high enough, the algorithm diverges even if the initial parameter estimates are set equal to the true parameter values. This can be seen from the last entry in Table 4 where the true parameter values were used with an input amplitude of 1. This scenario diverged.

Similar to the 2nd order system in Section 3.1.1, the $W_\phi(s)$ filter in Equation (3.112) is quite complex. Hence, we determine the performance of the algorithm after removing this filter. The adaptation law in Equation (3.109) then becomes

$$\dot{\phi} = Y(z_1, \hat{z}_2, \hat{z}_3, v)e_v \quad (3.113)$$

where

$$e_v = v - \ddot{y}_m. \quad (3.114)$$

With the filter removed none of the simulations tried converged. Even the 0.3 input amplitude scenarios that converged before with the filter, did not converge once the filter was removed from the adaptation law. The results obtained by removing the filter from the adaptation law for the 3rd order system are summarized in Table 5. The acronym MRAC (3rd WF) is used to represent this 3rd order Model Reference Nonlinear

Output Feedback Adaptive Control system without the filter in the adaptation law algorithm.

Table 5: MRAC (3rd WF): Other results

Input Amplitude	Initial Parameter Estimates (True Values: (-0.8,-0.8,0.2))	Converges/Diverges	$e_p \rightarrow 0$ Time (sec)
0.3	(-1,-0.6,0.25)	Diverges	-
0.3	(-0.8,-0.8,0.2)	Diverges	-
0.3	(0,0,0)	Diverges	-
1	(-1,-0.6,0.25)	Diverges	-
1	(-0.8,-0.8,0.2)	Diverges	-

It can be concluded that there is no region of convergence for this third order system with an input amplitude of 0.3 or 1 as it cannot track the desired trajectories even with the initial parameter estimates set to the true parameter values. This can be seen by the second and last simulation in the table. This indicates that for a higher order system, the filter in the adaptation law plays an important role. Without the filter, the algorithm fails to track the true values even at low amplitudes.

Chapter 4

4.0 Model Reference Nonlinear Output Feedback Adaptive Control (MRAC) - Without Observer

In this section we analyze the second model reference adaptive control algorithm proposed by Schwartz[7]. This algorithm does not use the velocity measurement for the links. Instead of using the velocity measurement it uses the model value.

We analyze this algorithm in two ways. First, we determine the effect of the normalizing factor on the stability of the algorithm. The concept of the normalizing factor has been previously used in the Normalized Least Mean Squares algorithm. The normalizing factor we use here is similar to the normalizing factor used in the augmented error method.[10]

Secondly, we analyze the effect of using all model values in the adaptation law. Currently, only the model velocity is being used. We want to analyze the effect of using the model position, model velocity and model acceleration in the adaptation law. The advantage of using the model values is that these values are always available and measurable. In fact, if need be, these values can also be computed offline before the algorithm starts.

4.1 Algorithmic Details

As mentioned previously, this algorithm uses the model velocity in the linearization law and the adaptation law instead of the link velocity. This prevents the

use of the noisy velocity signal and hence can prevent parameter drift and instability [7].

The major equations and adaptation laws of this algorithm are summarized next.

The two degree of freedom robotic manipulator system can be modelled as shown in Figure 14. The virtual control law, v , is calculated using the following equation.

$$v = \frac{-4}{s+10}v + \frac{400}{s+10}q - 44q + 4q_c \quad (4.1)$$

This virtual control law is the same as used in Equation (3.3) and the details of its derivation can be found in Equation (3.37) to Equation (3.47). Once the virtual control law is calculated, the torque applied to the robot can be calculated by

$$T_L = Y(q, \dot{q}_m, v)\hat{P} \quad (4.2)$$

where

$$Y = \begin{bmatrix} v_1 & 2v_1 + v_2 + 2v_1 \cos(q_2) - 2\dot{q}_{m1}\dot{q}_{m2} \sin(q_2) + v_2 \cos q_2 - \dot{q}_{m2}^2 \sin(q_2) \\ 0 & v_1 + v_1 \cos(q_2) + v_2 + \dot{q}_{m1}^2 \sin(q_2) \end{bmatrix} \quad (4.3)$$

and

$$\hat{P} = \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \end{bmatrix}. \quad (4.4)$$

It is important to note that in the linearization law in Equation (4.2), $\dot{q}_m$, the model velocity, is used instead of the estimated link velocity, $\hat{\dot{q}}$. The acceleration of the robot due to this torque is determined using the following equation.

$$\ddot{q} = M^{-1}(q)[T - C(q, \dot{q})] \quad (4.5)$$

where

$$M(q) = \begin{bmatrix} m_1 f^2 + 2m_2 f^2 + 2m_2 f^2 \cos(q_2) & m_2 f^2 + m_2 f^2 \cos(q_2) \\ m_2 f^2 + m_2 f^2 \cos(q_2) & m_2 f^2 \end{bmatrix} \quad (4.6)$$

and

$$C(q, \dot{q}) = \begin{bmatrix} -2m_2 f^2 \dot{q}_1 \dot{q}_2 \sin q_2 - m_2 f^2 \dot{q}_2^2 \sin q_2 \\ m_2 f^2 \dot{q}_1^2 \sin q_2 \end{bmatrix} \quad (4.7)$$

The adaptation law for the parameters is given by

$$\dot{\phi} = \frac{Y(q, \dot{q}_m, v) e_v}{(1 + \xi^T \xi)} \quad (4.8)$$

$$\text{where } \xi = W_m(s) I Y^T(q, \dot{q}_m, v) \quad (4.9)$$

$$\text{and } e_v = v - \ddot{q}_m. \quad (4.10)$$

Here we have used the normalizing factor, $(1 + \xi^T \xi)$. This is similar to the normalizing factor used in the augmented error method of MRAC. [10, page 208]. Again, it is important to note that the $\dot{q}_m$ measurement is used instead of $\dot{q}$ in the adaptation law in Equation (4.8). The $W_m(s)$ filter used in Equation (4.9) is the model transfer function and is given by

$$W_m(s) = \frac{4}{s^2 + 4s + 4}. \quad (4.11)$$

4.2 Results

The period of the input signal used is 8 seconds. The initial values are all set to zero except the initial parameter estimates that vary from scenario to scenario. We start off with a low input amplitude scenario. Here the input amplitude is set to 1 and the initial parameter estimates are set at (1,3). The algorithm converges for this scenario. The graphs for the trajectory error and the parameter error are shown in Figure 33 and 34.

Figure 33: MRAC (NWO): Trajectory error for amplitude 1, initial parameter estimates: 1,3

The acronym MRAC (NWO) is used to represent this Model Reference Nonlinear Output Feedback Adaptive Control normalized without an observer algorithm.

Figure 34: MRAC (NWO): Parameter error for amplitude 1, initial parameter estimates: 1,3

Since both the trajectory and parameter errors have reduced to zero, the algorithm has converged. Next we study the effect of increasing the initial parameter estimation error at a low amplitude. The initial parameter estimates are set at (10,8) and the input amplitude is still set at 1. This algorithm diverges. The graphs are shown in Figure 35 and Figure 36.

Figure 35: MRAC (NWO): Trajectory error for amplitude 1, initial parameter estimates: 10,8

Figure 36: MRAC (NWO): Parameter error for amplitude 1, initial parameter estimates: 10,8

Since, the trajectory error is not reducing and since the first parameter value has gone way past the expected true value, the algorithm has diverged. From these two simulations we notice that, the initial parameter estimates have an effect on the convergence of this algorithm.

Next we study the convergence region for a higher amplitude. Here we consider an amplitude of 2 with the initial parameter estimates set at (1,3). The graphs for the trajectory error and the parameter error are shown in Figures 37 and Figure 38.

Figure 37: MRAC (NWO): Trajectory error for amplitude 2, initial parameter estimates: 1,3

Figure 38: MRAC (NWO): Parameter error for amplitude 2, initial parameter estimates: 1,3

From Figure 37 and Figure 38 we can conclude that there is a region of convergence for amplitude 2 as this scenario converges.

Table 6 shows the results from various other simulations. As with the previous algorithms, the times for $e_p \rightarrow 0$ column were determined by considering any $e_p < 0.002$ as zero. Therefore, the time recorded was the first time that $e_p < 0.002$.

Table 6: MRAC (NWO): Other results

Input Amplitude	Initial Parameter Estimates (True Values: (1,2))	Converges/Diverges	$e_p \rightarrow 0$ Time(sec)
1	1,3	Converges	14
1	0.5,1	Converges	14
1	1.5,3	Converges	15
1	0.2,0.4	Converges	15
1	0,0	Converges	15
1	2,4	Converges	20
1	5,5	Converges	50
1	10,8	Diverges	-
2	1,3	Converges	250
2	1,3.1	Diverges	-
2	1,3.2	Diverges	-
3	1,3	Diverges	-
3	1,2	Converges	0
3	0,0	Diverges	-
3	1,2.1	Diverges	-
3	1.1,2	Diverges	-
3	5,5	Diverges	-
3	1.5,2.5	Diverges	-
10	1,2	Diverges	-
13	1,2	Diverges	-
15	1,2	Diverges	-

The above data indicates that for small input amplitudes, the region of convergence is greater than that of higher input amplitudes. For smaller input amplitudes, the algorithm converges even with a large initial parameter estimation error. But, if the initial parameter estimation error is too large, the algorithm diverges. For higher amplitudes, the convergence region is smaller. For instance, at input amplitude 2 and initial parameter estimates at (1,3) the algorithm converges but for initial parameter estimates at (1,3.1) the algorithm diverges. Also, if the amplitude is increased high enough, the region of convergence diminishes i.e., the algorithm fails to track even the true parameters. This can be seen from the simulations of high amplitudes like 10, 13, 15

etc. In these cases, the algorithm fails to converge even if the true parameter values, (1,2), are used as the initial parameter estimates.

We also notice that the time it takes for the parameter error to approach zero depends on the initial parameter estimation error and on the input amplitude. For input amplitude of 1, the time it takes for the parameter error to reduce to zero can take anywhere from 14-50 seconds depending on the initial parameter estimates. As the amplitude is increased it takes longer for the parameter error to reduce to zero. For instance, for amplitude 2 and the initial parameter estimates at (1,3), the $e_p \rightarrow 0$ time is 250 seconds, which is quite long compared to the 14 seconds of amplitude 1 and (1,3).

Next we study the effect of the normalizing factor on the algorithm by comparing the results of the above algorithm with the results obtained when the normalizing factor is removed. For this new non-normalized algorithm, the difference lies only in the adaptation law. The normalizing factor, $(1 + \xi^T \xi)$, has been removed. Hence, the new adaptation law becomes

$$\dot{\phi} = Y(q, \dot{q}_m, v) e_v \quad (4.12)$$

where

$$e_v = v - \ddot{q}_m. \quad (4.13)$$

Note that this adaptation law is the same as the algorithm described in Section 3.1.1 and Equations (3.50)-(3.51). The only difference is the use of the model velocity, $\dot{q}_m$, instead of the estimated velocity, $\dot{\hat{q}}$, in the adaptation law.

Again, we first consider a low amplitude of 1 with the initial parameter estimates set at (1,3). This algorithm converges with the parameter error reducing to zero in about

20 seconds. The trajectory and parameter error graphs are shown in Figure 39 and Figure 40.

Figure 39: MRAC (NNWO): Trajectory error for amplitude 1, initial parameter estimates: 1,3

The acronym MRAC (NNWO) is used to represent this Model Reference Nonlinear Output Feedback Adaptive Control non-normalized without observer algorithm.

Figure 40: MRAC (NNWO): Parameter error for amplitude 1, initial parameter estimates: 1,3

Next we consider a higher initial parameter estimation error. This time the initial parameter estimates are set to (10,8) with an input amplitude of 1. This algorithm diverges as seen from the graphs in Figure 41 and Figure 42.

Figure 41: MRAC (NNWO): Trajectory error for amplitude 1, initial parameter estimates: 10,8

Figure 42: MRAC (NNWO): Parameter error for amplitude 1, initial parameter estimates: 10,8

Parameter 1 has gone past the expected value and hence, this scenario diverges. This means, that though the algorithm has a region of convergence for lower amplitudes it starts to diverge if the initial parameter estimation error is too large. Next we consider a higher amplitude scenario. The estimates are set at (1,3) and the input amplitude is set to 2. The graphs for this scenario are shown in Figure 43 and Figure 44.

Figure 43: MRAC (NNWO): Trajectory error for amplitude 2, initial parameter estimates: 1,3

Figure 44: MRAC (NNWO): Parameter error for amplitude 2, initial parameter estimates: 1,3

The parameter values are past the expected values and hence, the algorithm has diverged. Actually, all of the amplitude 2 scenarios simulated diverged. This indicates that the region of convergence for this algorithm is smaller than that of the normalized algorithm. Several other cases were tried and these are summarized in Table 7.

Table 7: MRAC (NNWO): Other results

Input Amplitude	Initial Parameter Estimates (True Values: (1,2))	Converges/Diverges	$e_p \rightarrow 0$ Time (sec)
1	1,3	Converges	20
1	0.5,1	Converges	30
1	1.5,3	Converges	30
1	0.2,0.4	Converges	70
1	0,0	Diverges	-
1	5,5	Diverges	-
1	10,8	Diverges	-
2	1,2	Diverges	-
2	1,3	Diverges	-
2	1,3.1	Diverges	-
2	1,3.2	Diverges	-
3	1,3	Diverges	-
3	1,2	Diverges	-
10	1,2	Diverges	-

The above results indicate that for amplitude 1, it takes between 20-70 seconds for the parameter error to reduce to zero depending on the initial parameter estimation error. Comparing these results to the normalized algorithm results in Table 6, we can conclude that the normalized algorithm converges faster than this algorithm. It is important to note that the simulations with initial parameter estimates of (0,0) and (5,5) with input amplitude of 1, converged for the normalized algorithm but do not converge for this algorithm. Also, for the higher amplitudes two scenarios converged for the normalized algorithm whereas none converged for this algorithm. From this comparison it appears that the normalizing factor not only reduces the time it takes for the parameter error to reduce to zero but also increases the region of convergence.

The second issue we want to analyze is the use of model values in the adaptation law. In this algorithm variant, the link position, q , in the adaptation law is replaced by q_m

and the virtual control signal, v , is replaced by $\ddot{q}_m$. Hence, the adaptation law from Equation (4.12) now becomes

$$\dot{\phi} = Y(q_m, \dot{q}_m, \ddot{q}_m) e_v \quad (4.14)$$

$$\text{where } e_v = v - \ddot{q}_m. \quad (4.15)$$

Several simulations were executed for this scenario. We first consider the low amplitude scenarios and analyze the convergence region. The first simulation uses an input amplitude of 1 and the initial parameter estimates of (1,3). This algorithm converges as seen from the trajectory and parameter error graphs in Figures 45 and Figure 46. The acronym MRAC (MVNNWO) is used to represent this Model Reference Nonlinear Output Feedback Adaptive Control model values in the adaptation law and non-normalized without observer algorithm.

Figure 45: MRAC (MVNNWO):Trajectory error for amplitude 1, initial parameter estimates: 1,3

Figure 46: MRAC (MVNNWO): Parameter error for amplitude 1, initial parameter estimates: 1,3

This simulation indicates that there is a region of convergence for this algorithm for an input amplitude of 1. Next we analyze whether this region of convergence extends to higher initial parameter estimation errors. The next simulation we consider is with amplitude 1 and the initial parameter estimates set at (10,8). The trajectory and parameter error graphs are shown in Figure 47 and Figure 48.

Figure 47: MRAC(MVNNWO):Trajectory error for amplitude 1, initial parameter estimates: 10,8

Figure 48: MRAC(MVNNWO): Parameter error for amplitude 1, initial parameter estimates: 10,8

From the above graphs we see that as the initial parameter estimation error is increased the algorithm starts to diverge. We now consider a higher amplitude scenario. This time the amplitude is set at 2 and the initial parameter estimates are set at (1,3). This scenario also diverged. The results are shown in Figure 49 and Figure 50.

Figure 49: MRAC (MVNNWO):Trajectory error for amplitude 2, initial parameter estimates: 1,3

Figure 50: MRAC (MVNNWO): Parameter error for amplitude 2, initial parameter estimates: 1,3

Several other scenarios were also tried. The results obtained from the simulation of this algorithm are summarized in the Table 8.

Table 8: MRAC (MVNNWO): Other results

Input Amplitude	Initial Parameter Estimates (True Values: (1,2))	Converges/Diverges	$e_p \rightarrow 0$ Time (sec)
1	1,3	Converges	30
1	0.5,1	Converges	30
1	1.5,3	Converges	30
1	0.2,0.4	Converges	32
1	0,0	Converges	40
1	5,5	Diverges	-
1	10,8	Diverges	-
2	1,2	Diverges	-
2	1,3	Diverges	-
2	1,3.1	Diverges	-
2	1,3.2	Diverges	-
3	1,2	Diverges	-
3	1,3	Diverges	-
10	1,2	Diverges	-

The above results indicate that for lower amplitudes the algorithm has a wider convergence range than for higher amplitudes. For amplitude 1, the error goes to zero within 30-40 seconds depending on the initial parameter estimation error. If the initial parameter estimation error is high enough, for instance the initial estimates starting at (5,5), then the algorithm diverges. For higher amplitudes of 2 and 3 the algorithm diverges for all the cases tried.

Earlier we studied the effect of the normalizing factor and determined that it increases the region of convergence. Since, the region of convergence of this algorithm is small we want to determine if adding the normalizing factor would increase this region of convergence. Hence, we add the normalizing factor to the model regressor

$(q_m, \dot{q}_m, \ddot{q}_m)$ and analyze the results. The adaptation law in Equation (4.14) can be modified to yield

$$\dot{\phi} = \frac{Y(q_m, \dot{q}_m, \ddot{q}_m)e_v}{(1 + \xi^T \xi)} \quad (4.16)$$

where

$$\xi = W_m(s)IY^T(q_m, \dot{q}_m, \ddot{q}_m) \quad (4.17)$$

$$e_v = v - \ddot{q}_m. \quad (4.18)$$

First we analyze the scenario with amplitude 1 and initialize the parameter estimates to (1,3). The trajectory error and parameter error graphs are shown in Figure 51 and Figure 52.

Figure 51: MRAC (MVNWO): Trajectory error for amplitude 1, initial parameter estimates: 1,3

Figure 52: MRAC (MVNWO): Parameter error for amplitude 1, initial parameter estimates: 1,3

The acronym MRAC (MVNWO) is used to represent this Model Reference Nonlinear Output Feedback Adaptive Control model values in the adaptation law and normalized without observer algorithm.

As the graphs in Figure 51 and Figure 52 indicate the trajectory and parameter errors both reduce to zero in about 15 seconds. Thus, the algorithm has converged for this scenario.

Next we analyze the algorithm with a higher initial parameter estimation error. This time the initial parameters are set to (10,8) and the input amplitude is still at 1. The graphs are shown in Figure 53 and Figure 54.

Figure 53: MRAC (MVNWO): Trajectory error for amplitude 1, initial parameter estimates: 10,8

Figure 54: MRAC (MVNWO): Parameter error for amplitude 1, initial parameter estimates: 10,8

Again, the algorithm is able to reduce the trajectory and parameter errors to zero in about 25 seconds. This indicates that indeed the normalizing factor does increase the region of convergence as this scenario had diverged for the previous algorithm where the normalizing factor was not used.

Now we consider a higher amplitude scenario. An input amplitude of 2 with the initial parameter estimates set at (1,3) is considered. Figure 55 and Figure 56 show the trajectory error and the parameter error respectively.

Figure 55: MRAC (MVNWO): Trajectory error for amplitude 2, initial parameter estimates: 1,3

Figure 56: MRAC (MVNWO): Parameter error for amplitude 2, initial parameter estimates: 1,3

The trajectory and parameter errors have been reduced to zero in about 120 seconds. This re-enforces that the normalizing factor has increased the region of convergence as this scenario had diverged for the previous non-normalized algorithm.

Several other scenarios were simulated. The results of the simulations are summarized in Table 9.

Table 9: MRAC (MVNWO): Other results

Input Amplitude	Initial Parameter Estimate (True Values: (1,2))	Converges/Diverges	$e_p \rightarrow 0$ Time(sec)
1	1,3	Converges	20
1	0.5,1	Converges	15
1	1.5,3	Converges	15
1	0.2,0.4	Converges	20
1	0,0	Converges	20
1	5,5	Converges	25
1	10,8	Converges	25
2	1,3	Converges	120
2	1,3.1	Converges	100
2	1,3.2	Converges	120
2	1,2	Converges	0
3	1,3	Diverges	-
3	1,2	Converges	0
10	1,2	Diverges	-

Though the algorithm is not globally stable, the convergence region for this algorithm is quite good. It has a region of convergence greater than that of any other algorithm analyzed in this thesis. This algorithm has two advantages. First, it does not use an observer and secondly, it uses all model values which are always available and can be computed before hand if required. This simplifies the algorithm implementation greatly. The amount of time it takes for the parameter error to reduce to zero is also comparative with the other algorithms. Although, like the other algorithms, the convergence region of this algorithm also decreases as the input amplitude is increased. If the input amplitude is increased high enough, the convergence region diminishes and the algorithm fails to converge even if the true parameter values are used as the initial estimates. This can be seen from the very last entry in Table 9.

Chapter 5

5.0 Conclusion

From the results shown in Table 1, we can conclude that the robotic implementation of Jankovic's algorithm [6] as described in Section 2.1 has a region of convergence for lower input amplitudes like 1 and 2. But as the input amplitude is increased, this region of convergence diminishes. This can be seen from the scenarios where the initial parameter estimates were set to the true values and the algorithm failed to converged.

The rest of the algorithms simulated were output feedback model reference adaptive control algorithms. For comparison purposes, the different output feedback nonlinear model reference adaptive control algorithms were simulated using the same second order reference model. All the results obtained are summarized in Table 10 below. The first algorithm discussed is the algorithm that uses a linear observer to estimate the states and uses the filter in the adaptation law. This algorithm is described in Section 3.1 and is abbreviated as "F" in Table 10. The "WF" algorithm in Table 10 is the same linear observer algorithm but with the filter in the adaptation law removed. In the "NWO" algorithm, the observer was removed and the model value for the link velocity was used. This algorithm also used a normalizing factor. The normalizing factor was then removed to study its effect on the convergence region. The results of this algorithm with the normalizing factor removed are shown in the "NNWO" column. We then decided to study the effect of using all model values in the adaptation law. In this algorithm, the adaptation law used the model position, velocity and acceleration values.

The results of this simulation are shown in column "MVNNWO". Since, this algorithm had a small region of convergence compared to the others, we decided to add the normalizing factor to the adaptation law. The results of this algorithm are shown in "MVNWO" column.

The outcome is discussed in terms of whether the algorithm converged or diverged. If the parameter errors reduce to zero then the algorithm has converged otherwise it has diverged. The "C" in Table 10, indicates that the algorithm converged whereas a "D" indicates that it diverged. The first column of the table abbreviates the input amplitude, the initial parameter 1 estimate and initial parameter 2 estimates as (a,p1,p2).

Table 10: Algorithm Results

(a,p1,p2)	F	WF	NWO	NNWO	MVNNWO	MVNWO
(1,1,3)	C	C	C	C	C	C
(1,0.5,1)	C	C	C	C	C	C
(1,1.5,3)	C	C	C	C	C	C
(1,0.2,0.4)	C	C	C	C	C	C
(1,0,0)	C	C	C	D	C	C
(1,5,5)	C	C	C	D	D	C
(1,10,8)	C	D	D	D	D	C
(2,1,3)	D	C	C	D	D	C
(2,1,3.1)	D	C	D	D	D	C
(2,1,3.2)	D	C	D	D	D	C
(3,1,3)	D	D	D	D	D	D
(10,1,2)	D	D	D	D	D	D

From our analysis it can be concluded that none of the output feedback model reference adaptive control algorithms are globally stable as these algorithms did not converge for all the scenarios tried. From these model reference adaptive control algorithms, the Model Values Normalized Without Observer (MVNWO) algorithm is the

best in terms of convergence region. It converges for the most number of cases. It also has the advantage that it does not use an observer and uses the model values, which are always available and can be computed in advance, if required.

Some general conclusions can also be made about the convergence region of the six model reference adaptive control algorithms. From these simulations, it can be concluded that as the amplitude increases, the region of convergence of these algorithms decreases. If the amplitude is increased high enough then the algorithms fails to track even with the initial parameter estimates set to the true parameter values. This means that at these high amplitudes no convergence region exists and the algorithm is unstable.

Several other conclusions can be made from the analysis of the individual algorithms. For the Model Reference Nonlinear Output Feedback Adaptive Control algorithm described in Section 3.1, it can be concluded that the filter in the adaptation law does affect the region of convergence. We analyzed the effect of the filter on the convergence region for both a 2nd and a 3rd order system. For the 2nd order system, removing the filter increased the convergence region for input amplitude of 2. This can be seen from Table 2 and Table 3. With the filter, the algorithm was only able to track the true initial parameter values at amplitude 2 whereas without the filter the algorithm was able to track other initial parameter estimate values as well. Comparing the amplitude 1 scenarios we can also see that for higher initial parameter estimation error (10,8) the algorithm converged only when the filter was included in the adaptation law. Therefore, it seems that the removal of the filter increases the convergence region for higher input amplitude whereas it decreases the convergence region for lower input amplitudes. For the 3rd order system the impact of the filter can be seen from the

simulation results shown in Table 4 and Table 5. With the filter the input amplitude of 0.3 converged but once the filter was removed even the true parameters at the input amplitude of 0.3 did not converge. Hence, for the 3rd order system, the filter was important for the algorithm to converge.

We can also conclude that the normalizing factor plays an important role in the convergence region. As the simulation results indicate, the use of the normalizing factor can increase the convergence region. This can be seen from the simulation results shown in Table 6 and Table 7. Comparing Table 7 results to Table 6 results, we see that by adding the normalizing factor the two simulations of input amplitude 1 with initial parameter estimates of (0,0) and (5,5) converged. Hence, the addition of the normalized factor has increased the region of convergence. This conclusion is again supported by the simulations in Table 8 and Table 9. Comparing Table 8 and Table 9, we see that the algorithm with the normalizing factor has a greater region of convergence for both amplitudes 1 and 2. Two of amplitude 1 scenarios and three of amplitude 2 scenarios converged when the normalizing factor is added.

From the implementation and analysis of the model reference adaptive control algorithms we can conclude that the algorithms have a region of convergence that is influenced by the input amplitude, the initial parameter estimates, the normalizing factor and the filter in the adaptation law.

References

- [1] J.Craig, P.Hsu and S.Sastry, "Adaptive Control of Mechanical Manipulators", The International Journal of Robotics Research, Vol. 6, No. 2, 1987, pp. 16-28.
- [2] S.Sastry and A.Isodori, "Adaptive Control of Linearizable Systems", IEEE Transactions on Automatic Control, Vol. 34, No. 11, 1989, pp. 1123-1131.
- [3] E.Slotine and W.Li, "On the Adaptive Control of Robot Manipulators", The International Journal of Robotics Research, Vol.6, No.3, 1987, pp. 49-59.
- [4] H.Khalil and F.Esfandiari, "Semiglobal Stabilization of a Class of Nonlinear Systems Using Output Feedback", IEEE Transactions on Automatic Control, Vol. 38, No.9, 1993, pp.1412-1415.
- [5] H.Khalil, "Adaptive Output Feedback Control of Nonlinear Systems Represented by Input-Output Models", IEEE Transactions on Automatic Control, Vol. 41, No.2, 1996, pp.177-188.
- [6] M. Jankovic, "Adaptive Output Feedback Control of Non-linear Feedback Linearizable Systems", The International Journal of Adaptive Control and Signal Processing, Vol. 10, 1996, pp. 1-18.
- [7] H. Schwartz, "Model Reference Adaptive Control for Robotic Manipulators Without Velocity Measurements", The International Journal of Adaptive Control and Signal Processing, Vol. 8, 1994, pp. 279-285.
- [8] H. Schwartz, "An MRAC Output Feedback Controller for Robot Manipulators", Mediterranean Conference on Control and Automation, 1999, pp. 2293-2301.
- [9] V.V. Chalam, Marcel Dekker, "Adaptive Control Systems, Techniques and Applications", INC, 1987.
- [10] N.Annaswamy, "Stable Adaptive Systems", Prentice-Hall, Englewood Cliffs, NJ, 1988.